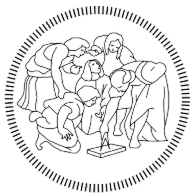
Image Analysis And Computer Vision

Prof. Vincenzo Caglioti



POLITECNICO
MILANO 1863

Christian Rossi

Student ID: 245631
Person code: 10736464

23rd December, 2024

Abstract

This project investigates the geometric and photometric properties of a rectangular parallelepiped observed in a single image taken by an uncalibrated, zero-skew camera. The furniture piece has a known width, while its depth and height are to be determined. The image features include a set of parallel lines, as well as projections of a horizontal circumference and an unknown planar curve at mid-height. Theoretical steps involve identifying the vanishing line of the horizontal plane, performing Euclidean rectification, estimating the parallelepiped's dimensions, computing the calibration matrix, and reconstructing the unknown curve in 3D. A MATLAB implementation is provided to extract features, apply theoretical methods, and visualize the reconstructed 3D model.

1 Feature Extraction

The first objective was to extract all the lines and curves shown in Figure 1.

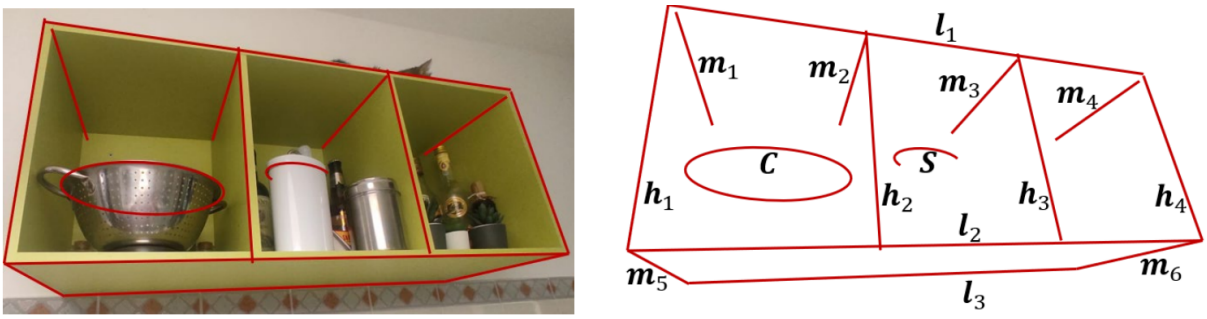


Fig. 1. Lines and curves needed to be extracted

For both lines and curves, points were extracted (two for each line and five for each curve) and parameters were computed based on these points.

To extract the required points, a preprocessing step was applied to the image, followed by the Harris feature detection algorithm implemented in MATLAB. The Harris method was chosen for its ability to localize corners with sub pixel accuracy, with an empirically determined correspondence error typically less than one pixel. The preprocessing involved the following steps:

1. Convert the image to a single channel (gray scale image).

2. Apply a Gaussian filter to smooth edges.

3. Detect edges using the Canny edge detection algorithm.

After preprocessing, the Harris algorithm was applied to the Region of Interest (ROI) containing the desired points. Points were manually refined to ensure they accurately corresponded to vertices or lines of interest.

1.1 Lines computation

Lines were computed by imposing passage through pairs of points. In two dimensions, the cross product of two points, $\mathbf{p}_1$ and $\mathbf{p}_2$, yields the line equation:

$$\mathbf{l} = \mathbf{p}_1 \times \mathbf{p}_2 \tag{1}$$

MATLAB

The script for feature extraction are in the following path:
`src/image_analysis/1_feature_extraction/1_lines_extraction/main.m`
In this script, the points within the ROI containing perpendicular objects were extracted. Points close to the desired positions were manually refined. Key variables include:

- `points`: contains all points used for line computation.
- `lines`: stores line parameters computed via cross product of point pairs.
- `curves_points`: contains points used for curve computation in the subsequent script.

The extracted points and lines are shown in Figure 2.

Fig. 2. Extracted points and lines

1.2 Curves computation

For curves, the general conic equation was used:

$$ax^2 + by^2 + cxy + dx + ey + f = 0 \tag{2}$$

Five points were used to form a system of equations. By computing the right null space of the matrix representing these equations and constraining the parameter f to one, the conic coefficients were determined. The conic matrix was then constructed as follows:

$$\mathbf{C} = \begin{bmatrix} a & \frac{b}{2} & \frac{d}{2} \\ \frac{b}{2} & c & \frac{e}{2} \\ \frac{d}{2} & \frac{e}{2} & f \end{bmatrix} \tag{3}$$

MATLAB

The script for feature extraction are in the following path:
`src/image_analysis/1_feature_extraction/2_curves_extraction/main.m`
This script imports the points identified in `lines_detection` and applies similar steps for curve computation. Singular Value Decomposition (SVD) was used to compute the right null space of matrix $\mathbf{A}$, representing the system of equations, to ensure robust and accurate results. The extracted curves are shown in Figure 3.

Fig. 3. Extracted curves

2 Image Analysis

This section outlines the steps required to analyze the image, including the processes for both camera calibration and localization. The operations performed here are critical for extracting accurate features, computing calibration parameters, and achieving precise localization of the camera in its environment.

