POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Image Analysis & Computer Vision Homework 2024/25

Author: **Paolo Riva - 10938975**

Professor: **Vincenzo Caglioti**

Academic Year: 2024-2025

Contents

1 | Introduction

This project aims at applying image analysis techniques to a given scene with known forms and dispositions in order to calculate spatial elements, rectification mappings and a calibration matrix, to determine the positions of objects and the camera placement in a given image.

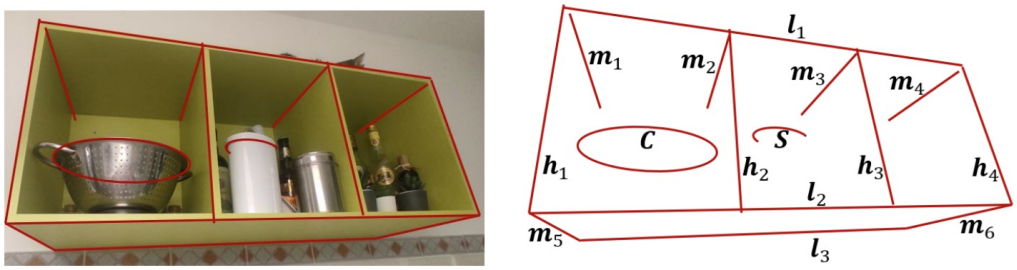

Figure 1.1: Presented scene.

The image visible in Figure 1.1 displays a scene with a rectangular parallelepiped, taken by an uncalibrated, zero skew camera, whose calibration matrix depends on four unknown parameters. Several sets of lines are visible and their images are extracted, among with images of a circumference and an unknown horizontal curve.
The report is split as such:

- **Theory**: Theoretical solutions to the given problems, with assumptions and solutions based on the given setup.
- **Practice**: MATLAB implementation of the theoretical solutions, along with preprocessing, feature extraction techniques exploited and resuting plots.

All solutions, plots and data computed in the following document can be directly checked and evaluated through MATLAB via the provided live script file "**homework.mlx**". All results and useful data is provided in *".mat"* files along with the live script.

2 | Theory

Theory Requests

1. From the l_i and m_j lines, find the vanishing line l'_∞ of the horizontal plane.
2. Using the results of the previous point, find a (Euclidean) rectification mapping H_R for a horizontal plane (e.g., the lower horizontal face of the parallelepiped), and compute the depth m of the parallelepiped.
3. From the results of the previous points, use the lines h_i to find the calibration matrix K.
4. Using the results of the previous points, determine the height h of the parallelepiped.
5. Using S and the results of previous points, compute the X-Y coordinates of a dozen points (at your choice) of the unknown horizontal curve.
6. Using K, localize the camera with respect to the parallelepiped.

2.1. Request 1: Vanishing Line l'_∞

The vanishing line l'_∞ represents the image of the line at infinity l_∞. This line serves as the collection of all points at infinity.

When two parallel lines intersect, their meeting point lies at infinity in the direction they share. The vanishing point is the projection of this point at infinity onto the image plane. Consequently, since parallelism isn't regularly preserved in a prospective image, to identify the vanishing line, we only need to compute two vanishing points from lines that would be parallel in a canonical 3D space.

Since all l_i lines are parallel to one another, and the same holds for the m_j lines, we can select two lines from each set, choosing l_1 and l_2 from the l_i group, and m_1 and m_2 from the m_j group.

To compute the vanishing points v_l and v_m, we determine the intersections of l_1 with l_3, and m_1 with m_6 to then derive the vanishing line l'_∞ from the cross product of the two vanishing points in homogeneous coordinates:

$$l_1^T \times v_l = 0;$$

$$l_3^T \times v_l = 0;$$

$$m_1^T \times v_m = 0;$$

$$m_6^T \times v_m = 0;$$

$$l'_\infty = v_l \times v_m;$$

Specifically, in this case, the vanishing line represents the locus of the vanishing points for the planar section of possible parallel lines formed by l_1 and l_3, and the planar section of possible parallel lines formed by m_1 and m_6 of the parallelepiped. These two sections define the perspective projections of the parallel lines in their respective groups, where each set of parallel lines shares a common vanishing point. The intersection of these vanishing points, as computed from the cross product of their homogeneous coordinates, defines the vanishing line l'_∞. This line acts as the boundary where the projection of all lines parallel to the two sets—l_i and m_j—converge, marking the transition to infinity in the image plane.

