POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

IMAGE ANALYSIS AND COMPUTER VISION

HOMEWORK

Author:
Niccolò Balestrieri

Student ID:
218338

Advisor:
Prof. Vincenzo Caglioti

Academic Year:
2023-2024

Contents

1 | Assignment

1.0.1. Briefly introduction

Figure 1.1: A cylindrical voult in Palazzo Te - Mantova

An image of a right circular cylinder, captured by an uncalibrated zero-skew camera, which is approximately a natural camera but we CANNOT make this assumption, prompts a series of theoretical and practical tasks. The cylinder is characterized by circular cross sections (C_1, C_2) and generatrix lines (l_1, l_2). The goal is to extract information and perform various analyses using the provided image.

1.0.2. Task

Theory:

- **Find Horizon Line** h: Utilize C_1, C_2 to determine the vanishing line of the plane orthogonal to the cylinder axis.

- **Find Image Projection a and Vanishing Point V**: Use l_1, l_2, C_1, C_2 to find the image projection a of the cylinder axis and its vanishing point V.
- **Find Calibration Matrix K**: Employ l_1, l_2, C_1, C_2 (and potentially h, a and V) to determine the calibration matrix KK
- **Determine Orientation of Cylinder Axis**: h, a and V to establish the orientation of the cylinder axis relative to the camera reference.
- **Compute Ratio**: Calculate the ratio between the radius of circular cross sections and their distance.

Matlab:

- **Image Analysis**: Analyze the image "PalazzoTe.jpg" using feature extraction techniques, including possible manual intervention, to identify images l_1, l_2 of generatrix lines and images C_1, C_2 of circular cross sections.
- **Matlab Program**: Develop a Matlab program implementing solutions to theoretical problems 1–5.
- **Rectification of Cylindric Surface**: Plot the unfolding of the cylindrical surface, specifically the part between the two cross sections, onto a plane.

2 | Image Analysis

2.0.1. Image preprocessing

For a proper visual analysis, it was necessary to rotate the image by 90 degrees clockwise.

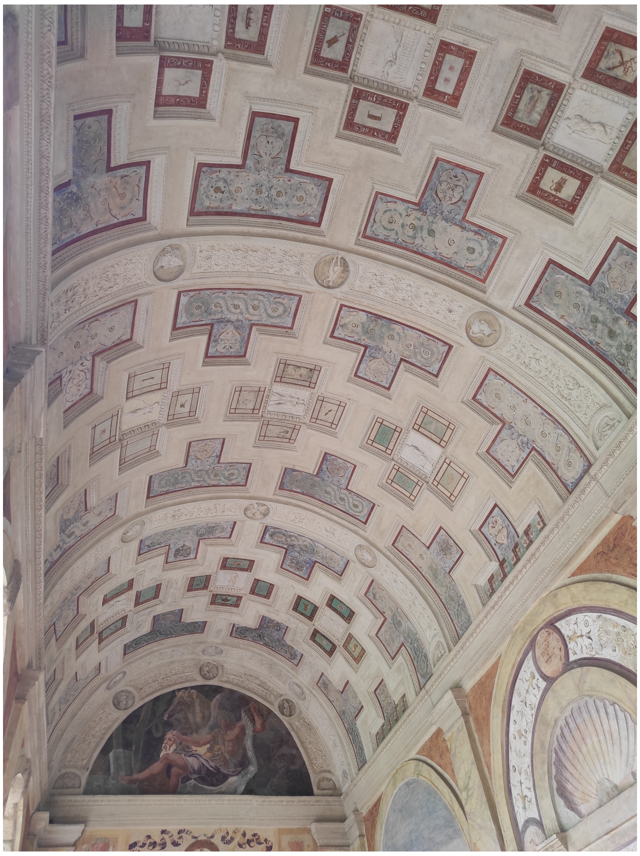

Figure 2.1: Rotated image

This was achieved using the `imrotate` function, which performed a rotation of the image to obtain the desired orientation. After that, to simplify subsequent processing, the image was converted to grayscale using the `rgb2gray` function. This step was essential to reduce chromatic information and focus on the luminance of the image.

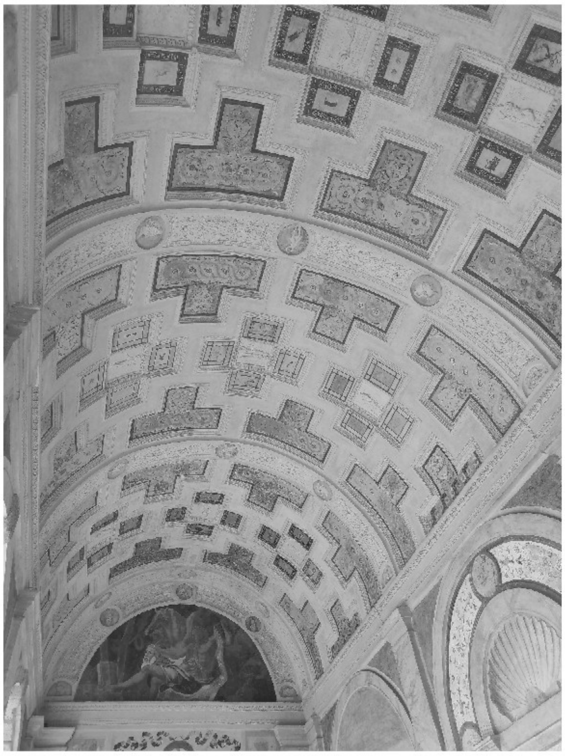

Figure 2.2: Grey Image

It is immediately noticeable from the grayscale image that additional preprocessing techniques need to be applied to enhance the edges more prominently. For this reason I experimented various image processing techniques and different filters such as `adapthisteq()`, and `histeq()`.

Figure 2.3: Histeq image

Figure 2.4: Adapt histeq image

After numerous trials, the most effective method proved to be the latter, which is an image processing technique and it is particularly useful for enhancing local contrast in an image. It adjusts the intensity distribution in different regions of the image, which can lead to improved visibility of details in both dark and bright areas. This technique, `adapt_histeq()`, has proven to be the most suitable for my purposes.

3 | Feature extraction

3.0.1. Edge detection

As a first step, I decided to test various edge detection algorithms to determine which one performed the best under default conditions. To ascertain the most suitable algorithm, I visually inspected the different images produced by each method and considered only the one that appeared most appropriate for my specific task.

