Image Analysis & Computer Vision
A.Y. 2023/2024

Carlo Fabrizio
M.Sc. Computer Science and Engineering, Politecnico di Milano - Milan, Italy
Email:
carlo.fabrizio@mail.polimi.it
Student ID: 10747982

1. Image Analysis and Feature Extraction

Consider the image PalazzoTe.jpg. Using feature extraction techniques (including those implemented in Matlab) plus possible manual intervention, extract both the images I1, I2 of useful generatrix lines and images C1, C2 of useful circular cross sections.

1.1. Image pre-processing

In the initial stage of image analysis, several crucial steps to preprocess the provided image have been applied. This preparatory action is fundamental to enhance the quality and highlight significant features for subsequent analyses.

To address potential contrast issues within the grayscale image, a histogram equalization technique is applied. This method redistributes pixel intensities, resulting in a more balanced representation of the image's intensity distribution. The goal is to improve visibility and accentuate features that may be crucial for subsequent analyses. The results of the equalization are provided in figure 1 and 2.

Figure 1: Original Image on the left, equalized image on the right.

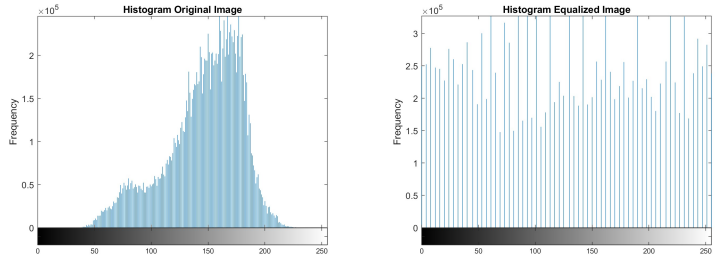

Figure 2: Histogram of the original Image on the left, histogram of the equalized image on the right.

Finally, a denoise filter using median operator is applied to the image in order to reduce the so called ”salt and pepper noise”.

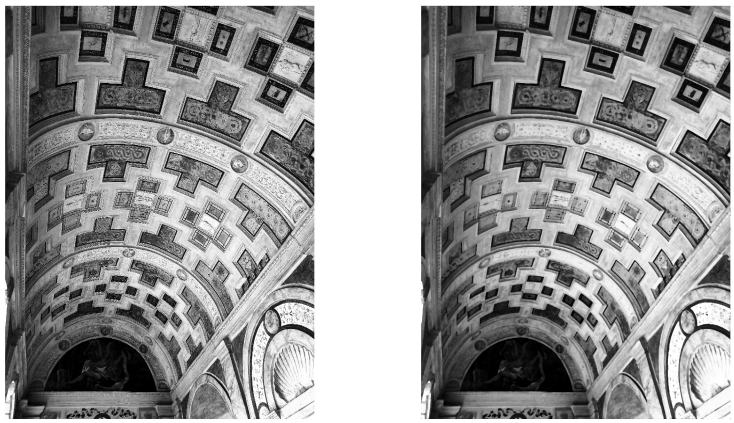

Figure 3: Non-denoised image on the left, denoised image on the right.

Hereafter, all operations concerning feature extraction will be base on the denoised - equalized - gray scale image.

1.2. Edge detection

All the Edge detector algorithms that have been used are base on derivatives estimation. A Gray scale image can be considered as a function $G : \mathbb{R}^2 \to \mathbb{R}$. In general, image derivatives are estimated by discrete approximation of continuous derivative operations:

$$\frac{\partial f(x_n, y_m)}{\partial x} \simeq \frac{f(x_{n+1}, y_m) - f(x_n, y_m)}{\Delta x} \tag{1.1}$$

$$\frac{\partial f(x_n, y_m)}{\partial y} \simeq \frac{f(x_n, y_{m+1}) - f(x_n, y_m)}{\Delta y}. \tag{1.2}$$

The latter operations can be easily implemented by means of convolutional filters: $\begin{bmatrix} 1 & -1 \end{bmatrix}$ for the X-axis, and $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$ for the Y-axis.

It is common practice to combine derivation with a smoothing operation, an example are the following derivatives operators:

- **Perwitt** derivator:

$$\textit{horizontal} \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \tag{1.3}$$

$$\textit{vertical} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{1.4}$$

- **Sobel** derivator:

$$\textit{horizontal} \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{1.5}$$

$$\textit{vertical} \begin{bmatrix} 1 & 2 & 1 \\ 1 & 2 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{1.6}$$

.

The general workflow of an edge detection algorithm consists in estimating derivatives along the axes, combining them to compute the magnitude and, finally, in cancelling out all the pixels whose magnitude is below a fixed threshold. A decent result has been obtained by applying the latter outline with Sobel derivatives estimation. The obtained binary mask for pixels is showed in figure 4.

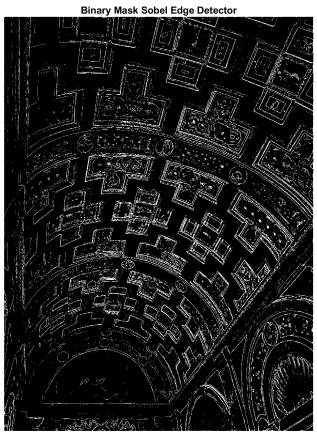

Figure 4: Sobel Edge detector binary mask.

The Canny edge detection algorithm has proven to yield the most favorable results. This algorithm employs a multi-step process to accurately identify edges within an image. The algorithm begins by estimating derivatives through a combination of Gaussian smoothing and classical derivative convolutional filters. This step helps highlight regions in the image with significant changes in intensity, indicative of potential edges. To enhance the reconstruction of lines and segments, Canny incorporates a mechanism known as Non-Maximum Suppression. This step involves thinning out detected edges by retaining only the local maxima along the gradient direction. By doing so, the algorithm refines the representation of edges, preserving only the most prominent features. Instead of employing a traditional thresholding operation, Canny utilizes a technique called Hysteresis Thresholding. This approach helps distinguishing pixels that truly belong to the same edge. Unlike a fixed threshold, hysteresis thresholding involves defining two thresholds: a higher threshold (strong edge) and a lower threshold (weak edge). Pixels with gradient magnitudes above the higher threshold are considered strong edges, while those between

the two thresholds are considered weak edges. The algorithm keeps all the strong edges and all the weak edges that can be directly connected to a strong edge. The result obtained is shown in figure 5 .