2.2. Request 2: Rectification Mapping H_R and depth m

In order to compute the Euclidean Rectification Mapping H_R, we need to consider the vanishing line computed in point 2.1, along with the extracted matrix C of the conic. To solve such problem, we need to determine the circular points (I', J'). We know that the intersection of the image of the conic C' and the vanishing line consists of two points, which are the images of the circular points I' and J'.

To find I' and J', we solve the following equation:

$$l'_\infty \cdot C' = 0$$

This yields two conjugate solutions, corresponding to the circular points. Using these, the image of the conic dual to the circular points is given by:

$$C^*_\infty{}' = I'J'^T + J'I'^T$$

Next, we apply singular value decomposition (SVD) to the image of the conic dual to the circular points to obtain the matrix U, and the rectifying transformation matrix H_R (named T in the MATLAB section) can be obtained consequently:

$$\text{svd}(C^*_\infty{}') = U \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 0 \end{bmatrix} U^T$$

$$H_R = \begin{bmatrix} \frac{1}{\sqrt{a}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{b}} & 0 \\ 0 & 0 & 1 \end{bmatrix} U^T$$

Finally, to determine the depth m of the parallelepiped, we compute the proportion between Euclidean distance in pixels and real-model meters. This is possible because we know that any line $1 = 1$ meter.

To achieve this, we find the intersection points of a line l_i with two lines m_j, obtaining the vertexes $p1$, $p2$, $p3$, $p4$ of the base of the parallelepiped (if not already available by data).

The rectified points $p1'$, $p2'$, $p3'$, $p4'$ with respect to the horizontal plane are obtained by rectifying through H_R (T) by:

$$p1' = H_R \times p1^T$$

$$p2' = H_R \times p2^T$$

$$p3' = H_R \times p3^T$$

$$p4' = H_R \times p4^T$$

Considering that $p1'$, $p2'$ are the points delimiting line l_2 We can then compute the depth m by proportion with respect to real life metrics and observed euclidean distances in the rectified space. The meters-to-pixels scaling can be computed as:

$$\text{scaling}_{px/m} = \frac{\text{dist}(p1', p2')}{1\,\text{m}}$$

We can then evaluate directly depth m from points $p1'$, $p3'$, delimiting line m_5, as:

$$m = \frac{\text{dist}(p1', p3')}{\text{scaling}_{px/m}}$$

2.3. Request 3: Calibration matrix K

Since the camera is known to be zero skew, i.e. the axes of the camera sensor are perfectly orthogonal and the pixels on the image sensor are arranged in a perfect rectangular grid, there are only four unknowns in the calibration matrix K:

$$K = \begin{bmatrix} f_x & 0 & U_0 \\ 0 & f_y & V_0 \\ 0 & 0 & 1 \end{bmatrix}$$

The image of the absolute conic ω can be considered from the calibration matrix K by:

$$\omega = (KK^T)^{-1}$$

Therefore, calibration matrix K can be found considering a rectified face and an orthogonal vanishing point to it. For this, we consider the rectifying homography T (H_R) from the previous point, its inverse $T = T^{-1} = [t_1 \;\; t_2 \;\; t_3]$, and the orthogonal vanishing point from lines h_i, which can be evaluated as the cross product of two h_i as showed for lines l_i or m_j in section 2.1. We also consider the vanishing points v_l, v_m from previous sections.