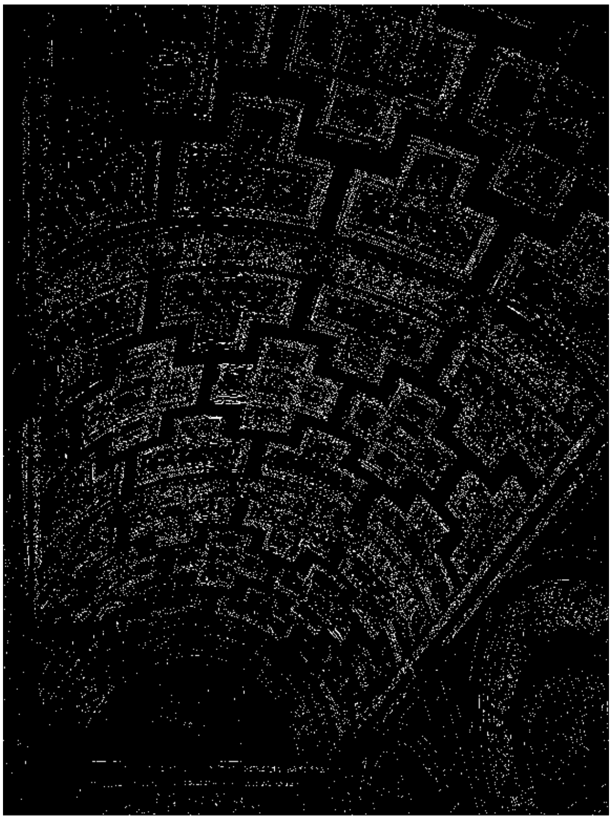

Figure 3.1: Sobel

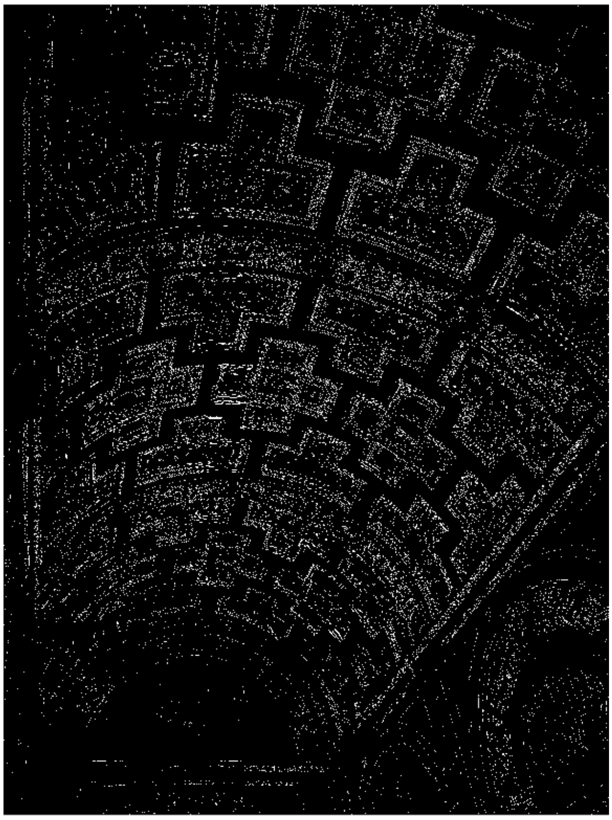

Figure 3.2: Prewitt

As can be observed from the first two images, the edge detection performed with **Sobel** and **Prewitt** was not deemed satisfactory, as it immediately discarded many edges along one of the two generatrix lines.

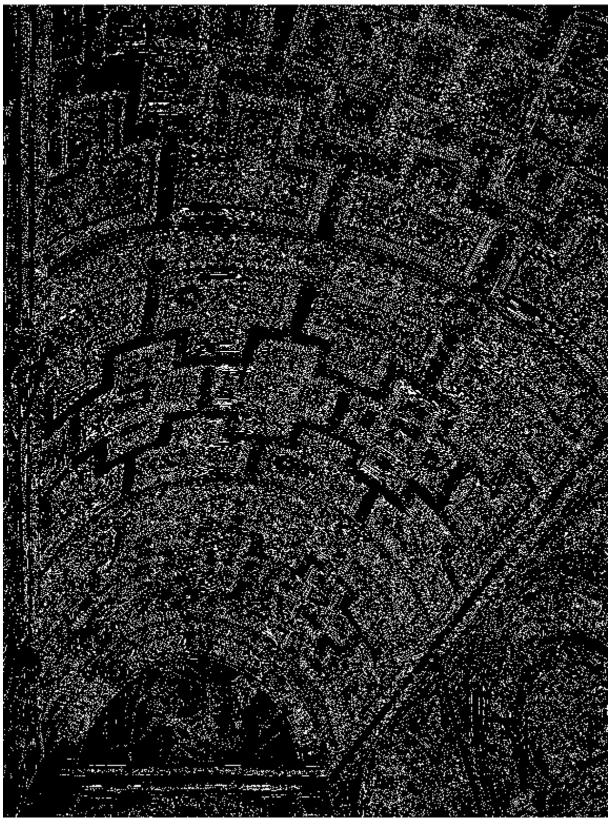

Figure 3.3: Canny

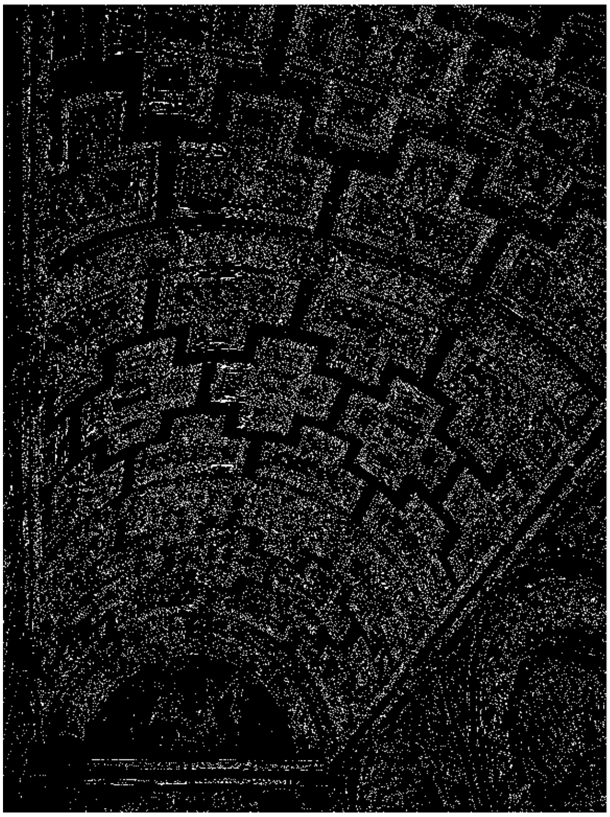

Figure 3.4: Log

In the last two images, in comparison to the previous ones, there is a clear change; both the **LoG** (Laplacian of Gaussian) and Canny edge detection methods capture many more edges. I chose to use **Canny** for my task because I considered it more beneficial to start with a noisier initial image and also because, different to the other differentiation methods, returns a binary image composed by lines, this result simplify the application of the lines' detection exploiting the **Hough transformation**.

Canny Algorithm

As mentioned above, I have decided to use the Canny algorithm. However, as evident from the image, it is still too noisy.