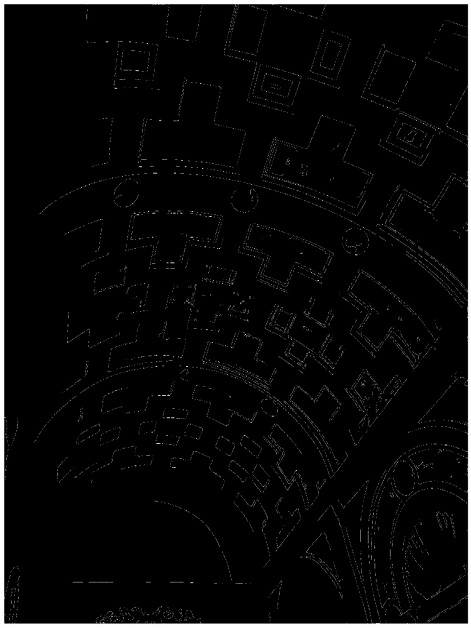

Figure 5: Canny edge detection result

The final step involves the application of the Hough transform for line detection. This transform is particularly useful for identifying straight lines within the image. The detected lines are visualized in figure 6.

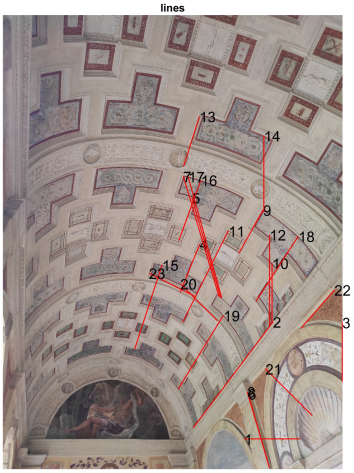

Figure 6: Canny edge detection result- lines

From the given lines, it has been possible to retrieve generatrix line $l2$ (line 2 in the image).

1.3. Corner Detection

The general idea of corner detection algorithms is that a corner is a point in the image that is meaningful along all the directions. The corner measure introduced by Moravec is the SSD (Sum of Square Difference) over a fixed set of displacements

$$E_{x,y}(r,c) = \sum_{(u,v)\in U_{r,c}} w_{r,c}(u,v) \cdot [I(u,v) - I(u-x, v-y)]^2 \tag{1.7}$$

where $U_{r,c}$ is a window centred in (r,c) that defines the neighborhood of (r,c), and $w_{r,c}(u,v)$ is the weight of the neighbor (u,v) of (r,c). An exponential decay for the weights of a neighborhood, like a Gaussian kernel, is often used. An approximation of eq. 1.7. is provided by the first-order terms in the Taylor expansion

$$E_{x,y}(r,c) \simeq \sum_{(u,v)\in U_{r,c}} w_{r,c}(u,v) \cdot [xI_x(u,v) + yI_y(u,v)]^2 \tag{1.8}$$

that can be rewritten as:

$$E_{x,y}(r,c) \simeq \begin{bmatrix} x & y \end{bmatrix} \cdot M_{r,c} \begin{bmatrix} x \\ y \end{bmatrix} \tag{1.9}$$

where

$$M_{r,c} = \begin{bmatrix} (I_x^2 \otimes w)(r,c) & (I_x I_y \otimes w)(r,c) \\ (I_x I_y \otimes w)(r,c) & (I_y^2 \otimes w)(r,c) \end{bmatrix}. \tag{1.10}$$

The approach that has been adopted is the Harris-Stevens solution, which works by computing:

$$CM = \frac{det(M)}{Tr(M) + \epsilon} \tag{1.11}$$

where M is the matrix given by $M_{r,c}$, while $det(\cdot)$ and $tr(\cdot)$ are, respectively, the determinant and the trace operators. Equation 1.11. can be rewritten as:

$$CM = \frac{J_x^2 J_y^2 - J_{xy}^2}{J_x^2 + J_y^2 + \epsilon} \tag{1.12}$$

where $J_x^2 = I_x^2 \otimes w$, $J_y^2 = I_y^2 \otimes w$ and $J_{xy} = I_x I_y \otimes w$. The first "partial" corner detection result is obtained by canceling out the pixels with a CM lower than a threshold, and the final result is obtained by computing the local maxima on the "thresholded" CM.

The result of Harris Corner Detection is shown in figure 7.

The corners identified by the algorithm allowed to define two conics $C1$ and $C2$ that are shown in figure 8.

Unfortunately, it was not possible to retrieve the two "natural" cross sections given by the arcades because the Harris detected points on them yielded degenerate conics. I preferred to minimize manual interventions to the essentials, so I decided to use the conics found here.