We can then find ω solving the system of equations derived from vanishing point ortogonality and rectification conditions between ω and T^{-1}:

$$v_l^T \omega v_h = 0$$

$$v_m^T \omega v_h = 0$$

$$t_1^T \omega t_2 = 0$$

$$t_1^T \omega t_1 - t_2^T \omega t_2 = 0$$

By solving this system numerically for the parameters f_x f_y u_0 v_0, we determine ω and its parameters. The calibration matrix K is then derived by applying the Cholesky factorization of $\omega^{-1} = KK^T$.

2.4. Request 4: Height h

Starting from the calibration matrix K obtained from section 2.3, and the image of the absolute conic ω from the previous step, we can compute the circular points of the vertical plane considering the vanishing line of the vertical plane. We already have the vanishing points required to compute it as v_l, v_h, so we can compute the vanishing line as:

$$l'_{1\infty} = V \times v_1$$

We consider v_l and v_h to evaluate it used because the plane of interest contains the lines l_i and h_j. Circular points I'_1 and J'_1 can be computed as the intersection between ω and the vanishing line $l'_{1\infty}$. This corresponds to solve a system of equations. The equations for the first circular point I'_1 are:

$$I'_1 \times l'_{1\infty} = 0$$

$$I'^T_1 \omega I'_1 = 0$$

The same process is repeated for the second circular point J'_1.

Once both circular points are determined, as for previous sections we compute the image of the conic dual to the circular points I'_1 and J'_1:

$$C^\star_{1\infty} = I'_1 J'^T_1 + J'_1 I'^T_1$$

By applying singular value decomposition (SVD) to the image of the conic dual to the circular points, we then obtain the matrix U and a new rectification matrix H_{Rvert} for the vertical plane, in the same way from section 2:

$$\text{svd}(C^\star_{1\infty}) = U \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 0 \end{bmatrix} U^T$$

$$H_{Rvert} = \begin{bmatrix} \frac{1}{\sqrt{a}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{b}} & 0 \\ 0 & 0 & 1 \end{bmatrix} U^T$$

Finally, to determine the height h of the parallelepiped, we compute the ratio between pixels and meters in the same way we computed it to find depth m.

Since we already have two points for the line l_2, namely $p1$, $p2$, we only need one additional point ($p5$ that forms a vertical line h_i with either of the two points, to define the height h. We then compute the rectified points ($p1'$, ($p2'$ and ($p5'$ with respect to the vertical plane:

$$p1' = H_{Rvert} \times p1^T$$

$$p2' = H_{Rvert} \times p2^T$$

$$p5' = H_{Rvert} \times p5^T$$

The scaling factor considering line l_i, as before, is computed as:

$$\text{scaling}_{px/m} = \frac{\text{dist}(p1', p2')}{1\,\text{m}}$$

We can then evaluate directly height h from points $p1'$, $p5'$ as:

$$h = \frac{\text{dist}(p1', p5')}{\text{scaling}_{px/m}}$$

2.5. Request 5: Rectification of S

For each point s_i, where $i = 1, \ldots, 12$, we need to satisfy the following equation:

$$S \times s_i = 0$$

This ensures that each point s_i lies on the curve S. To find the corresponding X-Y coordinates of the horizontal curve with respect to the horizontal plane delimited by lines m_5, m_6, we apply the rectification matrix H_R (Matrix T):

$$S_i = H_R \times s_i$$

This way, we obtain the homogeneous X-Y coordinates for each selected point, after normalizing w.r.t. the third coordinate of each point S_i.