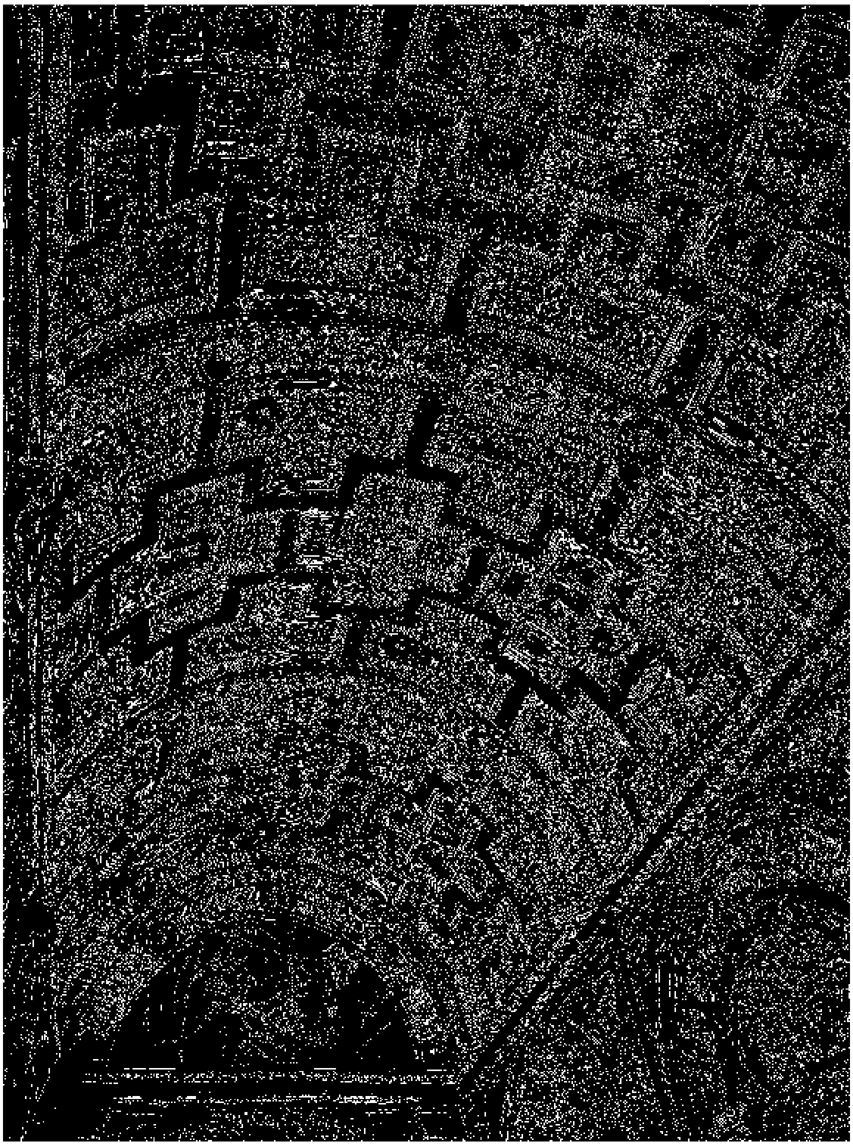

Figure 3.5: Canny algorithm

Therefore, I have chosen to adjust the threshold and σ filter parameters to reduce the noise and obtain an image that could meet my task: automatically detecting the two generatrix lines (l_1, l_2). As a first step, the ideal threshold values were selected through fine-tuning to mitigate the noise issue. However, as the problem persisted, for the second step, I opted to use the parameter σ. Through trial and error, I selected the value that I deemed most appropriate. These two steps are importants since too many edges cause the Hough Tranform to fail considering useless lines and as a result of them, I obtained the following edge detection:

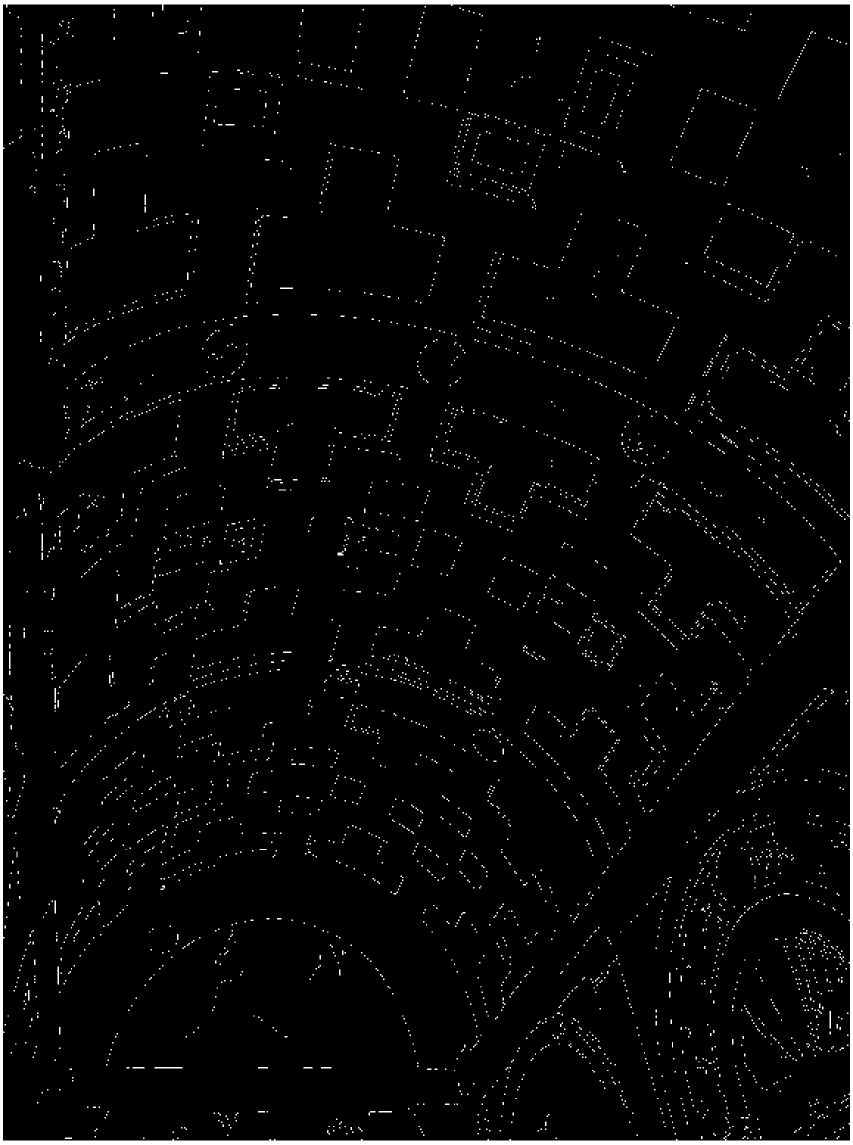

Figure 3.6: Canny with treshold and sigma

It is immediately noticeable that the noise, especially in the central part of the image, has been eliminated while preserving the edges. This is a crucial step, because the image was used in the Hough transform to automatically calculate the two generatrix lines (l_1, l_2).

3.0.2. Line detection - Hough Transform

The Hough Transform is an algorithm used for detecting lines in an image. In a nutshell, it converts image points into a parametric coordinate space, where a line can be represented by a unique pair of parameters. This facilitates line detection by counting intersections in the accumulator cells.

The formula for representing a line in Hough space is given by:

$$rho = x\cos(\theta) + y\sin(\theta)$$

where (rho, θ) are the parameters of the line and (x, y) are the image coordinates. In particular rho is the perpendicular distance from the origin to the line while θ, is the angle of the perpendicular projection from the origin to the line, measured in degrees clockwise from the positive x-axis and its range is $90 \leq \theta < 90$.

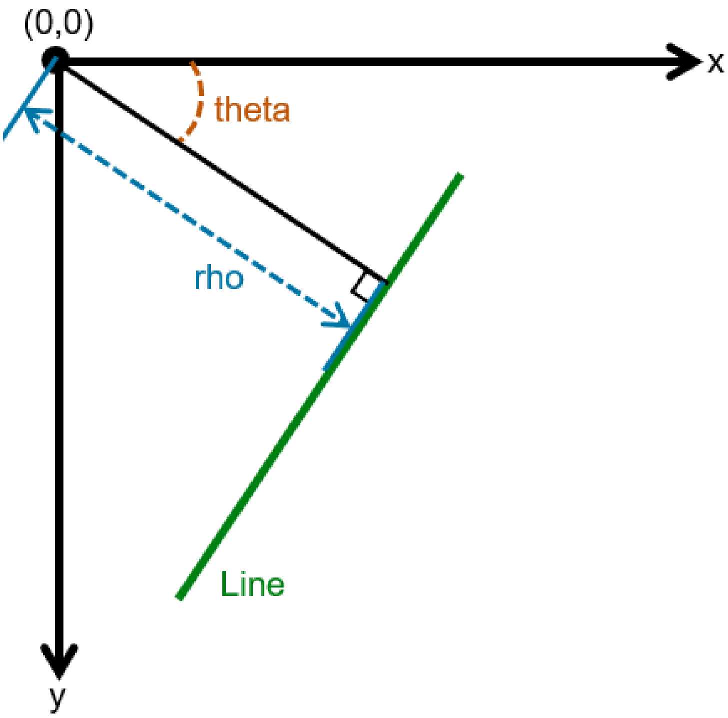

Figure 3.7: Hough explanation

For each pixel at (x, y) and its neighborhood, the Hough transform algorithm determines whether there is enough evidence of a straight line at that pixel; and if so, it will calculate the parameters (rho, θ) of that line. From this algorithm and a series of consideration on parameters, a little set of the best lines are in the following image:

Figure 3.8: Line detection with Hough transform

Utilizing the Hough transform, I automatically obtained the two generatrix lines $\mathbf{l}_1$ and $\mathbf{l}_2$ as follows:

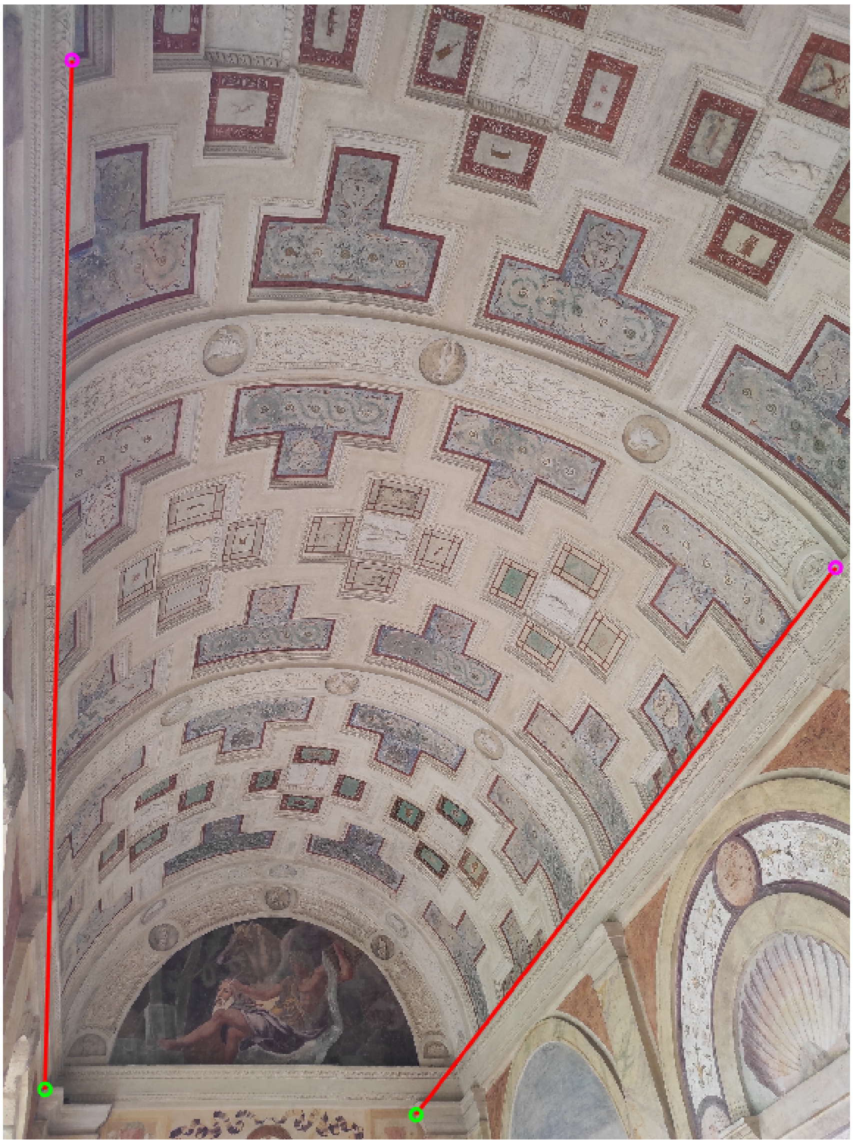

Figure 3.9: Generatrix lines with Hough transform

3.0.3. Ellipse detection

Regarding ellipse detection, after numerous attempts to automatically identify the two cross sections, C1 and C2, using 'regiornprops' without success, I have opted for a manual extraction method. This involves identifying them manually by drawing them.

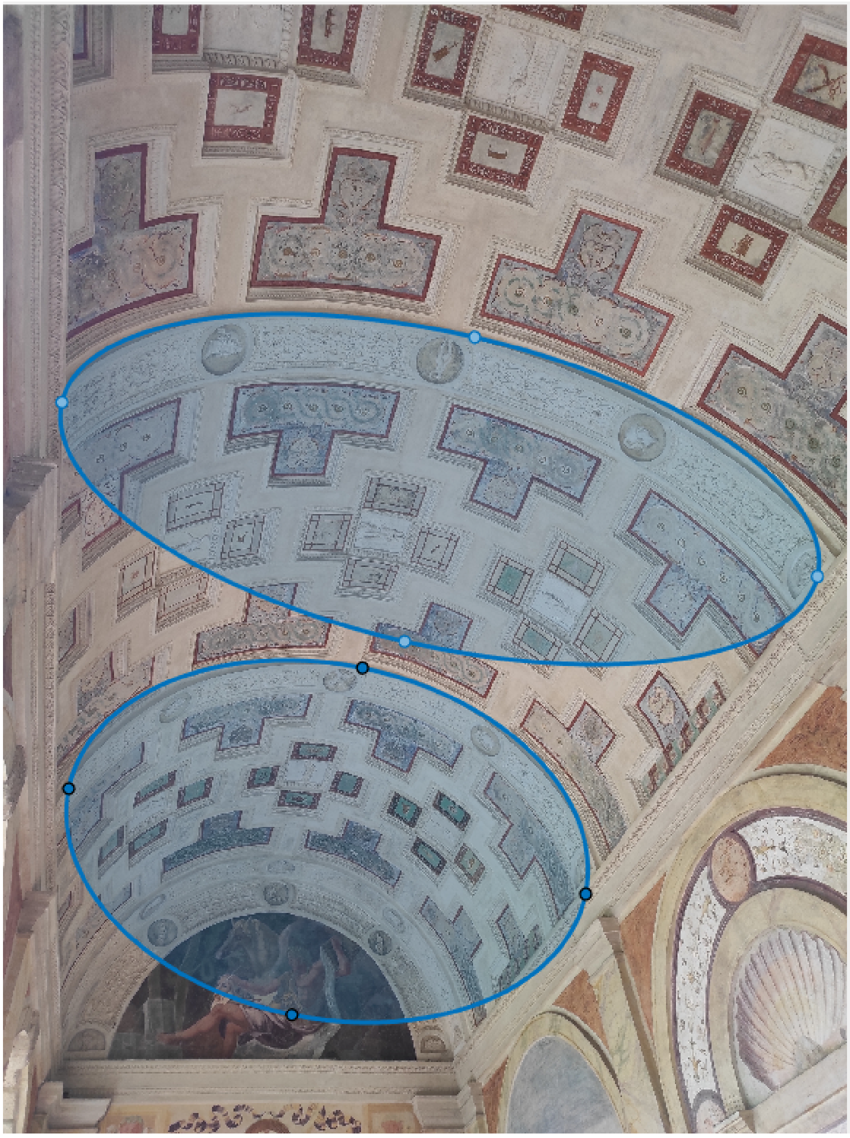

Figure 3.10: Detected cross section

3.0.4. Corner detection

Corner Detection is a computer vision approach used for feature detection(corners), corners are basically the intersection of two edges. Imagine you are putting pieces of a jigsaw puzzle together by looking at the picture. You tend to find those pieces first that have corners or edges in them and build the entire puzzle around them. That's because corners or edges are easy to locate due to the difference between the colour of the object and the background. This is the basic idea behind Corner Detection. If there is a change in the colour in both the x and y directions, it is assumed to be a corner. Corners are the considered to be more robust features in an image because they are invariant to translation, rotation, and illumination.