2.1 Horizontal vanishing line identification

The goal of this task is to determine the vanishing line, denoted as $\mathbf{l}'_{\infty}$, of the horizontal plane. This plane is defined by the length and depth of the object shown in Figure 4.

Fig. 4. Image of the scene

The horizontal plane is characterized by two sets of parallel lines: $\mathbf{l}_2$ and $\mathbf{l}_3$ for the length, and $\mathbf{m}_5$ and $\mathbf{m}_6$ for the depth.

To determine the vanishing points, we compute the intersections of these parallel lines in the image space. Parallel lines in the real world appear to converge at vanishing points in the image, as stated by the following theorem: the image of a set of parallel lines $\mathbf{n}_i$ is a set of concurrent lines $\mathbf{n}'_i$ that intersect at a common point $\mathbf{p}'$, referred to as the vanishing point of the direction of lines $\mathbf{n}_i$.

For the length and depth directions, the vanishing points are calculated as:

$$\mathbf{p}_l = \mathbf{l}_2 \times \mathbf{l}_3 \tag{4}$$

$$\mathbf{p}_m = \mathbf{m}_5 \times \mathbf{m}_6 \tag{5}$$

The coordinates of the vanishing points are expressed as:

$$\mathbf{p}_l = \begin{bmatrix} x_l \\ y_l \\ w \end{bmatrix} \qquad \mathbf{p}_m = \begin{bmatrix} x_m \\ y_m \\ w \end{bmatrix} \tag{6}$$

The vanishing line for the horizontal plane, $\mathbf{l}'_\infty$, is defined as the line connecting these two vanishing points. This line is computed as the cross product of $\mathbf{p}_l$ and $\mathbf{p}_m$:

$$\mathbf{l}'_\infty = \mathbf{p}_m^T \times \mathbf{p}_l^T = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{7}$$

MATLAB The script for identifying vanishing points and the vanishing line is located at: `src/image_analysis/2_theory_implementation/1_vanishing_line/main.m`
In this script, previously extracted lines were imported, and vanishing points for all directions

were computed, as they may be useful for later analysis. The identified vanishing points are illustrated in Figure 5, where $\mathbf{p}_l$, $\mathbf{p}_m$, and $\mathbf{p}_h$ (the vanishing point for the vertical facade) are labeled as 1, 2, and 3, respectively.
The calculated coordinates are as follows:

$$\mathbf{p}_l = \begin{bmatrix} 4,136.52 \\ 651.09 \\ 1 \end{bmatrix} \qquad \mathbf{p}_m = \begin{bmatrix} 608.91 \\ 876.13 \\ 1 \end{bmatrix} \qquad \mathbf{p}_h = \begin{bmatrix} 664.69 \\ -1,370.49 \\ 1 \end{bmatrix} \tag{8}$$

To compute the horizontal line at infinity, the line passing through $\mathbf{p}_l$ and $\mathbf{p}_m$ was determined using the cross product. The resulting line, shown in Figure 5, has the following parameters:

$$\mathbf{l}'_\infty = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} -225.03 \\ -3,527.60 \\ 3,227,659.25 \end{bmatrix} \tag{9}$$

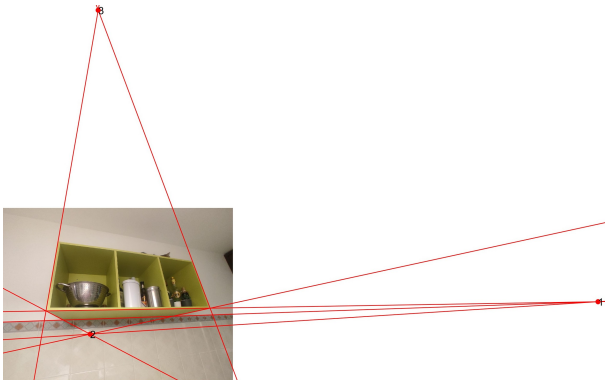


Fig. 5. Vanishing points for depth and length and vanishing line

2.2 Image rectification

The objective of this section is to establish a rectification mapping for the horizontal plane analyzed earlier and subsequently compute the object's depth. This process is implemented using a stratified approach, starting with affine rectification followed by metric rectification.

From geometric theory, the line at infinity identified in the previous step must be mapped to the real line at infinity in the scene. Specifically, the line $\mathbf{l}'_\infty$ in the image must be transformed back to $\mathbf{l}_\infty$:

$$\mathbf{l}'_\infty = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \xrightarrow{\text{mapped back to}} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \mathbf{l}_\infty \tag{10}$$

To achieve this, we construct a homography matrix that aligns $\mathbf{l}'_\infty$ with $\mathbf{l}_\infty$:

$$\mathbf{H}_{\text{rect}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a & b & c \end{bmatrix} \tag{11}$$

By applying this transformation matrix to all pixels in the image, we obtain a rectified version of the scene.