2.6. Request 6: Camera Localization

From theory, homography H can be expressed as the product between the calibration matrix K and the matrix composed of the rotation vectors r_1, r_2 and the translation vector o_π with respect to the reference plane π of the subject:

$$H = K \begin{bmatrix} r_1 & r_2 & o_\pi \end{bmatrix};$$

The vectors r_1 and r_2 are expected to be orthogonal by definition. As a result, the camera's pose can be evaluated from H and K given by:

$$\begin{bmatrix} r_1 & r_2 & o_\pi \end{bmatrix} = K^{-1}H.$$

Since the reference plane π is the parallelepiped, and the homography H between the plane π and its image is the inverse of the rectification matrix H_R computed earlier (since H_R maps from the image to the plane), we can compute the camera's pose as:

$$\begin{bmatrix} r_1 & r_2 & o_\pi \end{bmatrix} = K^{-1}H_R^{-1}$$

3 | MATLAB Implementation

MATLAB Implementation Requests

1. Consider the image `Look-outCat.png`. Using feature extraction techniques (including those already implemented in Matlab) plus possible manual intervention, extract the images of useful lines and both the image C, of the circumference and the image S of the other planar curve.
2. Write a Matlab program that implements the solutions to problems 1–6 and show the obtained results.
3. Plot the rectified curve S and show different views of the recovered 3D model of the rectangular parallelepiped.

3.1. MATLAB Request 1: Data Extraction

We first need to extract all marked information from the image, i.e., lines $l_1, l_2, l_3, m_1, m_2, m_3, m_4, m_5, m_6, h_1, h_2, h_3, h_4$, along with conic C and curve S.
This translates into obtaining the line vectors for each, along with conic C Matrix and curve S point displacement.

Manual techniques were implemented in order to extract these information as follows:

- lines l, m, h: the cross product between two points was computed and normalized with respect to the third variable to obtain the line vector.
- Conic C: Conic C was defined using six points by constructing matrix A from their coordinates and solving for the null space via Singular Value Decomposition (SVD). The conic parameters $[a, b, c, d, e, f]$ were extracted from the last column of V in the SVD, forming the symmetric 3×3 conic matrix C.
- Curve S: 12 points were selected and extracted from the scene, for computation in further points of the project.

The functions **getpts** and **ginput(n)** were exploited to extract points from the source image manually.

This resulted in the following extracted scene:

Figure 3.1: Extracted Scene

Accordingly, some of the line equations were verified by intersecting them with the image borders, in order to ensure correctness of the computed vector:

Figure 3.2: Line l_2

Figure 3.3: Line m_1

Figure 3.4: Line h_1

3.2. MATLAB Request 2.1:

Starting from the data extracted from the previous section, the vanishing points were computed as the intersection of pairs of parallel lines in the image. Specifically, the vanishing point $\mathbf{v}_{l_1 l_3}$ was obtained by taking the cross product of the homogeneous coordinates of lines $\mathbf{l}_1$ and $\mathbf{l}_3$, while the vanishing point $\mathbf{v}_{m_1 m_6}$ was computed as the cross product of lines $\mathbf{m}_1$ and $\mathbf{m}_6$. Both vanishing points were normalized to their homogeneous coordinates by dividing by their third component.

The vanishing line $\mathbf{l}_\infty$ was then calculated as the cross product of the two vanishing points $\mathbf{v}_{l_1 l_3}$ and $\mathbf{v}_{m_1 m_6}$, and normalized to its homogeneous form. This line represents the horizon in the image plane, corresponding to the line at infinity in the projective space.

The computed vanishing points and the vanishing line were displayed for verification. The vanishing point from l_1 and l_3 was v_{l1_l3}, the vanishing point from m_1 and m_6 was v_{m1_m6}, and the vanishing line was l'_∞. This implementation successfully computed the vanishing line l'_∞ of the horizontal plane.

Figure 3.5: Vanishing line l_∞

The **blue** vanishing point was obtained from the intersection of lines m_1 and m_6, while the **red** vanishing point was obtained from the intersection of lines l_1 and l_3.

The following results were obtained:
Vanishing point from l_1 and l_3:

$$\mathbf{v}_{l1\_l3} = \begin{bmatrix} 3380.0 \\ 6706.0 \\ 1 \end{bmatrix}$$

Vanishing point from m_1 and m_6:

$$\mathbf{v}_{m1\_m6} = \begin{bmatrix} 588.0953 \\ 876.4468 \\ 1 \end{bmatrix}$$

Vanishing line (l_∞):

$$\mathbf{l}_\infty = \begin{bmatrix} -0.0001 \\ -0.0011 \\ 1 \end{bmatrix}$$

3.3. MATLAB Request 2.2:

The Euclidean rectification matrix T was computed by utilizing the vanishing line and a conic C extracted from the image.