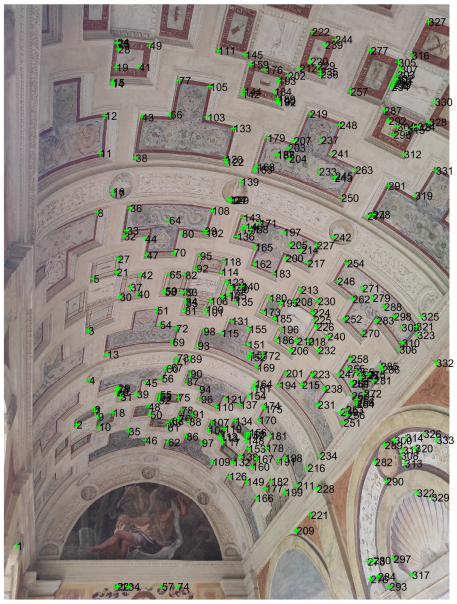

Figure 7: Corners by Harris Algorithm

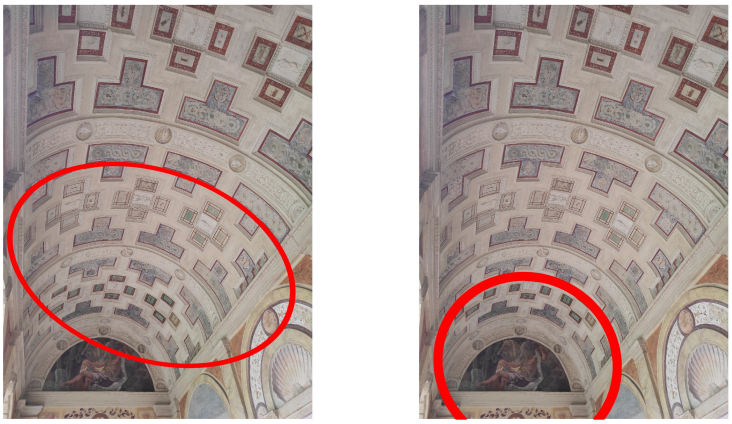

Figure 8: Conics C1 and C2

1.4. Manual Intervention

All manually retrieved points were obtained through a process of 'pseudo' manual intervention. I employed the Sobel Edge Detector binary mask as a reference for the overall structure. The method involved choosing two distinct points on the image, calculating the line passing through these points, and ultimately selecting the point on the intersection between that line and the Sobel binary mask. With the latter method, I ensured a precise selection of the point directly on an edge. Line $l1$ is shown in figure 9

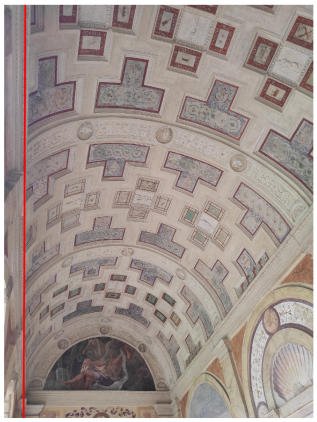

Figure 9: Line l1

1.5. Extraction of the two conics

Geometric element definition

In Euclidean coordinates a conic C is defined by the following equation:

$$aX^2 + bXY + cY^2 + dX + eY + f = 0 \tag{1.13}$$

using replacements: $x = \frac{X}{w}$ and $y = \frac{Y}{w}$, the equation of a conic in homogeneous coordinates is given:

$$ax^2 + bxy + cy^2 + dxw + eyw + fw^2 = 0 \tag{1.14}$$

that in matrix form becomes:

$$x^T C X = 0 \tag{1.15}$$

where the matrix C is:

$$C = \begin{bmatrix} a & \frac{b}{2} & \frac{d}{2} \\ \frac{b}{2} & c & \frac{e}{2} \\ \frac{d}{2} & \frac{e}{2} & f \end{bmatrix}. \tag{1.16}$$

Note that the coefficient matrix C is symmetric. Furthermore, notice that multiplying matrix C by any $\lambda \in \mathbb{R}^+$ leave invariant the equation. Thus, C is a homogeneous representation of a conic, with 6 elements but only 5 degrees of freedom (the five ratios {a:b:c:d:e:f}).

Fit a conic through 5 points

Since the conic has 5 degrees of freedom, we need 5 points to uniquely identify a conic in the projective space $\mathbb{P}^2$. Given the five points: x_1, x_2, x_3, x_4, $x_5 \in \mathbb{P}^2$, we can retrieve the conic's parameters by solving the system:

$$\begin{cases} x_1^T C x_1 &= 0 \\ x_2^T C x_2 &= 0 \\ x_3^T C x_3 &= 0 \\ x_4^T C x_4 &= 0 \\ x_5^T C x_5 &= 0 \end{cases} \tag{1.17}$$

.

Unrolling the equations by means of equation 1.14 for each point $x_i = (x_{i,1}, x_{i,2}, 1)^T$, the system can be rewritten as:

$$A^T C = 0 \tag{1.18}$$

where

$$A = \begin{bmatrix} x_{1,1}^2 & x_{2,1}^2 & ... & x_{5,1}^2 \\ x_{1,1}x_{1,2} & x_{2,1}x_{2,2} & ... & x_{5,1}x_{5,2} \\ x_{1,2}^2 & x_{2,2}^2 & ... & x_{5,2}^2 \\ x_{1,1} & x_{2,1} & ... & x_{5,1} \\ x_{1,2} & x_{2,2} & ... & x_{5,2} \\ 1 & 1 & ... & 1 \end{bmatrix} C = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}. \tag{1.19}$$

Using the latter formulation, it is possible to retrieve the conic:

$$C = RNS(A^T) \tag{1.20}$$

where $RNS(\cdot)$ is the Right Null Space operator that, here, yields ∞^1solutions, all of them identifying the same conic in euclidean coordinates.

2. Point 1

From C1, C2 find the horizon (vanishing) line h of the plane orthogonal to the cylinder axis.

2.1. Procedure

- Intersect the images of C1 and C2 to compute the images of the two circular points of the plane orthogonal to the cylinder axis.
- Compute the horizon line by doing the cross product between the two images of circular points computed in the previous step.

2.2. Extraction of the image of circular points of plane orthogonal to the cylinder axis

In Space (3D) Projective Geometry, each plane π has its own line at infinity $l_\infty(\pi)$ and its own circular points $I(\pi), J(\pi)$, and, parallel planes share the same line at infinity and circular points.
Therefore, the plane that goes through conic $C1$ and the one that goes through conic $C2$, being parallel planes, share the same circular points and the same line at infinity.
Given that, every conic that is on a plane π goes through the circular points $I(\pi), J(\pi)$, and thus we have that both $C1$ and $C2$ goes through the same two circular points.
Since the order of contact is invariant through 2d projective mapping, the images of the two conics $C1'$, $C2'$ intersect each other on the images of the two circular points $I(\pi)', J(\pi)'$.

To compute $I(\pi)'$ and $J(\pi)'$ it is then sufficient to intersect $C1'$ and $C2'$.

Intersection of the image of two conics

Given the equations of the two conics, calculated as in paragraph 1.5.,

$$a_1X^2 + b_1XY + c_1Y^2 + d_1X + e_1Y + f_1 = 0$$
$$a_2X^2 + b_2XY + c_2Y^2 + d_2X + e_2Y + f_2 = 0$$

it is possible to compute their intersection by solving for variables X, Y the system:

$$\begin{cases} a_1X^2 + b_1XY + c_1Y^2 + d_1X + e_1Y + f_1 &= 0 \\ a_2X^2 + b_2XY + c_2Y^2 + d_2X + e_2Y + f_2 &= 0 \end{cases}. \tag{2.1}$$

Note that the two equations are quadratic and, therefore, there will be 4 different solutions but only two of them will refer to the images of circular points. In general, the 4 solutions can contain either one pair of complex conjugate points or two pairs of complex conjugate points[1]. It the first case, the images of the two circular points are exactly the two complex conjugate points. On the contrary, in the second case, due to the ambiguity, it would be necessary to use a third conic to uniquely define the images of circular points.