Harris Corner Detector

The one I used is harris, because after several trials with other techniques I felt that for my task it was the best technique. The whole image is divided into similar blocks called windows. Harris Corner Detector determines which windows produce a huge difference in intensities when moved in both x and y directions. For each window, a score R is calculated. We define a threshold to this score and strong corners are found.
The results I obtained are the following:

Figure 3.11: Harris corner detector

4 | Horizont Line

To determine the horizon (vanishing) line of the plane orthogonal to the cylinder axis, it was crucial to identify the image of circular points. To achieve this goal, calculating the intersection of two ellipses was imperative. These ellipses were represented in the form of two second-degree equations, characteristic of each conic:

$$\text{Ellipse 1: } a_1x^2 + b_1xy + c_1y^2 + d_1x + e_1y + f_1 = 0$$

$$\text{Ellipse 2: } a_2x^2 + b_2xy + c_2y^2 + d_2x + e_2y + f_2 = 0$$

where $a_1, b_1, c_1, d_1, e_1, f_1$ are the coefficients of the first ellipse, and $a_2, b_2, c_2, d_2, e_2, f_2$ are the coefficients of the second ellipse.
In Euclidean geometry, circles and ellipses are distinct, with ellipses described algebraically by second-degree equations. While two ellipses typically intersect in four points, the geometric insight is that distinct circles intersect at most two points. This uniqueness arises due to the existence of two additional complex solutions.
Notably, the result of the system of equations yields a pair of conjugate complex solutions s_1, s_2, s_3, s_4, crucial for determining two non-overlapping horizon lines.

$$\mathbf{s}_1 = 1.0e^{03} * \begin{bmatrix} 3.0198 + 4.7730i \\ -0.0109 + 1.5335i \\ 0.0010 + 0.0000i \end{bmatrix} \quad \mathbf{s}_2 = 1.0e^{03} * \begin{bmatrix} 3.0198 - 4.7730i \\ -0.0109 - 1.5335i \\ 0.0010 + 0.0000i \end{bmatrix}$$

$$\mathbf{s}_3 = 1.0e^{03} * \begin{bmatrix} 1.6099 + 0.3535i \\ 2.7125 + 0.0904i \\ 0.0010 + 0.0000i \end{bmatrix} \quad \mathbf{s}_4 = 1.0e^{03} * \begin{bmatrix} 1.6099 - 0.3535i \\ 2.7125 - 0.0904i \\ 0.0010 + 0.0000i \end{bmatrix}$$

This points are the image of the circular points and they aren't on the scene but on the image. To calculate $l_{\text{inf}_1}, l_{\text{inf}_2}$it was necessary to perform the cross product with s_1, s_2, s_3, s_4:

$$\mathbf{l}_{\text{inf}_1} = s_1 \times s_2 \quad \mathbf{l}_{\text{inf}_2} = s_3 \times s_4$$

Using all this information, it was possible to draw the horizon line (for simplicity, only one of the two was considered):

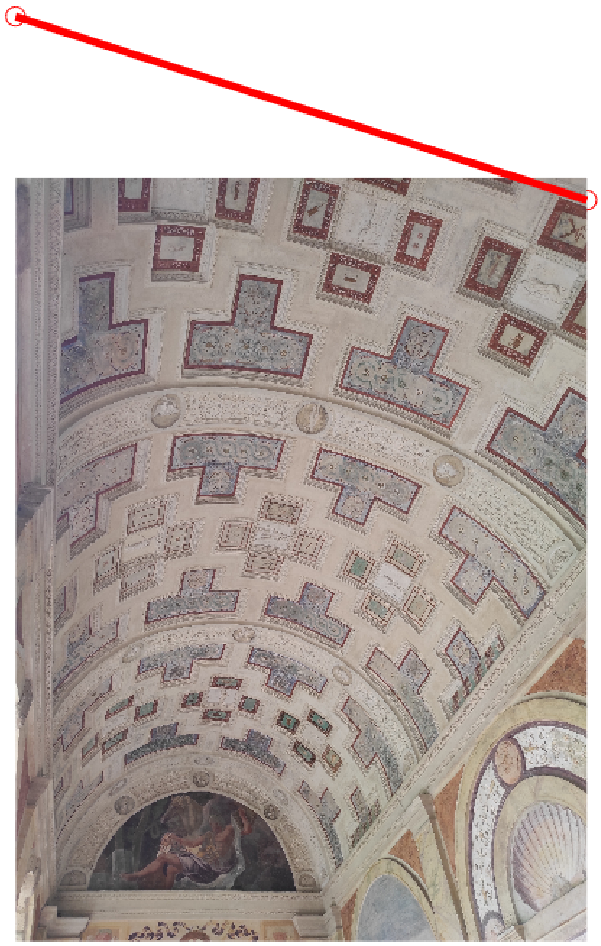

Figure 4.1: Horizon line h

5 | Vanishing point and image projection of the cylinder axis

The concept of a vanishing point refers to the visual perspective in which two parallel lines appear to converge towards a point at infinity along the direction of both lines. In other words, it is the projected intersection point at infinity of two parallel lines. Suppose we have two parallel lines, l_1 and l_2, which means:

$$l1 = \begin{bmatrix} a & b & c_1 \end{bmatrix}^T$$
$$l2 = \begin{bmatrix} a & b & c_2 \end{bmatrix}^T$$

The point $x = \begin{bmatrix} x & y & w \end{bmatrix}^T$ common two these lines satisfies both:

$$\begin{cases} ax + by + c_1 w = 0 \\ ax + by + c_2 w = 0 \end{cases} \longrightarrow x = \begin{bmatrix} b & -a & 0 \end{bmatrix}^T$$

In this case to compute the vanishing point V, the two parallel lines used are the generatrix lines l_1 and l_2 with equations $a_1 x + b_1 y + c1 = 0$ and $a_2 x + b_2 y + c_2 = 0$. The coordinates of the vanishing point P were obtained by finding the intersection of these lines. This involved solving the simultaneous equations formed by setting the two line equations equal to each other:

$$\begin{cases} -0.0035x - 0.0001y + 1.0000 = 0 \\ -0.0002x - 0.0001y + 1.0000 = 0 \end{cases}$$

The resulting solution provides the coordinates of the vanishing point V in the rotated image plane.

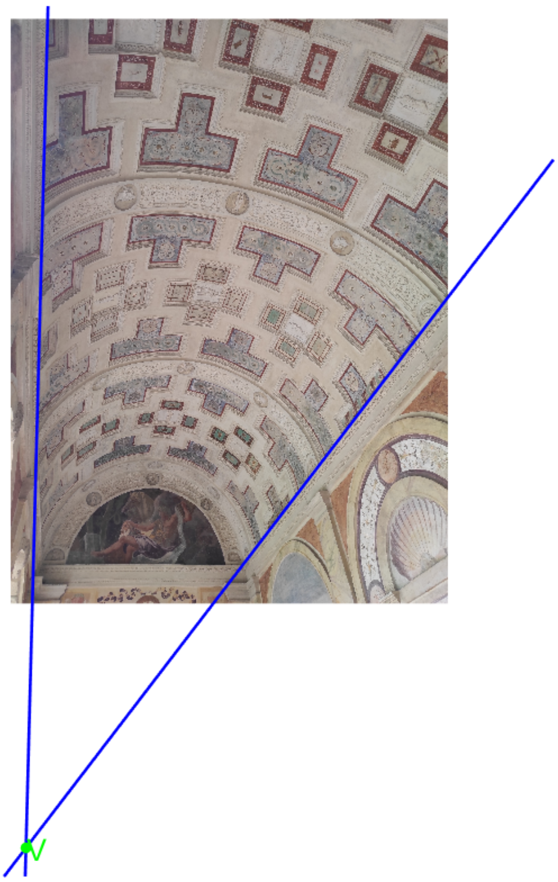

Figure 5.1: Vanishing point

In the process of obtaining the image projection a of the cylinder axis, the concept of the polar line has been effectively leveraged. Specifically, consider a point y and a conic C, where the line $l = Cy$ is defined.

$$\mathbf{l} = Cy = \begin{bmatrix} 1 & 0 & -X_0 \\ 0 & 1 & -Y_0 \\ -X_0 & -Y_0 & X_0^2 + Y_0^2 - r^2 \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -r^2 \end{bmatrix}$$

The polar line equation is $-r^2 w = 0$ i.e. $w = 0$ i.e. the line at the infinity l_∞ and l is denoted as the polar of y with respect to C, while y is identified as the pole of l with respect to C.