First, the extracted data was loaded, including vanishing lines, and vanishing points, from the previous sections of the code.

Starting from matrix C of the conic, the Circular Points II and JJ were computed solving the system:

$$a_1 \cdot x^2 + b_1 \cdot x \cdot y + c_1 \cdot y^2 + d_1 \cdot x + e_1 \cdot y + f_1 = 0$$

$$l_{\infty,1} \cdot x + l_{\infty,2} \cdot y + l_{\infty,3} = 0$$

By solving for **x** and **y** through MATLAB's function **solve()**, since the image of the circular points are the intersection of the conic C with the vanishing line, I obtained the two complex Circular points. For determining the image of the dual conic, matrix $C_\infty^{*\prime}$ was computed as:

$$C_\infty^{*\prime} = I'J'^T + J'I'^T$$

From here, after computing a rectifying homography matrix H using the identity matrix and the vanishing line, following professor's Luca Magri method (from *"Exercise Lecture E4D"*), a Rotation Matrix **U** was defined enforcing the ellipse C's axes equal. This exploited the **AtoG** function available as course material to convert the Conic's coefficient into geometrical parameters, in order to define matrix U's angle constraints. Through a Scaling matrix **S** and a consequent affinity matrix **A** computed as $K = U * S * U'$ the final transformation matrix **T** was obtained as composition of transformations as:

$$T = A * H;$$

The final values obtained are:

$$\mathbf{T} = \begin{bmatrix} 1.3965 & -0.6187 & 0 \\ -0.6187 & 1.9656 & 0 \\ -8.0165 \times 10^{-5} & -0.0011 & 1 \end{bmatrix}$$

The resulting Euclidean Rectification applied to the image yielded a scene (Figure **3.6**) giving a bottom view rectified perspective of the subject, preserving angles in order to compute the depth **m** requested. The ratio in pixel between the line m_5 and line l_1,

known to be of length 1 meter, allowed to compute the depth, obtaining:

$$\mathbf{m} = 0.35458m$$

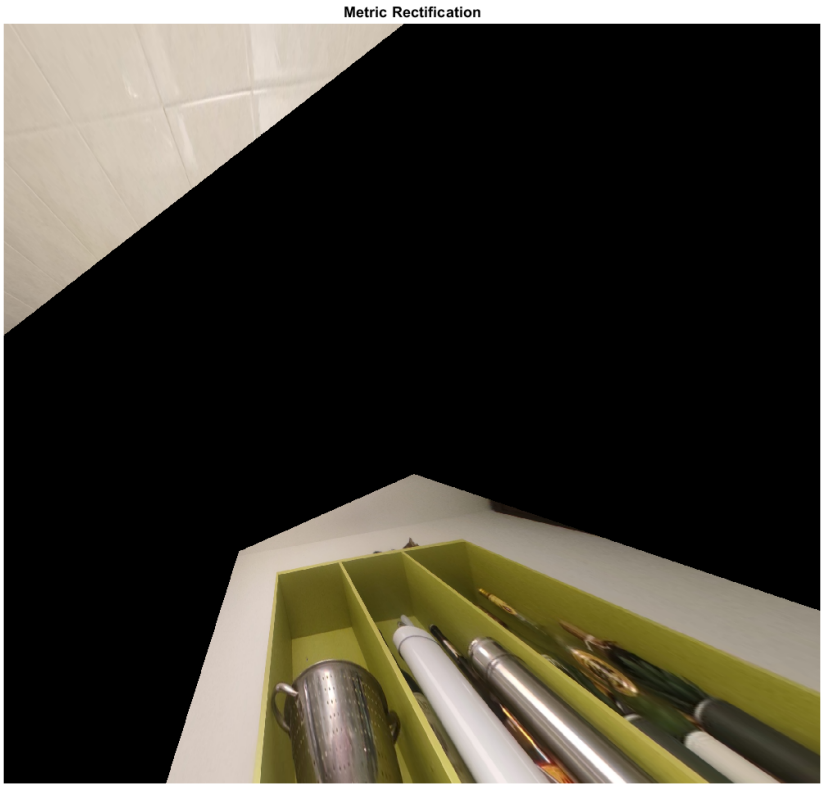

Figure 3.6: Euclidean Rectification from C

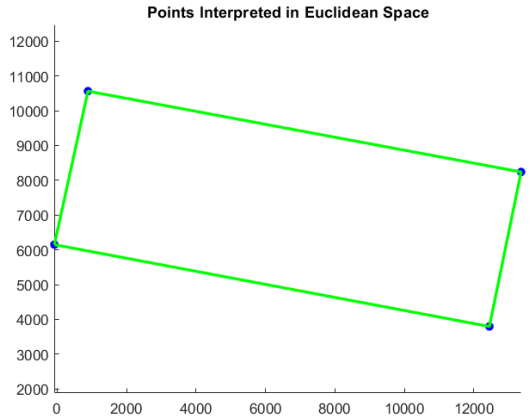

Figure 3.7: Base m of the subject

We can verify how the metric transformation preserved the angles for the base composed of line m_5 and line m_6 from the plot in Figure 3.7, checking the reconstructed base of the parallelepiped.

3.4. MATLAB Request 2.3:

In order to compute the calibration matrix $\mathbf{K}$ of the camera, we're required to compute the matrix from constraints on orthogonal vanishing points. For determining the orthogonal vanishing point, lines h_1, h_2 were considered to obtain the vanishing point v_{h1h2}, the only one not computed in previous sections.
First, the columns of the inverse of matrix T were considered as:

$$\mathbf{T^{-1}} = [t_1, t_2, t_3]$$

Using symbols f_x, f_y, u_0, v_0, matrices $\mathbf{K}$ and ω were defined symbolically. Following the theoretical solution, I employed four equations using previously calculated vanishing points and the rectification matrix's inverse from the prior problem. Indeed, through Zhang's Method for *camera calibrations from known planar shapes*, we can define the following system of equations:

$$\mathbf{v}_{l1l3}^{\top}\,\boldsymbol{\omega}\,\mathbf{v}_{h1h2} = 0$$

$$\mathbf{v}_{m1m6}^{\top}\,\boldsymbol{\omega}\,\mathbf{v}_{h1h2} = 0$$

$$\mathbf{t}_1^{\top}\,\boldsymbol{\omega}\,\mathbf{t}_2 = 0$$

$$\mathbf{t}_1^{\top}\,\boldsymbol{\omega}\,\mathbf{t}_1 - \mathbf{t}_2^{\top}\,\boldsymbol{\omega}\,\mathbf{t}_2 = 0$$

Through MATLAB's `vpasolve` function, in order to find a numerical solution to the four equations given the high number (4) of symbolic variables to solve, the calibration matrix $\mathbf{K}$ was evaluated resulting as:

$$\mathbf{K} = \begin{bmatrix} 762.4680 & 0 & 805.3771 \\ 0 & 778.8190 & 556.8548 \\ 0 & 0 & 1 \end{bmatrix}$$

3.5. MATLAB Request 2.4:

To determine the vanishing line of the vertical plane as $l1_{\inf_p rime}$, the vanishing points v_{h1_h2} and v_{l1_i3} were crossed.