Experimental Result

The four solutions to the conics' intersection are:

```
s1 = 3×1 complex
10^3 ×
   2.2498 - 4.0141i
   0.0644 - 1.1343i
   0.0010 + 0.0000i

s2 = 3×1 complex
10^3 ×
   2.2498 + 4.0141i
   0.0644 + 1.1343i
   0.0010 + 0.0000i

s3 = 3×1
10^3 ×
   0.4347
   3.3731
   0.0010

s4 = 3×1
10^3 ×
   2.2261
   4.0416
   0.0010
```

Figure 10: Conics' intersections

. There is a single pair of complex conjugate points that are exactly the images of the two circular points.

2.3. Extraction horizon of plane orthogonal to the cylinder axis

Given the images of the two circular points: $I(\pi)'$ and $J(\pi)'$, the vanishing line of the plane π is computed as the line going through those two points:

$$l_\infty(\pi)' = I(\pi)' \times J(\pi)' \tag{2.2}$$

Experimental results

The computed horizon is shown in figure 11.

3. Point 2

From l1, l2, C1, C2 find the image projection a of the cylinder axis, and its vanishing point V.

3.1. Procedure

- Compute the images of the two centers of $C1$ and $C2$.
- Compute the image of the axis by intersecting the images of the two centers.
- Compute the image of the vanishing point V by intersecting l1, l2 and the axis.

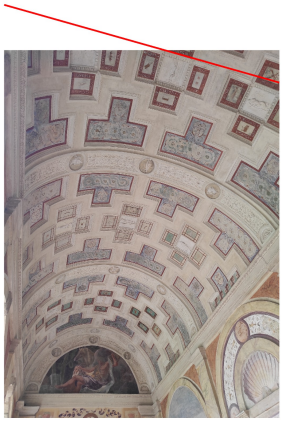

Figure 11: Horizon vertical plane

3.2. Computation of the image of the center of a circumference

Let's consider a circumference in homogeneous coordinates $C \in \mathbb{P}^2$, with center $X = (X_0, Y_0, 1)^T$

$$C = \begin{bmatrix} 1 & 0 & -X_0 \\ 0 & 1 & -Y_0 \\ -X_0 & -Y_0 & -r^2 + X_0^2 + Y_0^2 \end{bmatrix}$$

the polar line of point X w.r.t. circumference C is

$$l = \begin{bmatrix} 1 & 0 & -X_0 \\ 0 & 1 & -Y_0 \\ -X_0 & -Y_0 & -r^2 + X_0^2 + Y_0^2 \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ Y_0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -r^2 \end{bmatrix} = l_\infty \tag{3.1}$$

namely, the line at infinity. By inversion, the center X of a circumference C is:

$$X = C^{-1} \cdot l_\infty. \tag{3.2}$$

The same holds for the images of those geometric elements and, therefore, the image of the center of a circumference is given by:

$$X' = C'^{-1} \cdot l'_\infty \tag{3.3}$$

3.3. Image of the cylinder axis

The image of the cylinder axis is found computing the line going through the images of the two centers of the circumferences.

$$axis = center1 \times center2 \tag{3.4}$$

Experimental results

The image of the cylinder axis is shown in figure 12.

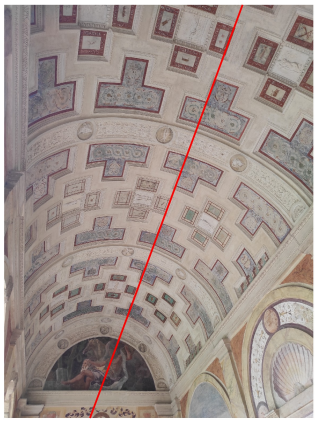

Figure 12: Cylinder Axis

3.4. Vanishing point

Considering the plane π_{ax} on which the cylinder axis lies on, the lines $l1$, $l2$ and $axis$ are all parallel in the planar scene. Therefore, they all intersect each other in the vanishing point v. Thus, their images $l1'$, $l2'$ and $axis'$, meet all together at the image of the vanishing point V. V is then the solution to the the system:

$$\begin{cases} l1^T \cdot V & = 0 \\ l2^T \cdot V & = 0 \\ axis^T \cdot V & = 0 \end{cases} \tag{3.5}$$

that we can rewrite as:

$$\begin{bmatrix} l1^T \\ l2^T \\ axis^T \end{bmatrix} \cdot V = 0 \tag{3.6}$$

However, in practice, due to noise in the measurements, estimations and numerical approximations, the only exact solution to the latter system will be $V = (0, 0, 0)^T$. Therefore, it is necessary to look for the approximate solution that minimize the Algebraic Error:

$$\| \begin{bmatrix} l1^T \\ l2^T \\ axis^T \end{bmatrix} \cdot V \|_2.$$

The approximate solution is obtained via SVD decomposition

$$\begin{bmatrix} l1^T \\ l2^T \\ axis^T \end{bmatrix} = U \cdot S \cdot V^T \tag{3.7}$$

, and is given by the last column of matrix V.

Experimental results

The image of the vanishing point V is shown in figure 13.

4. Point 3

From l1, l2, C1, C2 (and possibly h, a, and V), find the calibration matrix K.

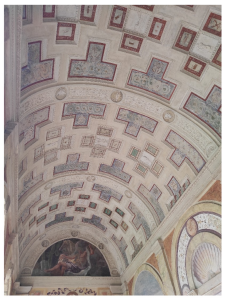

Figure 13: Horizon vertical plane

4.1. Procedure

- Compute the image of the conic dual to the circular points $C'_{D,\infty}$ given the images of the two circular points.
- Compute the homography (H) of the plane orthogonal to the cylinder axis.
- Compute the image of the absolute conic (IAC) ω
- Reconstruct calibration matrix K from ω

4.2. Image of the conic dual to the circular points

In general, in $\mathbb{P}^2$, any degenerate dual conic with rank 2 can be decomposed as:

$$C_D = pq^T + qp^T \tag{4.1}$$

where p and q are two points. If p and q are the circular points, then, the created conic is called the conic dual to the circular points:

$$C_{D,\infty} = IJ^T + JI^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.2}$$

.