A notable scenario arises when y corresponds to the center of a circumference, and its polar line is the line at infinity of the plane containing the circumference. In this particular case, the polar line, denoted as h (horizon line previously computed), assumes a fundamental role. Commencing from this line and the two ellipses $ellipse1, ellipse2$, it becomes feasible to derive their respective centers $c1_{center}(A), c2_{center}(B)$.

$$\mathbf{c1}_{center} = 1.0e^{04} * \begin{bmatrix} -2.7910 \\ -2.7395 \\ -0.0015 \end{bmatrix} \quad \mathbf{c2}_{center} = 1.0e^{05} * \begin{bmatrix} -1.1572 \\ -2.7567 \\ -0.0008 \end{bmatrix}$$

Subsequently, with both centers computed, it becomes possible to ascertain the line passing through these two points, representing the image projection a of the cylinder axis.

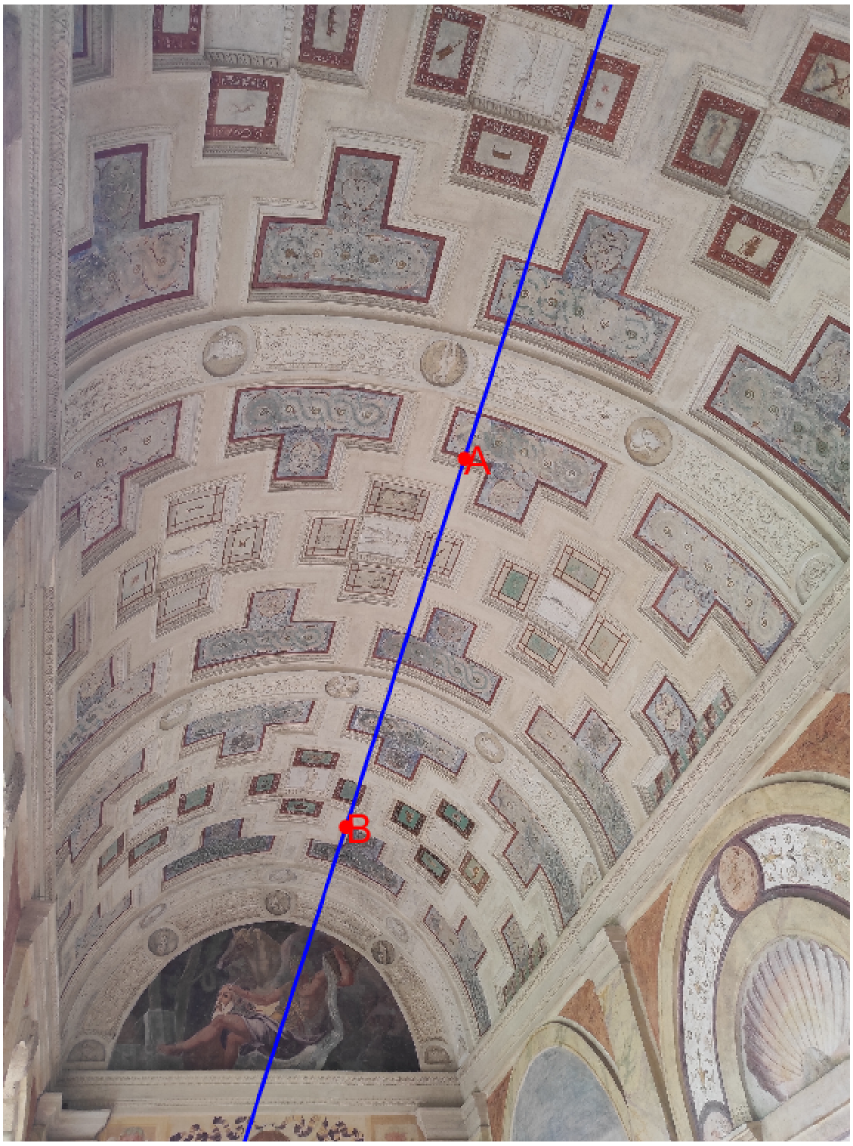

Figure 5.2: Axis a

This methodology demonstrates a systematic approach that capitalizes on the polar line concept to precisely infer the image projection of the cylinder axis based on the geometric properties of the ellipses and their relationship to the line at infinity.

6 | Calibration Matrix K

In camera calibration, the quest is to determine the intrinsic parameter matrix K of the camera define as:

$$K = \begin{bmatrix} fx & a & u_0 \\ 0 & fy & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Where f_x, f_y are the focal distance, U_o, V_o are the principal point and a skew factor aspect ratio a which in this case is equal to 0. In this domain exist a projection matrix P that maps world points (X) into 2D image points (x) thanks to:

$$x = PX$$

As we said P is the matrix and it can be define as:

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

Where R corresponds to rotation between the camera and the world. Meanwhile, o is the location of the camera in the world reference frame in cartesian coordinates. It is crucial to emphasize that the matrix K can be conveniently determined by leveraging the image of the absolute conic.

We notice that w (the image of the absolute conic) depends only on the camera calibration matrix K:

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

In order to determine K we need to specify some constraints on w (the image of the absolute conic). For a zero skew camera the image of the absolute conic is given by:

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

So, if we manage to find w, we have also found K. The image of the absolute conic is a symmetric matrix and has 4 degrees of freedom, so we need to get only 4 equations are needed to determine it. To achieve this purpose, starting from the calibration from rectified face plus orthogonal vanishing point, the necessary equations for obtaining the IAC w were derived:

- from $h_1^T wv = 0$ and $h_2^T wv = 0$, and $l'_\infty = h_1 \text{x} h_2 \longrightarrow l'_\infty = wv$ (2 equations)
- from $I' = H_R^{-1} I = H_R^{-1} \begin{bmatrix} 1 \\ i \\ 0 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} \begin{bmatrix} 1 \\ i \\ 0 \end{bmatrix} = h_1 + ih_2$, and $(h_1 + ih_2)^T w (h_1 + ih_2) = 0 \longrightarrow \begin{cases} h_1^T w h_2 = 0 \\ h_1^T w h_1 - h_2^T w h_2 = 0 \end{cases}$

Which are linear equations in w.
Breaking down the first equation and extracting the components of the rectifying homography matrix $h1, h2$, che è stata calcolata a partire dalla image of the dual conic:

$$C^{*}_{\infty}{}' = II * JJ.' + JJ * II.'$$

On it, the Singular Value Decomposition (SVD), svd($C^*_\infty{}'$), is performed.
Developing this process yields H_{rect} of the plane of ellipses orthogonal to the axis:

$$H_{rect} = \begin{bmatrix} -0.1357 & -0.0316 & -0.0000 \\ 0.2271 & -0.9739 & 0.0011 \\ -0.0000 & 0.0001 & 0.1394 \end{bmatrix}$$

Then the obtained system of equations is the following:

$$\begin{cases} h1' * w * h2 = 0 \\ h1' * w * h1 - h2' * w * h2 = 0 \\ V(1) * w1 + V(3) * w3 = h(1) \\ V(1) * w2 + V(2) * w3 + V(3) * w4 = h(3) \end{cases}$$

Where V, is the vanishing point previously calculated. By solving this system, it has become possible to find w and its parameters:

$$w = 1.0e^{06} * \begin{bmatrix} 0.0000 & 0 & -0.0000 \\ 0 & 0.0000 & -0.0005 \\ -0.0000 & -0.0005 & 3.1135 \end{bmatrix}$$

And as mentioned earlier, using the **Cholesky factorization**, the IAC has made it possible to find the calibration matrix K.