Using the stratified approach, we proceed to the metric rectification step, which imposes orthogonality constraints to achieve a full metric reconstruction of the image.
We select two pairs of orthogonal lines: one pair along the lower side of the object ($\mathbf{l}_2$ and $\mathbf{m}_5$) and another pair intersecting near the top ($\mathbf{d}_1$ and $\mathbf{d}_2$). From these pairs, we compute the orthogonality conditions and populate a matrix $\mathbf{A}$. Each row of $\mathbf{A}$ corresponds to the constraints imposed by one pair of lines, computed as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{l}_x \cdot \mathbf{m}_x & \mathbf{l}_x \cdot \mathbf{m}_y + \mathbf{l}_y \cdot \mathbf{m}_x & \mathbf{l}_y \cdot \mathbf{m}_y \\ \cdots & \cdots & \cdots \end{bmatrix} \tag{12}$$

After constructing $\mathbf{A}$ using the selected pairs of lines, we compute the right null space using Singular Value Decomposition (SVD) for accuracy and robustness. The last row of the resulting $\mathbf{V}$ matrix contains the coefficients for the conic matrix $\mathbf{S}$, expressed as:

$$\mathbf{S} = \begin{bmatrix} \mathbf{V}(1,3) & \mathbf{V}(2,3) \\ \mathbf{V}(2,3) & \mathbf{V}(3,3) \end{bmatrix} \tag{13}$$

To compute the homography matrix, we first find the transformation matrix $\mathbf{G}$, computed as:

$$\mathbf{A} = \mathbf{U}\sqrt{\mathbf{D}}\mathbf{V}^T \tag{14}$$

Here, $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ are obtained by performing SVD on $\mathbf{S}$.
The final metric rectification homography matrix is then given by:

$$\mathbf{H}_{\text{metric}} = \begin{bmatrix} \mathbf{G} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix}^{-1} \tag{15}$$

Applying $\mathbf{H}_{\text{metric}}$ to the rectified image transforms it into a fully metric rectified image.

The final step is to compute the object's depth using the metric rectified image. Assuming the object's length is known and unitary, we measure both the length and depth directly on the rectified image using Euclidean distances between corresponding points.
The real-world depth is calculated as:

$$\text{real depth} = \frac{\text{depth}}{\text{width}} \cdot 1 \tag{16}$$

MATLAB The MATLAB script for finding the vanishing points and lines is located in the following path:
`src/image_analysis/2_theory_implementation/2_rectification/main.m`

For the initial rectification, the line at infinity for the horizontal plane found earlier was used. The rectification matrix obtained is:

$$\mathbf{H}_{\text{rect}} = \begin{bmatrix} 1 & 0 & -0.0000697 \\ 0 & 1 & -0.0010929 \\ 0 & 0 & 1 \end{bmatrix} \tag{17}$$

The rectified image obtained using this homography matrix is shown in Figure 6.

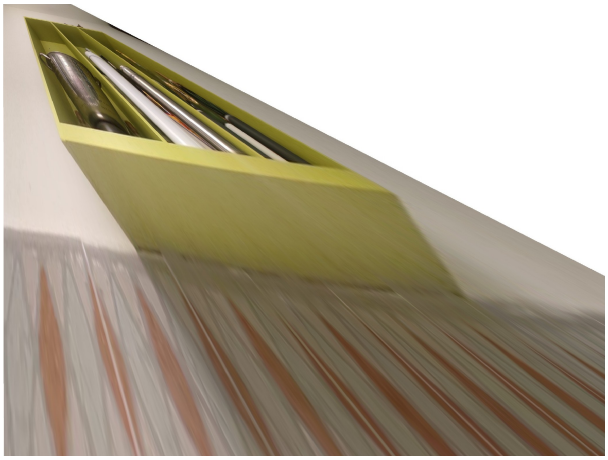


Fig. 6. Affine rectification of the image

For metric rectification, two pairs of orthogonal lines were selected in the original scene, as shown in Figure 7.

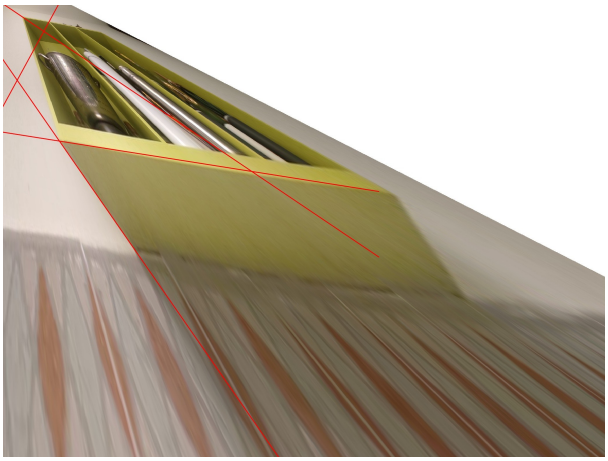


Fig. 7. Pairs of orthogonal lines

After imposing orthogonality conditions on these lines, the matrix $\mathbf{A}$ was constructed, and the right null space was computed via SVD. This operation allowed the reconstruction of the

matrix $\mathbf{S}$, on which another SVD was performed to derive $\mathbf{G}$. Finally, by inverting the matrix comprising $\mathbf{G}$ and ones on the diagonal, the metric rectification matrix was obtained:

$$\mathbf{H}_{\text{met}} = \begin{bmatrix} 1.4041 & -0.6193 & 0 \\ -0.6193 & 1.8519 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{18}$$

The rectified image obtained using this matrix is shown in Figure 8.