Therefore, it is possible to build the image of the conic dual to circular points simply by:

$$C'_{D,\infty} = I'J'^T + J'I'^T \tag{4.3}$$

where I' and J' are the images of the circular points.

4.3. Homography (H)

Knowing the image of the conic dual to the circular points ($C'_{D,\infty}$), and, given that it is also obtained from the homography H as:

$$C'_{D,\infty} = H \cdot C_{D,\infty} \cdot H^T = H \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot H^T \tag{4.4}$$

it is possible to derive H by SVD decomposition of $C'_{D,\infty}$. Ideally, the result would be:

$$C'_{D,\infty} = U \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot V^T \tag{4.5}$$

so that $H = U$. However, in practice, due to errors in the measurements and estimations, the result will likely be:

$$C'_{D,\infty} = U \cdot \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot V^T \tag{4.6}$$

but we can cancel out that errors by doing a further decomposition:

$$C'_{D,\infty} = U \cdot \begin{bmatrix} \sqrt{a} & 0 & 0 \\ 0 & \sqrt{b} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \sqrt{a} & 0 & 0 \\ 0 & \sqrt{b} & 0 \\ 0 & 0 & 1 \end{bmatrix}^T \cdot V^T \tag{4.7}$$

so that:

$$H = U \cdot \begin{bmatrix} \sqrt{a} & 0 & 0 \\ 0 & \sqrt{b} & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.8}$$

4.4. Absolute Conic Ω_∞

Given a generic sphere $S \in \mathbb{P}^3$

$$S = \begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ -X_0 & -Y_0 & -Z_0 & X_0^2 + Y_0^2 - r^2 \end{bmatrix} \tag{4.9}$$

it intersects the plane at infinity $\pi_\infty = (0, 0, 0, 1)^T$, in a conic Ω_∞ called the Absolute Conic.

$$\Omega_\infty = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.10}$$

The absolute conic is the union of all the circular points of all the planes in $\mathbb{P}^3$.

4.5. Calibration Matrix

In the context of Camera Geometry and Single View Geometry, we have that the mapping from the 3D world scene to the 2D image plane is given by a projection described by a 3x4 matrix P:

$$\begin{bmatrix} X_{hom,img} \\ Y_{hom,img} \\ W_{hom,img} \end{bmatrix} = \begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} \\ p_{2,1} & p_{2,2} & p_{2,3} & p_{2,4} \\ p_{3,1} & p_{3,2} & p_{3,3} & p_{3,4} \end{bmatrix} \cdot \begin{bmatrix} X_{hom,wrd} \\ Y_{hom,wrd} \\ Z_{hom,wrd} \\ 1 \end{bmatrix}. \tag{4.11}$$

In general, matrix P can be decomposed as:

$$P = K \cdot [I|0] \cdot \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} \tag{4.12}$$

where the parameters of K are called *internal camera parameters* and the ones of R and C are called *external parameters*. In particular, K is the calibration matrix, while matrices R and C describes the roto-translation necessary to convert coordinates in the *world coordinate frame* to the *camera coordinate frame*. More precisely, R is a 3x3 rotation matrix and C is a 3x1 vector representing the euclidean coordinates of the center of the camera in the world coordinate frame. In our scenario, we are considering the world coordinate frame equal to the camera coordinate frame, therefore R is the identity matrix and C is the zero vector. For a general projective camera, matrix K is:

$$K = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.13}$$

being m_x and m_y the number of pixels per unit distance along respectively coordinate x and y, parameters f_x and f_y are defined as $f_x = f \cdot m_x$ and $f_y = f \cdot m_y$, where f is the focal plane distance. Parameters p_x and p_y represent the offset of the principal point, i.e., the coordinates of the principal point in the image plane. Parameter s is the skew parameter. In our scenario, we are considering a zero-skew camera, therefore matrix is:

$$K = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.14}$$

thus dependent on 4 parameters.
Suppose we are considering a ray defined by a point x and the camera center (line going through x and the camera center) with direction **d** in the world, then all the points on that ray can be written as $\hat{X} = \lambda \mathbf{d}$, with $\lambda \in \mathbb{R}^+$, in the camera Euclidean coordinate frame. Then, all the points on that ray are mapped to the same image point x

$$x = K[I|0] \cdot \begin{bmatrix} \lambda \mathbf{d} \\ 1 \end{bmatrix} \propto K\mathbf{d}. \tag{4.15}$$

Conversely, the direction **d** is obtained from the image point x as $\mathbf{d} = K^{-1} \cdot x$.

The angle between two rays with directions d_1 and d_2, which corresponds to image points x_1 and x_2, is given by:

$$\begin{aligned} cos\theta = \frac{d_1^T \cdot d_2}{\sqrt{d_1^T \cdot d_1}\sqrt{d_2^T \cdot d_2}} = \frac{(K^{-1}x_1)^T \cdot (K^{-1}x_2)}{\sqrt{(K^{-1}x_1)^T \cdot (K^{-1}x_1)}\sqrt{(K^{-1}x_2)^T \cdot (K^{-1}x_2)}} = \\ = \frac{x_1^T (K^{-T}K^{-1}) \cdot x_2}{\sqrt{x_1^T (K^{-T}K^{-1}) \cdot x_1}\sqrt{x_2^T (K^{-T}K^{-1}) \cdot x_2}} \end{aligned} \tag{4.16}$$

4.6. Relationship Calibration matrix and Image of the Absolute Conic

Considering the points on the plane at infinity π_∞, they can be written as $X_\infty = (\mathbf{d}^T, 0)^T$ in homogeneous world coordinate frame, and their image is then given by:

$$x = PX_\infty = KR\mathbf{d}.$$

Therefore, $H = KR$ is the homography that maps points on the plane at infinity to the image plane. Being the absolute conic Ω on the plane at infinity it follows the same mapping, thus:

$$\omega = (KR)^{-T} \cdot I \cdot (KR)^{-1} = K^{-T}RR^{-1}K^{-1} = (KK^T)^{-1} \tag{4.17}$$

where ω is the image of the absolute conic. For a zero-skew camera the latter can be written as:

$$\omega = (KK^T)^{-1} = \begin{bmatrix} a^2 & 0 & -p_x a^2 \\ 0 & 1 & -p_y \\ -p_x a^2 & -p_y & f_y^2 + a^2 p_x^2 + p_y^2 \end{bmatrix} \tag{4.18}$$

thus having only 4 parameters. In order to compute ω is sufficient to impose 4 linearly independent constraints.