7 | Localization

In this point we have to find the relative position of the camera with respect to the reference points placed on the plane of the axis of the cylinder.
If we identify the plane of the cylinder axis with π (know as planar object) we can write the position of a point (in homogenous coordinates) in the object reference frame as:

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

while this point, in the world reference, can be written as:

$$X_w = \begin{bmatrix} i_\pi & j_\pi & k_\pi & o_\pi \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \\ w \end{bmatrix} = \begin{bmatrix} i_\pi & j_\pi & k_\pi & o_\pi \\ 0 & 0 & 0 & 1 \end{bmatrix} X_\pi$$

Where $X_\pi$is the position of the point in the plane reference frame and that plane has the x axis identical to the line at the bottom of the face, going right, the y axis with the same direction of the left line and the z axis orthogonal to both axis using the right hand rule. By putting the world reference frame on the camera, what we obtain is:

$$u = K \begin{bmatrix} i_\pi & j_\pi & o_\pi \end{bmatrix} x_\pi$$

Where x are the coordinates of the point on the plane. This allows transitioning to:

$$H = K \begin{bmatrix} i_\pi & j_\pi & o_\pi \end{bmatrix}$$

Where H can be considered as the matrix which map the world point into image point. In this context, it is important to note that H is equal to H_{rect}, calculated previously, and K corresponds to the calibration matrix computed earlier. Thus, the procedure involved solving the following equation:

$$localization = H_{rect} * K^{-1};$$

Starting from this matrix, I extracted i_π, j_π, o_π, and used them to create the 3D position of the camera (rotation matrix):

$$\begin{bmatrix} i_\pi & j_\pi & i_\pi \times j_\pi & 0_\pi \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, I utilized this matrix to compute the 3D coordinates of the two ellipse centers in camera reference:

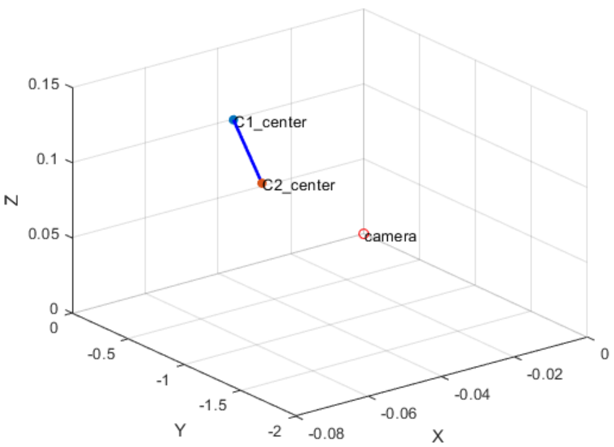

Figure 7.1: camera Reference

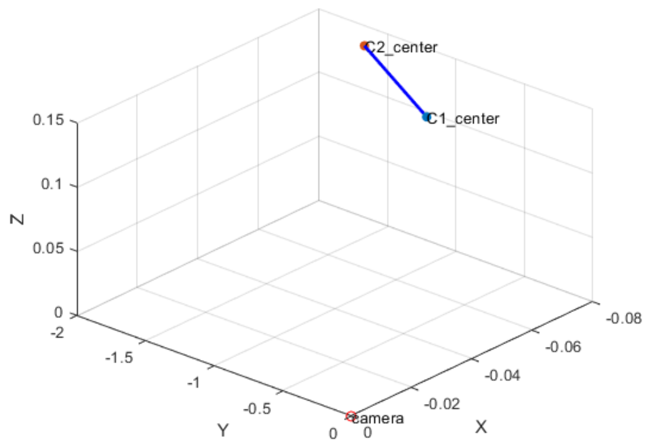

Figure 7.2: Camera Reference

8 | Compute the ratio between the radius of the circular cross sections and their distance

To calculate the ratio between the radii of the two circular cross-sections, it was first necessary to apply metric rectification to the initial image.
This rectification method is capable of preserving 90-degree angles and ratios of lengths between non-parallel lines. It is important to specify that rectification was performed with respect to the two generative lines and the corresponding axis of the conic, in order to envision an image where radii and distances from the center are preserved:

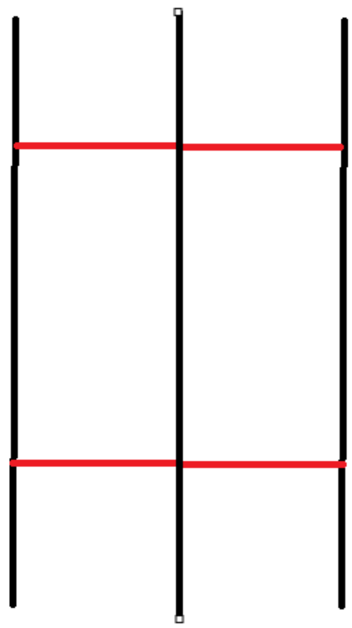

Figure 8.1: Rettificazione ideale

To implement it, it is necessary to find the vanishing line of the plane defined by the two generative lines and the axis of the cylinder. To calculate it, the vanishing point V (previously computed) and another vanishing point P were utilized. Once these two points are obtained, computing the line passing through them allows identification of the new vanishing line. To find P, the procedure involved taking the centers of the two diameters of the conics and the intersections between the ellipses and the generative lines.

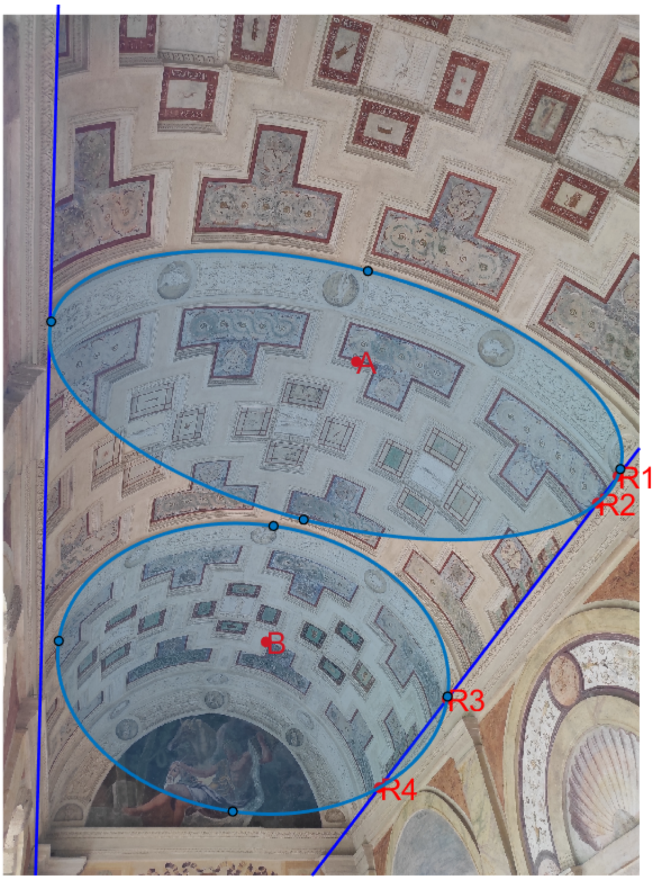

Figure 8.2: Intersection between Ellipses and rigth generatrix line

The coordinates of this points, are:

$$\mathbf{R1} = 1.0e^{03} * \begin{bmatrix} 3.2806 \\ 2.4612 \\ 0.0010 \end{bmatrix} \qquad \mathbf{R2} = 1.0e^{03} * \begin{bmatrix} 3.1712 \\ 2.6039 \\ 0.0010 \end{bmatrix}$$

$$\mathbf{R3} = 1.0e^{03} * \begin{bmatrix} 2.3700 \\ 3.6487 \\ 0.0010 \end{bmatrix} \qquad \mathbf{R4} = 1.0e^{03} * \begin{bmatrix} 2.0082 \\ 4.1204 \\ 0.0010 \end{bmatrix}$$

In my case, the utilized generative line is the right one because, as evident from the illustration, it is the only line intersecting both drawn ellipses. This is due to the challenge of tracing them more accurately (critical as it results in a small margin of error in the final outcomes).
It is crucial that P is determined based on two directions orthogonal to that of the first vanishing point.