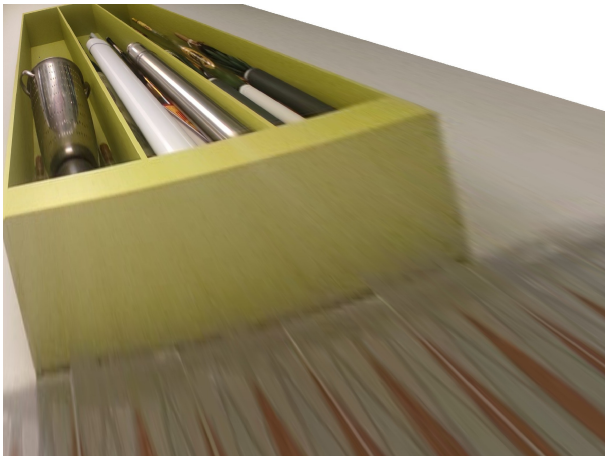


Fig. 8. Pairs of orthogonal lines

After applying the transformation to both the points on the base and two lines for verifying orthogonality, the depth-to-length ratio was found to be:

$$\text{real depth} = \frac{\text{depth}}{\text{width}} \cdot 1 = \frac{\text{4,174.11}}{\text{19,374.39}} = 0.35 \tag{19}$$

The checked lines confirmed orthogonality, achieving the desired rectification and depth calculation results.

2.3 Calibration

The aim of this section is to compute the calibration matrix $\mathbf{K}$, which encapsulates the intrinsic parameters of the camera. These parameters are essential for mapping 3D scene points into the 2D image plane. The calibration matrix $\mathbf{K}$ is defined as:

$$\mathbf{K} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{20}$$

Here:

- f_x and f_y are the effective focal lengths along the x and y axes, respectively.

- u_0 and v_0 represent the coordinates of the principal point, which is the optical center in the image.

- s is the skew parameter, representing the angle between the x and y axes of the image. For most modern cameras, the axes are orthogonal, and the skew parameter is assumed to be $s = 0$.

The computation of $\mathbf{K}$ relies on the Image of the Absolute Conic (IAC), represented by the symmetric matrix $\boldsymbol{\omega}$:

$$\boldsymbol{\omega} = \begin{bmatrix} a & 0 & b \\ 0 & 1 & c \\ b & c & d \end{bmatrix} \tag{21}$$

The IAC encodes the intrinsic parameters of the camera and is pivotal for recovering $\mathbf{K}$.

To estimate $\boldsymbol{\omega}$, geometric constraints are imposed based on the vanishing points, line at infinity, and the combined homography matrix. hese elements include:

1. *Vanishing points*: $\mathbf{p}_h$, $\mathbf{p}_m$, and $\mathbf{p}_l$.

2. *Metric homography matrix*: complete homography matrix from original image to the metric reconstructed one.

3. *Line at the infinity*: the line at infinity of the horizontal plane.

The constraints imposed on $\boldsymbol{\omega}$ are formulated as follows:

$$\begin{cases} \left[\mathbf{l}_\infty^T\right] \times \boldsymbol{\omega} \times \mathbf{p}_h = 0 \\ \mathbf{p}_m^T \times \boldsymbol{\omega} \times \mathbf{p}_l = 0 \\ \mathbf{h}_1^T \times \boldsymbol{\omega} \times \mathbf{h}_2 = 0 \\ \mathbf{h}_1^T \times \boldsymbol{\omega} \times \mathbf{h}_1 - \mathbf{h}_2^T \times \boldsymbol{\omega} \times \mathbf{h}_2 = 0 \end{cases} \tag{22}$$

Here, $\mathbf{h}_1$ and $\mathbf{h}_2$ are the first and second columns of the homography matrix $\mathbf{H}$.
These constraints form a system of linear equations that is solved to determine the elements a, b, c, and d of the IAC matrix $\boldsymbol{\omega}$. Once the IAC matrix $\boldsymbol{\omega}$ is determined, the intrinsic camera parameters are computed using the following formulas:

$$\begin{cases} f_x = \sqrt{d - \alpha^2 \cdot u_0^2 - v_0^2} \\ f_y = \frac{f_y}{\alpha} \\ u_0 = -\frac{b}{\alpha^2} \\ v_0 = -c \end{cases} \tag{23}$$

Here, $\alpha = \sqrt{a}$. With these parameters, we can construct the final calibration matrix $\mathbf{K}$.

MATLAB The calibration matrix computation script is located at:
`src/image_analysis/2_theory_implementation/3_calibration/main.m`
The workflow in the script is as follows:

1. Import the vanishing points, the line at infinity for the horizontal plane, and the transformation matrix $\mathbf{H}$.
2. Normalize the inputs to mitigate numerical inaccuracies.
3. Formulate the system of equations based on the constraints in Equation (22).
4. Solve the system using linear regression to account for measurement noise and numerical imprecisions.