4.7. Computation of the Image of the Absolute Conic

(Theory based on [2])
Since $\omega = (KK^T)^{-1}$, it follows from eq. 4.16 that the angle between two rays is given by:

$$cos\theta = \frac{x_1^T \omega \cdot x_2}{\sqrt{x_1^T \omega \cdot x_1}\sqrt{x_2^T \omega \cdot x_2}}. \tag{4.19}$$

As a result, we can obtain the following relationships :

- The vanishing points v_1, v_2 of two lines with orthogonal directions satisfy:

$$v_1^T \omega v_2 = 0 \tag{4.20}$$

- The vanishing line l of a plane and the vanishing point V of a line perpendicular to the plane satisfy:

$$l = \omega v \tag{4.21}$$

  which is equivalent to

$$l \times \omega v = 0. \tag{4.22}$$

The latter results in two linearly independent constraints on ω, and can be used in our scenario as:

$$l_\infty(\pi)' \times \omega \cdot V = 0 \tag{4.23}$$

where $l_\infty(\pi)'$ is the horizon computed at point 2.3 and V is the vanishing point computed at point 3.4.

Given that, the image of the absolute conic ω contains all the images of the circular points, and we can derive the following constraint:

$$I'^T \omega I' = 0. \tag{4.24}$$

Since we know the homography H and the images of the circular points I' and J' of the plane orthogonal to the cylinder axis, we can impose:

$$(HI)^T \omega (HI) = ([h_1, h_2, h_3] \begin{bmatrix} 1 \\ i \\ 0 \end{bmatrix})^T \omega ([h_1, h_2, h_3] \begin{bmatrix} 1 \\ i \\ 0 \end{bmatrix}) = (h_1 + ih_2)^T \omega (h_1 + ih_2) \tag{4.25}$$

that results in the two constraints of vanishing real and imaginary parts:

$$\begin{cases} h_1^T \omega h_1 - h_2^T \omega h_2 & = 0 \\ h_1^T \omega h_2 & = 0 \end{cases}. \tag{4.26}$$

The four constraints to derive ω are therefore:

$$\begin{cases} l_\infty(\pi)' \times \omega \cdot V & = 0 \\ h_1^T \omega h_1 - h_2^T \omega h_2 & = 0 \\ h_1^T \omega h_2 & = 0 \end{cases}. \tag{4.27}$$

Note that the first equation yields two constraints.

4.8. Computation of the calibration Matrix K

Once ω is computed, we can derive the matrix K by Cholensky decomposition of $\omega^{-1} = K \cdot K^T$. Note that, in matlab, Cholensky decomposition of matrix B finds the matrix A such that $B = A^T A$, so, at first, the Cholensky factorization of $(KK^T)^{-1} = K^{-T}K^{-1}$ is applied to find matrix K^{-1}, and then the inverse is applied to retrieve K.

Experimental results

The computed calibration matrix K is shown in figure 14.

```
K = 3×3
10^3 ×
    3.3966         0    1.5527
         0    3.2321    2.2983
         0         0    0.0010
```

Figure 14: Calibration matrix

Note that f_x and f_y are very close to each other so the pixels are almost square pixels.

5. Point 4

From h, K, and V determine the orientation of the cylinder axis wrt the camera reference.

5.1. Theory

Given a known planar object and an image of it taken by a calibrated camera, it is possible to derive its 3D position with respect to the camera reference frame.

The planar object is computed by applying a shape reconstruction (H_{sr}) to the given image. Therefore, we are able to express the coordinates of the two circumferences' centers in the plane reference as in figure 15. Note that we are expressing the points in a 3D reference frame, but the z coordinate is equal to zero for all the points in the plane. Therefore a general point on that plane is

$$x_\pi = \begin{bmatrix} x \\ y \\ 0 \\ w \end{bmatrix} \tag{5.1}$$

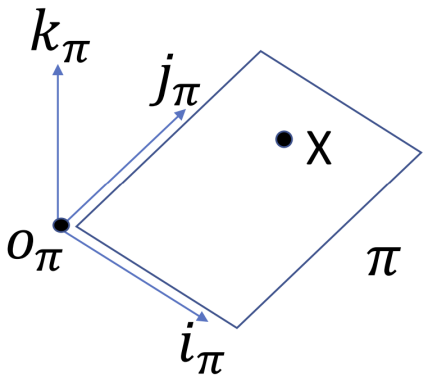

Figure 15: Plane reference

Now the problem consists in estimating a mapping from points in the 3D reference (i_π, j_π, k_π) of figure 15, to the 3D wolrd/camera reference:

$$X_W = \begin{bmatrix} i_\pi & j_\pi & k_\pi & o_\pi \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 0 \\ w \end{bmatrix} \tag{5.2}$$

We know that the real 3D world object corresponding to the planar object of figure 15 is mapped onto the image plane by means of the projection matrix P of equation 4.12 ($x_{image} = P \cdot X_{world}$). Now, we are considering the world reference equal to the camera reference, therefore, the rotation matrix R is equal to the Identity 3x3 matrix and the camera center C is equal to the 3x1 zero vector. The resulting matrix P is therefore

$$P = \begin{bmatrix} K & 0 \end{bmatrix} \tag{5.3}$$

Thus, a point in 3D camera (world) reference frame, is mapped on the image plane as:

$$x_{img} = P \cdot X_W = \begin{bmatrix} K & 0 \end{bmatrix} \cdot X_W \tag{5.4}$$

Substituting X_W from equation 5.2 into equation 5.4, we find how a point on the plane given by (i_π, j_π) is mapped onto the image:

$$x_{img} = K \cdot \begin{bmatrix} i_\pi & j_\pi & o_\pi \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ w \end{bmatrix} \tag{5.5}$$

However, the mapping from the plane $(i_\pi, j_\pi$) to the image is also given by the homography $H = H_{sr}^{-1}$, therefore, the two mappings must be equal and thus we obtain:

$$H = K \cdot \begin{bmatrix} i_\pi & j_\pi & o_\pi \end{bmatrix}. \tag{5.6}$$

Finally, (i_π, j_π, o_π) are obtained

$$\begin{bmatrix} i_\pi & j_\pi & o_\pi \end{bmatrix} = K^{-1} \cdot H. \tag{5.7}$$

The missing component k_π is simply

$$k_\pi = i_\pi \times j_\pi. \tag{5.8}$$

Now, it is possible to map points from the shape reconstructed image to the world/camera reference.