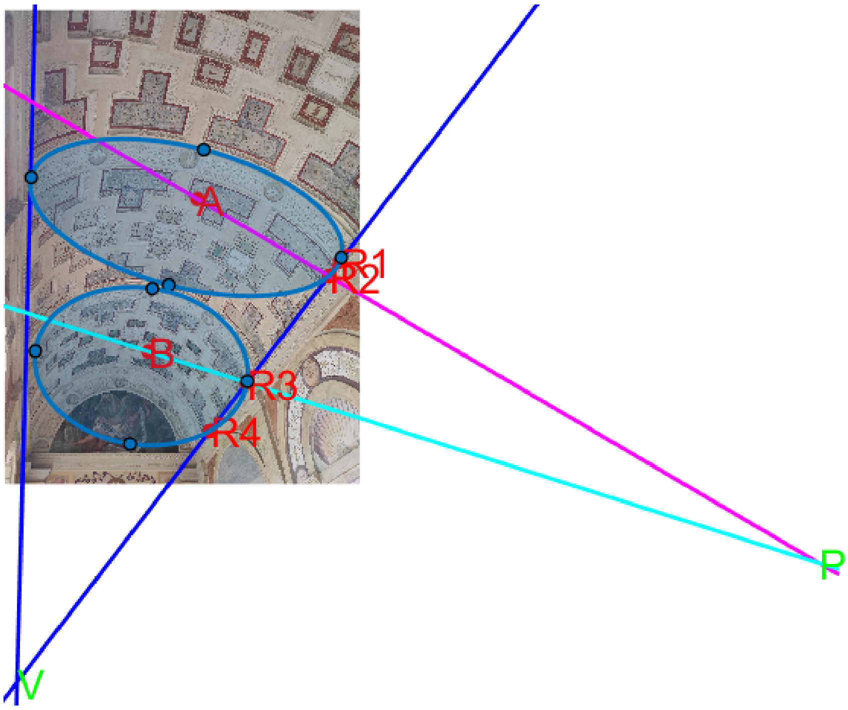

Figure 8.3: Vanishing points V and P

In this way, it will be possible to calculate the line passing through the two vanishing points, which represents the line at infinity.

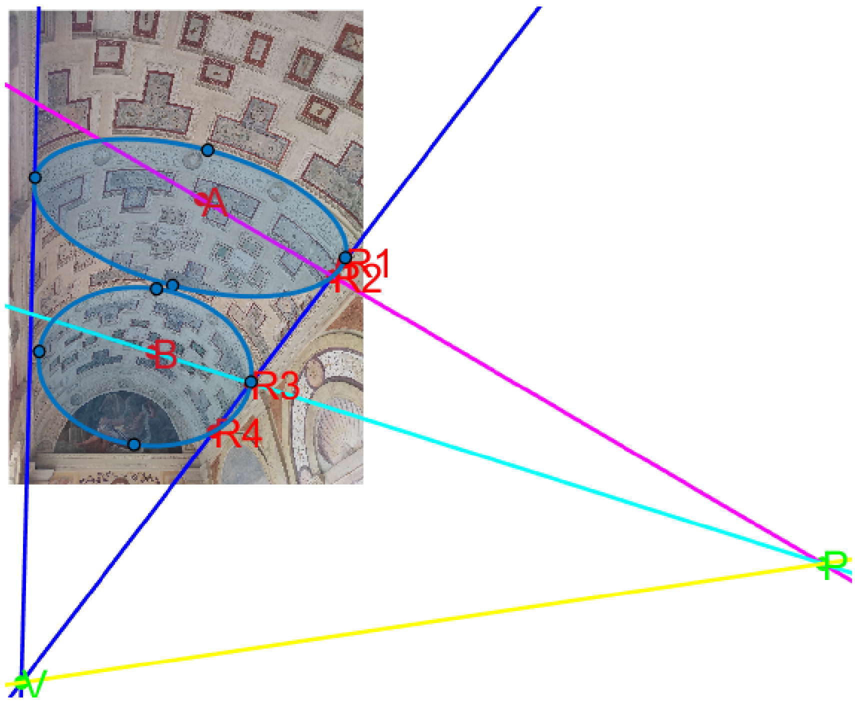

Figure 8.4: Horizon line of the axis plane

The line at infinity (depicted in yellow), of the plane parallel to the cylinder axis, can be intersected with the image of the absolute conic, denoted as IAC, which is unique for the entire image.

Once intersected, what is obtained are the images of circular points; using these, the imDCCP (image of the dual conic) is calculated as follows:

$$C_{\infty}^{*}{}' = II * JJ.' + JJ * II.'$$

On it, the Singular Value Decomposition (SVD), svd($C_{\infty}^{*}{}'$), is performed.

Developing this process yields H_{rect}, which differs from H_{rect} of the plane of ellipses orthogonal to the axis because it belongs to the plane parallel to the axis.

$$H_{rect} = \begin{bmatrix} -0.2433 & -0.0003 & -0.0000 \\ 0.0012 & -1.0000 & -0.0002 \\ -0.0000 & -0.0000 & 0.2433 \end{bmatrix}$$

It is then applied to the centers of these two circles and the intersection points used to calculate the diameters. Subsequently, using Euclidean distance, which is the norm of the difference between the two vectors of points found, the radii are determined. These radii are then used to calculate their respective ratios, which is calculate as the ratio between the radius and the distance between the two ellipse.

For the first ellipse the ratio is:

$$ratio = 0.5041$$

For the second ellipse the ratio is:

$$ratio = 0.4760$$

Due to the imprecision in obtaining the cross sections, noise has influenced the results.

9 | Unfolding of the part of the surface, included between the two cross sections, onto a plane

To fulfill this requirement, my approach was to take the surface enclosed by the two ellipses, rectify it, and then project it onto an unrolled plane. To achieve this, I experimented with various methods, and the one that yielded the best results was as follows. Given the difficulty of cropping the image directly in MATLAB, I utilized Paint functionalities to extract the desired image region. Subsequently, I applied rectification to this image and obtained the following result:

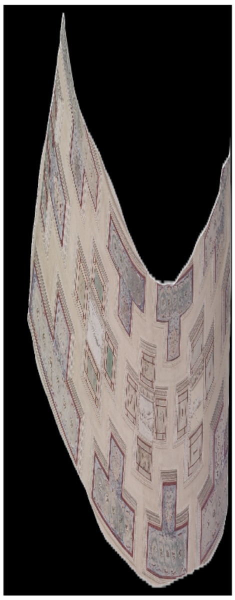

Figure 9.1: Rectified surface

Afterward, unfortunately, I encountered several challenges in implementing the unfolding of the image onto a rectangular plane. What I tried and thought of doing was to convert the Cartesian coordinates of the rectified image into cylindrical coordinates; after which, use the new coordinates to plot the image in the new plane.

10 | Computed Results

10.0.1. Horizon line

The computed and normalized horizon line h is equal to:

$$h = \begin{bmatrix} -0.0003 \\ 0.0010 \\ 1.0000 \end{bmatrix}$$

10.0.2. Useful vanishing points

Two vanishing points have been calculated, the vanishing point V:

$$V = 1.0e^{+03} * \begin{bmatrix} 0.1217 \\ 6.5803 \\ 0.0010 \end{bmatrix}$$

The second one is P:

$$P = 1.0e^{+03} * \begin{bmatrix} 7.9689 \\ 5.4156 \\ 0.0010 \end{bmatrix}$$

10.0.3. Calibration parameters

The camera calibration matrix is:

$$K = 1.0e + 03 * \begin{bmatrix} 7.3385 & 0 & 0.1217 \\ 0 & 1.6997 & 0.4730 \\ 0 & 0 & 0.0010 \end{bmatrix}$$

List of Figures