The solution of the system returned the following IAC matrix $\boldsymbol{\omega}$:

$$\boldsymbol{\omega} = \begin{bmatrix} 0.3625 & 0 & -368.1149 \\ 0 & 1 & -622.9999 \\ -368.1149 & -622.9999 & 1,214,856.9057 \end{bmatrix} \tag{24}$$

Using this IAC, the intrinsic camera parameters were computed, resulting in the calibration matrix $\mathbf{K}$:

$$\mathbf{K} = \begin{bmatrix} 1,117.7730 & 0 & 1,015.5215 \\ 0 & 672.9779 & 622.9999 \\ 0 & 0 & 1 \end{bmatrix} \tag{25}$$

2.4 Facade reconstruction

The key idea here is to identify the image of the circular points by intersecting the line at infinity for the vertical plane, $\mathbf{l}'_{v\infty}$, with the IAC $\boldsymbol{\omega}$. These circular points are crucial for rectifying the image and performing metric measurements.

To determine the image of the circular points, we impose the condition that the vanishing line of the vertical plane passes through the IAC. This leads to the following system of equations:

$$\begin{cases} \boldsymbol{\omega}_{11}x^2 + 2\boldsymbol{\omega}_{12}xy + \boldsymbol{\omega}_{22}y^2 + 2\boldsymbol{\omega}_{13}x + 2\boldsymbol{\omega}_{23}y + \boldsymbol{\omega}_{33} = 0 \\ \mathbf{l}'_{\infty}(1)x + \mathbf{l}'_{\infty}(2)y + \mathbf{l}'_{\infty}(3) = 0; \end{cases} \tag{26}$$

By solving this system, the two circular points, denoted as $\mathbf{I}$ and $\mathbf{J}$, are obtained. These points lie on the conic defined by the IAC.

Using the circular points, the image of the dual conic $\mathbf{imDCCP}$ is computed as:

$$\mathbf{imDCCP} = \mathbf{I}\mathbf{J}^T + \mathbf{J}\mathbf{I}^T \tag{27}$$

This dual conic is then used to compute the homography that maps the current image to a canonical position, where metric properties like distances can be measured.

To align the image to the canonical position, the following steps are performed:

1. Perform Singular Value Decomposition (SVD) on $\mathbf{imDCCP}$.

2. Compute the homography matrix $\mathbf{H}$ as:

$$\mathbf{H} = \left(\mathbf{U}\sqrt{\mathbf{D}}\right)^{-1} \tag{28}$$

The homography $\mathbf{H}$ is applied to the image and the selected points used for measuring dimensions.

After applying the transformation, the height and length of the facade are computed as the Euclidean distance between respective pairs of points. The real height of the facade is determined using the ratio:

$$\text{real height} = \frac{\text{height}}{\text{length}} \cdot 1 \tag{29}$$

MATLAB The MATLAB implementation for this process is located at:
`src/image_analysis/2_theory_implementation/4_vertical_reconstruction/main.m`
The script includes the computation of the the line at infinity $\mathbf{l}'_{v\infty}$, and importing the previously computed IAC $\boldsymbol{\omega}$. Those variables are used to solve the system to recover the circular points:

$$\mathbf{I} = \begin{bmatrix} 2,500.5769 - 1,733.0292i \\ -301.4896 - 1,009.1154i \\ 1 \end{bmatrix} \qquad \mathbf{J} = \begin{bmatrix} 2,500.5769 + 1,733.0292i \\ -301.4896 + 1,009.1154i \\ 1 \end{bmatrix} \tag{30}$$

From these, $\mathbf{imDCCP}$ is derived as:

$$\mathbf{imDCCP} = \begin{bmatrix} 0.7351 & -0.5661 & 0.0006 \\ -0.5661 & -0.2098 & -0.00007 \\ 0.0006 & -0.00007 & 0.0000002 \end{bmatrix} \tag{31}$$

Finally, the homography matrix is computed:

$$\mathbf{H} = \begin{bmatrix} -0.9057 & 0.4239 & -0.0005 \\ -0.6152 & -1.3146 & 0.0005 \\ -0.0003 & 0.0006 & 0.9999 \end{bmatrix} \tag{32}$$

Applying the homography yielded the rectified facade, as shown in Figure 9.
For better visualization, the image was rotated, and points were manually extracted for dimension measurements (Figure 10). The ratio between height and length is:

$$\text{real height} = \frac{\text{height}}{\text{length}} \cdot 1 = \frac{373.1287}{1,570.0815} = 0.2376 \tag{33}$$

Fig. 9. Reconstructed (and cropped) facade

Fig. 10. Extracted length and height

The angle between the computed lines is $86.1263°$, indicating a small deviation from orthogonality. This discrepancy arises due to:

- Numerical imprecision in point selection.
- Resolution limitations of the image.
- Numerical instability during computation.

2.5 Curve identification

The goal of this step was to extract at least a dozen points from the unknown curve $\mathbf{S}$. Two potential methods were considered for this task:

1. *Harris detection algorithm*: a popular feature detection method that identifies corner-like features on the image.
2. *Random sampling from the conic equation*: this approach extracts points directly from the conic equation derived during the feature detection step.