5.2. Practice

5.2.1. Procedure.

- Derive homography (H_{ax}) of the axial plane.
- Map the two center of the circumferences to the plane of figure 15, using H_{ax}^{-1}.
- Derive (i_π, j_π, o_π) from H_{ax} and K using eq. 5.7.
- Compute roto-translation matrix of eq. 5.2. using (i_π, j_π, o_π) from the previous point and k_π computed using eq. 5.8.
- Project the two circumferences' centers in the 3D world reference frame using eq. 5.2.
- The axis vector is given by $X_{W,C1} - X_{W,C2}$
- The axis orientation is computed by doing for each of the frame axis $A_1 = (1,0,0)^T$, $A_2 = (0,1,0)^T$ and $A_3 = (0,0,1)^T$

$$\theta_i = cos^{-1}(\frac{A_i^T \cdot axis}{||A_i||_2||axis||_2}) \tag{5.9}$$

5.2.2. Homography H_{ax}.

We know that the absolute conic Ω_∞ contains all the circular points of all the planes. The same holds for the images of those geometric elements. Therefore, the image of the absolute conic ω contains all the images of the circular points of all the planes. For each plane π, the two circular points (π) and $J(\pi)$, are located on the line at infinity of that plane ($I(\pi)$, $J(\pi) \in l_\infty(\pi)$) and the images of the two circular points are located on the image of the line at infinity ($I'(\pi)$, $J'(\pi) \in l'_\infty(\pi)$). It follows that, to retrieve I' and J' of the axial plane, it is sufficient to intersect the image of the absolute conic with the image of the line at infinity of the axial plane. Note that the line at infinity of the axial plane has been computed as the line passing through vanishing point V and the vanishing point V_h, computed in section 7.2.1.

5.2.3. Experimental results.

Figure 16 shows the 3D position of the cylinder axis and some points of the cylindrical vault with respect to the camera reference frame.

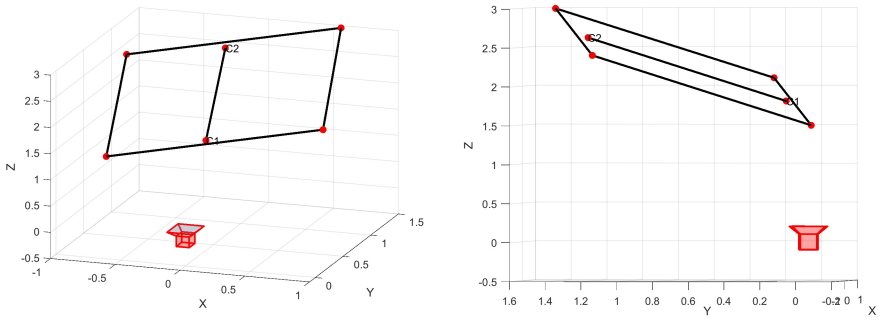

Figure 16: 3D localization

The orientation of the cylinder axis with respect to the camera reference frame axis are, respectively,

- X-axis : 103.05^o
- Y-axis : 38.9^o
- Z-axis : 54.1^o

6. Point 5

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

6.1. Theory

In order to asses the ratio of two lengths of objects in the axial plane, the shape reconstruction of that plane is used (H_{ax}^{-1}), so that the reconstructed axial plane is a similarity mapping of the original axial plane. Once the axial plane has been reconstructed (shape reconstruction), the ratio of lengths of objects in that plane is preserved and, therefore, it is possible to compute the ratio of the two lengths of interest.

6.2. Practice

6.2.1. Procedure.

- Map the needed points using homography (H_{ax}^{-1}).
- Compute the distances between the points of interest.
- Compute the ratio between the distances computed.

6.2.2. Experimental results.

The points have been computed by intersecting the conics with the generatrix lines in order to avoid manual intervention. In figure 17 are shown the selected points to compute the distance ratio:

Figure 17: Points selected in the image p1, center1, p3

In figure 18 it is shown the mapping of these points in the reconstructed axial plane.

As we can see from figure 18, in the reconstructed axial plane the points are deployed creating a rectangle (Almost a square). The distances' ratio is obtained by computing the distance between $c1_r$ and $p1_r$ and the one between $p1_r$ and $p3_r$. The obtained ratio is then: 0.503.

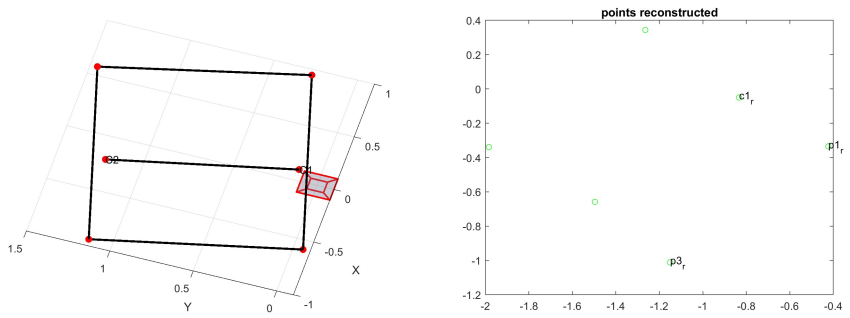

Figure 18: Shape Reconstruction of axial plane

7. Rectification of Cylindrical Surface

Rectification of a cylindrical surface: Plot the unfolding of the part of the surface, included between the two cross sections, onto a plane.