To identify the circular points, as outlined in Problem 2.2, two equations were established using the symbolic variables x and y:

$$l1^T_{\inf_p rime} \times I_{\text{prime}} = 0$$

$$I^T_{\text{prime}} \omega I_{\text{prime}} = 0$$

These equations were solved through **solve()** for the symbolic variables **x, y**, in order to obtain once again the image of the circular points and the dual conic. The rectifying transformation $\mathbf{H}_{Rvert}$ was then derived using Singular Value Decomposition of the conic dual to the circular points. After applying this transformation to rectify the image points at the vertexes of the considered vertical rectangle, this enabled to calculate the height in the same way depth was computed in the previous paragraph obtaining:

$$\mathbf{h} = 0.38462m$$

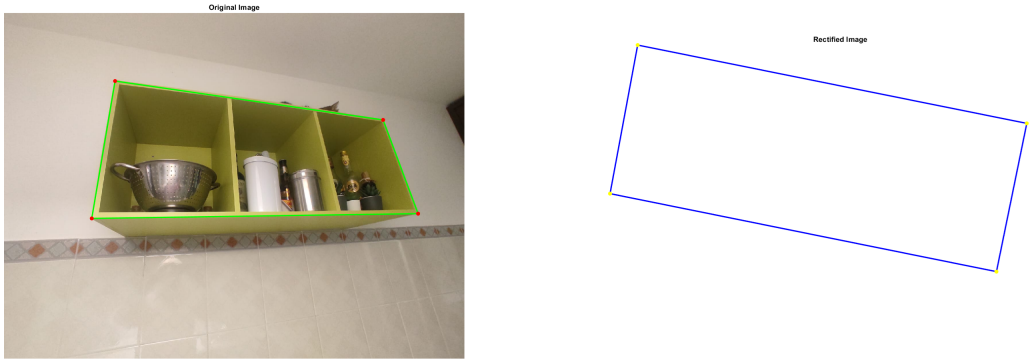

Figure 3.8: Rectified Vertical Plane

From the second subplot, we can observe how the rectified vertical plane of the parallelepiped preserves the angles as expected with little to no noise, allowing to compute the height correctly.

3.6. MATLAB Request 2.5:

The Rectification of the 12 points of **S** is computed directly from matrix **T** obtained in the previous sections. Transformation T mapping the lower side of the parallelepiped is applied to the points of **S** as:

$$S_i = T * s_i$$

This yields the X-Y coordinates:

$$\mathbf{S_{rect}} = \begin{bmatrix} 2941.3517 & 1381.8627 \\ 2851.1471 & 1310.1073 \\ 2769.5589 & 1274.9302 \\ 2682.4338 & 1249.9673 \\ 2589.8124 & 1237.6670 \\ 2521.8416 & 1246.5530 \\ 2441.3100 & 1281.7297 \\ 2389.3326 & 1333.0092 \\ 2362.0393 & 1380.9353 \\ 2349.8979 & 1455.6222 \\ 2361.9884 & 1516.2134 \\ 2398.9329 & 1570.0334 \end{bmatrix}$$

3.7. MATLAB Request 2.6:

To obtain both rotation vectors and translation vector with respect to π, we solve the pose matrix from theory with matrix **K** and **T**, yielding:

$$R = [r_1 \quad r_2 \quad \alpha_r] = \begin{bmatrix} 0.9326 & 0.0965 & -1.0562 \\ -0.3587 & 0.9453 & -0.7149 \\ -0.03901 & -0.3113 & 1 \end{bmatrix}$$

The resulting camera placement can be seen by plotting according to rotation vectors r_1, r_2 and the translation vector o_π with respect to the reference plane π of the subject.

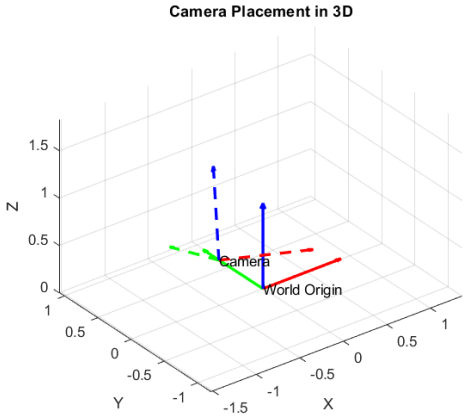

Figure 3.9: Camera Placement

If we consider the plotted camera placement, we can spot how the X-axis of the camera (**red dotted line**) points right with respect to the X-axis of the world, while the Z-axis of the camera (**blue dotted line**) is slightly tilted. This suggest that the camera is being pointed tilted rightwards and slightly upwards, which is in accordance with the presented source view.