The second method was chosen, as it ensures that all selected points lie precisely on the same curve $\mathbf{S}$. Furthermore, this approach facilitates the approximation of points located on non-visible sections of the curve, offering a more complete representation of the geometry.

The conic equation provided a parametric framework for identifying candidate points on the curve. By sampling random points, the algorithm verified their validity by ensuring they satisfied the mathematical definition of the curve. This iterative process was repeated until the desired number of points (twelve) was successfully identified. The selected method not

only guarantees consistency in point placement but also addresses the challenges posed by occluded or partially visible regions of the object.

MATLAB The MATLAB implementation for curve point estimation is located in:
`src/image_analysis/2_theory_implementation/5_curve_point_estimation/main.m`
In this script, the number of points to be extracted is specified, and a fixed seed is set for the random number generator to ensure the reproducibility of results. The function `generate_conic_points` is used to sample points from the conic equation. Each sampled point is verified to ensure it satisfies the equation of the curve $\mathbf{S}$. Points that do not meet this criterion are discarded, and the process is repeated until the required number of points is obtained. Once the points are identified, they are visualized in the original image coordinate space to confirm their placement. The extracted points are illustrated in Figure 11.

Fig. 11. Extracted points for the curve $\mathbf{S}$

While the points are extracted in the original image coordinate space, their locations can be seamlessly transformed into alternate coordinate systems using the corresponding homography matrix $\mathbf{H}$.

2.6 Localization

In this task, the goal was to determine the relative position of the camera with respect to a reference frame located at the bottom-left corner of the vertical facade shown in Figure 12. The camera's position was computed using the known dimensions of the vertical facade, including its length-height ratio, and the homographies required for metric reconstruction. With the calibration matrix $\mathbf{K}$, the image data, and the facade's geometric properties, the camera's position was obtained. The first step was to identify the mapping between the reconstructed image points in metric space and the new world reference frame defined by the plane π. With this, any point in the rectified image could be mapped to the world frame using the equation:

$$\left[\mathbf{R}_\pi \mid \mathbf{o}_\pi\right] \tag{34}$$

Fig. 12. Vertical facade vertices that forms a plane

Here, $\mathbf{X}_\pi$ represents a point on the plane in homogeneous coordinates:

$$\mathbf{X}_\pi = \begin{bmatrix} 0 \\ y \\ z \\ w \end{bmatrix} \tag{35}$$

The transformation from 3D points in the world space $\mathbf{X}$ to image points $\mathbf{x}$ is given by the projection matrix $\mathbf{P}$:

$$\mathbf{P} = \Big[\mathbf{KR} \mid -\mathbf{KR}_o\Big] \tag{36}$$

where $\mathbf{R}$ is the rotation matrix describing the orientation of the camera with respect to the world frame, and $\mathbf{o}_\pi$ is the camera's position in Cartesian coordinates. The relationship between image points and 3D world points is:

$$\mathbf{x} = \mathbf{PX} \tag{37}$$

Using this projection mapping, the following relationship was derived:

$$U = \Big[\mathbf{K} \mid \mathbf{0}\Big] \begin{bmatrix} \mathbf{j}_\pi & \mathbf{k}_\pi & \mathbf{o}_\pi \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y & z & w \end{bmatrix} \tag{38}$$

Here, the parameters $\mathbf{j}_\pi$, $\mathbf{k}_\pi$, and $\mathbf{o}_\pi$ represent the basis vectors and the origin of the world frame. These parameters were derived from the homography $\mathbf{H}$, which maps real-world points to image coordinates:

$$\mathbf{K}\Big[\mathbf{j}_\pi \mid \mathbf{k}_\pi \mid \mathbf{o}_\pi\Big] = \mathbf{H} \tag{39}$$

Given the known shape and size of the horizontal face in the world, $\mathbf{H}$ was known, allowing the calculation of:

$$\Big[\mathbf{j}_\pi \mid \mathbf{k}_\pi \mid \mathbf{o}_\pi\Big] = \mathbf{K}^{-1}\mathbf{H} \tag{40}$$

Next, the rotation matrix $\mathbf{R}$ that describes the orientation of the plane π relative to the camera was computed using the following set of equations:

$$\begin{cases} \lambda = \frac{1}{|\mathbf{K}^{-1}\mathbf{h}_1|} \\ \mathbf{j}_\pi = \mathbf{K}^{-1}\mathbf{h}_1\lambda \\ \mathbf{k}_\pi = \mathbf{K}^{-1}\mathbf{h}_2\lambda \\ \mathbf{i}_\pi = \mathbf{j}_\pi \times \mathbf{k}_\pi \\ \mathbf{o}_\pi = \mathbf{K}^{-1}\mathbf{h}_3\lambda \end{cases} \tag{41}$$

Here, $\mathbf{h}_1$, $\mathbf{h}_2$, and $\mathbf{h}_3$ are the columns of the homography $\mathbf{H}$. The scalar λ is a normalization factor ensuring the proper scale for the vectors.