7.1. Theory

The problem consists in finding a way to map all the points on a circular cross section onto a straight line. To do so, the shape reconstruction of the vertical plane is used (plane orthogonal to the cylinder axis). In the latter, all the circular cross sections, which are ellipses in image, are mapped to circumferences. Thus, given a point $X = [x, y, w]^T$ on the cylinder axis in the reconstructed image, it is possible to retrieve all the points belonging to the circumference that has X as center. To do so the only missing piece is the radius R of that specific circular cross section in the reconstructed axial plane; its computation will be explained later.

The unfolding is done by scanning all the points of the cylinder axis in the image, mapping them on the shape reconstructed image, retrieving the radius of the specific circular cross section for a specific axial point, collecting all the points of the circumference, mapping those points in the original image, and, finally, copying them on a straight line. A more clear explanation of the procedure is provided in the pseudocode 1. All the lines are then linked together, creating a rectangle. The result is shown in figure 11.

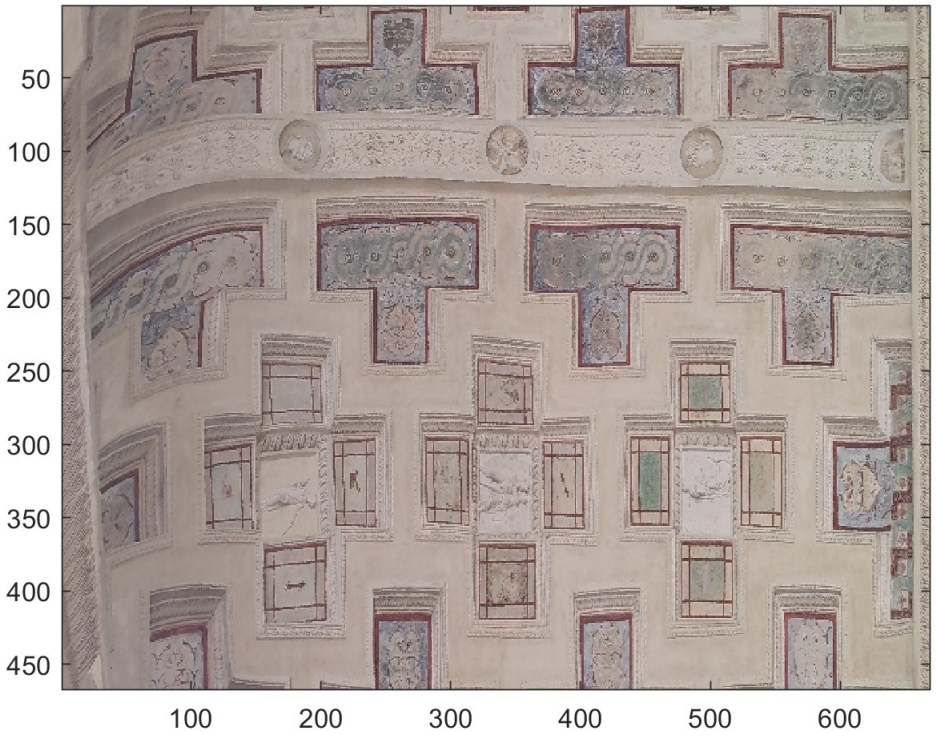

Figure 19: Unfold of the surface between the two circular cross sections

7.2. Practice

7.2.1. Computation of the horizontal vanishing point V_h.

To determine the vanishing point of lines within the axial plane that are perpendicular to the generatrix lines l1 and l2, we calculate lines l3 and l4, that are shown in figure 20. They are computed by using

Figure 20: Lines l3, l4

the points shown in section 5.2.3, were were retrieved as intersections between the conics and the generatrix lines. The vanishing point V_h is obtained by intersecting lines l3 and l4.

7.2.2. Computation of the Radius.

Given a point on the cylinder axis, it is possible to compute the radius of the circular cross section of the reconstructed image which has that point as center. To do so, it is first computed the line L passing through that point and the vanishing point V_h calculated in section 7.2.1. Then, a point e is computed by intersecting line L with the generatrix l1. Point e and the given point are both mapped onto the shape reconstructed image. The radius is given by the distance between those two points in the reconstructed image.

7.2.3. Computation of points in the circular cross section.

Given a point on the axis, it is mapped on the shape reconstructed image and the radius of the circular cross section is derived as explained in section 7.2.2. Therefore, we have now the center of the circumference ,$\hat{X}_{c,i} = (x_{c,i}, y_{c,i})^T$ and its radius R, hence, all points of the circular cross section (X_i, Y_i) are computed by changing the angle $\theta \in [0, \pi]$ in the following equation

$$\begin{cases} X_i = R \cdot \cos(\theta_i) + x_{c,i} \\ Y_i = R \cdot \sin(\theta_i) + y_{c,i} \end{cases} . \tag{7.1}$$

All the points are then mapped back to the original image, so that we retrieve all the ellipses.

7.3. Procedure - pseudocode

Algorithm 1 Cylindric Surface Rectification

P $\leftarrow$ {points on the cylinder axis in the original image}
R $\leftarrow [\emptyset]$ ## Empty rectangle
$i \leftarrow 1$
for p in P **do**
  $\hat{p} \leftarrow H_{sr} \cdot p$ ## Image of p in the reconstructed scene
  $L \leftarrow V_h \times p$ ## Line going through the horizontal vanishing point V_h and the point p
  $e \leftarrow L \times l_1$ ## Intersection point between L and the generatrix $l1$
  $\hat{e} \leftarrow H_{sr} \cdot e$ ## Image of e in the reconstructed scene
  $Radius \leftarrow \|\hat{e} - \hat{p}\|_2$ ## Radius of the circumference in with center p in the reconstructed image
  $R(i,:) \leftarrow$ {points in the reconstructed image on the semi-circle (0-π) with center = $\hat{p}$ and radius = $Radius$
  $i \leftarrow i + 1$
end for
Return R

References

[1] Long Quan Guang Jiang. Single axis geometry by fitting conics, 2002.

[2] Andrew Zisserman Richard Hartley. Multiple view geometry in computer vision, 2003.