3.8. MATLAB Request 3:

We can display curve **S** with the top-down view directly from the rectified points computed in MATLAB request 2.5:

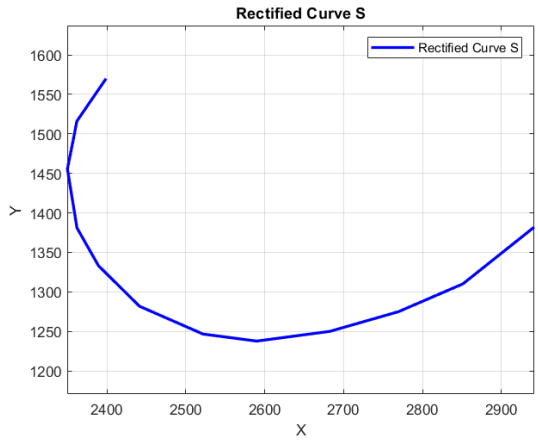

Figure 3.10: Curve S

As we can observe from the plot, the rectified image of the curve is compatible with the perspective we expect from the top down view of the considered object.

The 3D model of the parallelepiped can be built from the computed depths **m** and height **h** from the previous sections. Indeed, the 8 vertexes of the parallelepiped can be defined using m, l and h as variables, and the faces of the parallelepiped are consequently defined as sets of vertexes. Three views are plotted respectively in order to show the 3D rendering of the model: `view(30, 30)`, `view(-30, 30)`, `view(60, 60)`.

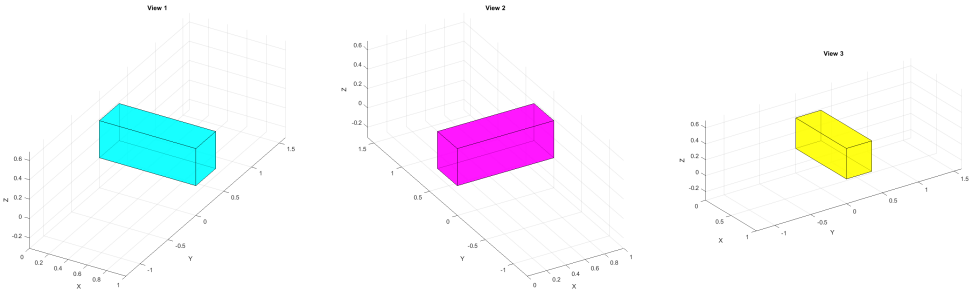

Figure 3.11: 3D View

4 | Conclusions

This Project allowed to apply Rectification and Reconstruction techniques to an existing, real scene of a parallelepiped object, in order to exploit Image Analysis methods to extract features, compute dimensions and transformation matrices, and perform the calibration of a zero-skew camera.
Both the original position of the camera with respect to the subject, and the 3D model of the parallelepiped itself, were reconstructed allowing to perceive the spacial disposition of the involved elements and their dimensions/orientations. Further improvements could be implemented, for example considering Section 1 of the MATLAB section. Methods for automatic detection of lines were considered and tested, involving **houghpeaks()**, **houghlines()** and **Harris Detector Method**.
All solutions, plots and data computed in the following document can be directly checked and evaluated through MATLAB via the provided live script file "**homework.mlx**". All results and useful data is provided in *".mat"* files along with the live script.

A | References

- Theory Lectures from "IACV" Course - Politecnico di Milano
- Exercise Lectures from "IACV" Course - Politecnico di Milano
- MATLAB Documentation (upd. 2025, version M2023B)