Due to noise in the data, the rotation matrix $\mathbf{R}$ may not be perfectly orthogonal. To resolve this, SVD was applied to approximate an orthogonal matrix for $\mathbf{R}$, yielding:

$$\mathbf{R} = \mathbf{U}\mathbf{V} \tag{42}$$

MATLAB The MATLAB implementation of this procedure can be found in the script located at:
`src/image_analysis/2_theory_implementation/6_localization/main.m`
In this script, all the necessary variables mentioned above were imported, and the metric transformation was applied to the image points.

Once the transformation was applied, the world reference frame was defined, and the mapping between the world and image coordinates was computed, considering the translation from the image to the world frame.

With the homography in hand, the previously derived relationships were used to extract the camera's rotation and position matrices. The camera's position relative to the reference frame was found to be:

$$\text{cameraPosition} = \begin{bmatrix} 101.4514 \\ -182.9117 \\ 834.2182 \end{bmatrix} \tag{43}$$

Similarly, the camera's rotation (after SVD orthogonalization) relative to the reference frame was:

$$\text{cameraRotation} = \begin{bmatrix} 0.9022 & 0.0430 & 0.4292 \\ -0.1053 & -0.9430 & 0.3157 \\ 0.4183 & -0.3300 & -0.8463 \end{bmatrix} \tag{44}$$

Figures 13 show different views of the camera position relative to the object in the world frame. In these visualizations, the camera is represented by a red circle, and the three axes indicate the following orientations:

- *Red* (x): points to the horizontal direction of the camera.

- *Green* (y): points downward along the vertical axis of the camera.
- *Blue* (z): points in the direction the camera is facing.

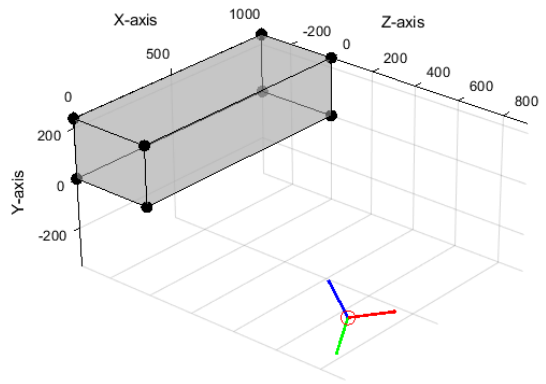


Fig. 13. View of the camera position with respect to the object in world frame

It's important to note that the reference frame used here differs slightly from the one initially suggested, as the y and z axes have been swapped. This change is purely for visualization purposes, chosen for convenience in representing the main facade as a plane. Despite this adjustment, the results remain consistent. The swap of these two axes does not affect the final outcome, as the fundamental relationships between the camera and the object are preserved.

3 Visualization

The final task involves visualizing the rectified curve and presenting various views of the reconstructed three-dimensional model of the object.

3.1 Rectified curve visualization

To visualize the rectified curve, the previously computed homography matrices were utilized to transform specific points along the curve. These transformed points were then used to reconstruct the curve's matrix. Once the matrix of the rectified curve was determined, the curve was plotted directly onto the rectified image.

MATLAB The MATLAB implementation for rectified curve visualization is available in the script located at:
`src/image_analysis/3_visualization/1_rectified_curve_visualization/main.m`
This script leverages the data already computed during earlier steps, including the metric rectification for all points of the image. The following steps summarize the process:

1. Import the necessary data, including homography matrices and the curve $\mathbf{S}$ points.
2. Apply the computed transformations to the image and the curve points to rectify them.

3. Use the `conic_extractor` function to derive the rectified curve's matrix from the transformed points.

4. Plot the resulting rectified curve onto the metric rectified image.

The result, as shown in Figure 14, demonstrates the rectified curve.

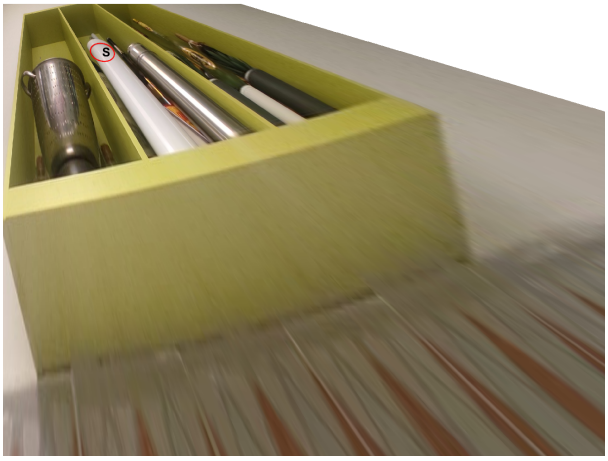


Fig. 14. Rectified curve plotted in the metric rectified image

The rectified curve $\mathbf{S}$ closely resembles a perfect circumference, which aligns with the expectations and validates the accuracy of the applied transformations.

3.2 Three-dimensional visualization

The second part of the task involved visualizing the reconstructed three-dimensional model of the object along with the camera's position and orientation.

MATLAB The MATLAB implementation for this task is available in the script located at: `src/image_analysis/3_visualization/2_three_dimensional_visualization/main.m` This script utilizes the parameters and transformations computed in previous steps to create a comprehensive three-dimensional representation. The process includes:

1. Plotting the three-dimensional reference frame, incorporating the object and the camera position.

2. Ensuring that the camera's orientation and axes align with the computed parameters.

3. Capturing visualizations from multiple perspectives to illustrate the spatial relationships between the object and the camera.

The resulting views, shown in Figure 19, provide insights into the reconstructed model from different angles.

The camera's position and orientation relative to the object are consistent with the initial image, validating the accuracy of the reconstruction process. These views highlight how the computed parameters successfully describe the spatial relationship between the object and the camera in the world reference frame.

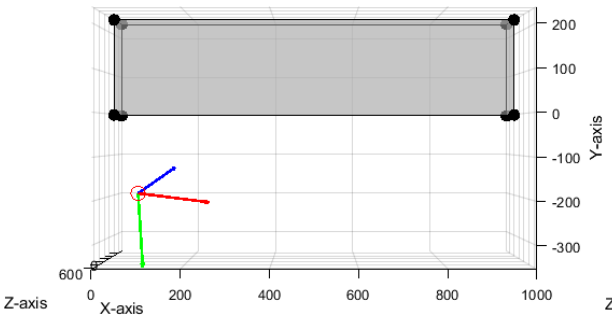


Fig. 15. Front view

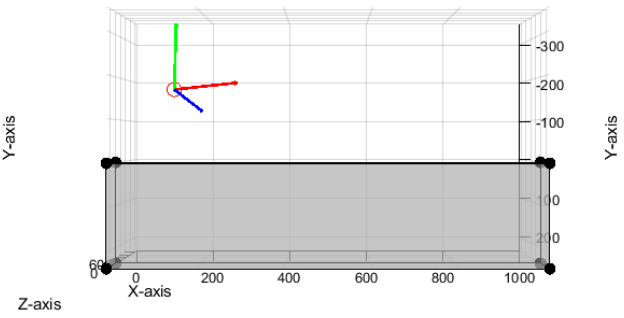


Fig. 16. Rear view

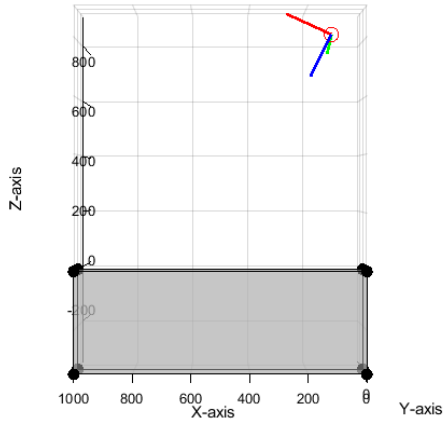


Fig. 17. Top view

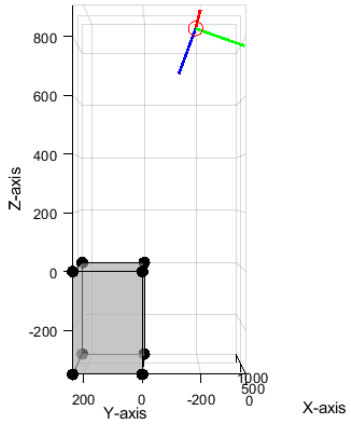


Fig. 18. Lateral view

Fig. 19. Multiple views of the camera position and orientation with respect to the reconstructed object in the world frame

4 Additional Information

All results presented in this report are approximations, aimed at improving readability and simplifying the understanding of the process. For more accurate results, the full values and computations are available through the provided scripts and saved variables. These can be accessed by running the relevant scripts or by checking the stored variables directly.
The folder structure and associated scripts are organized as follows:

```
src
├── image_analysis ..... Scripts designed to solve the problems described
│   ├── 1_feature_extraction
│   │   ├── 1_lines_extraction
│   │   └── 2_curves_extraction
│   ├── 2_theory_implementation
│   │   ├── 1_vanishing_line
│   │   ├── 2_rectification
│   │   ├── 3_calibration
│   │   ├── 4_vertical_reconstruction
│   │   ├── 5_curve_points_estimation
│   │   └── 6_localization
│   └── 3_visualization
│       ├── 1_rectified_curve_visualization
│       └── 2_three_dimensional_visualization
├── images ..... Directory containing the resulting images
├── utils ..... Utility functions shared by various scripts
└── variables ... Folder containing variables with the computed results
```

Each sub-folder within `1_feature_extraction`, `2_theory_implementation`, and `3_visualization` contains a single script, `main.m`. Running this script will compute all necessary variables and transformations required for each corresponding step of the process.

The primary folder to focus on for executing all the scripts is `src`. It contains essential sub-folders, such as those for images, variables, and utility functions, which are needed to perform the computations and generate the results.

All the information used to complete this homework was sourced from the recommended book [1] and the lecture slides [2].

5 References

1. Zisserman, A. & Hartley, R. *Multiple View Geometry in Computer Vision* (Cambridge University Press, Cambridge, 2004).
2. Caglioti, V. *Single View Geometry* (Politecnico di Milano, 2024).