POLITECNICO
MILANO 1863

# Image Analysis and Computer Vision

Homework 2022-2023

Supervisor: Professor Vincenzo Caglioti

By: Saeed Hadizadeh Aghuei (10753621)

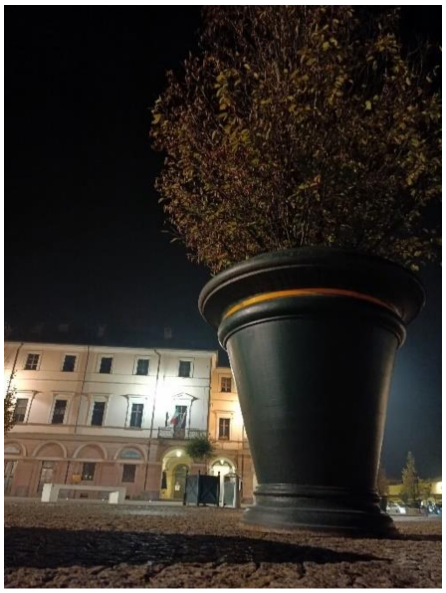
Jan 2023

## Part A. Feature Extraction

Q. Consider the real image below. Using feature extraction techniques (including those implemented in MATLAB) plus possible manual intervention, extract both the straight lines of the cone apparent contour and image of useful circular cross sections.

### A.1 Edge Detection

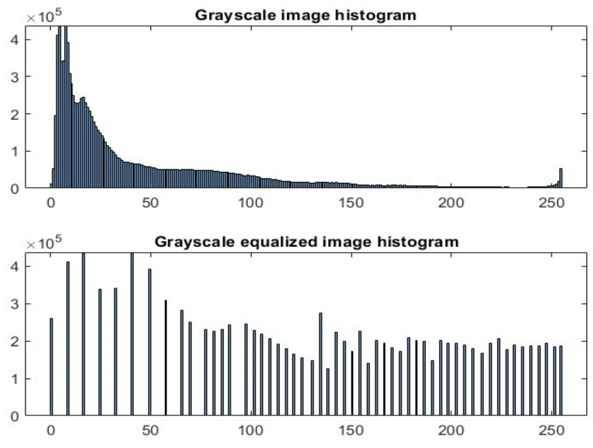
The given image is in RGB format. To obtain better result from edge detection algorithms, it is needed to do some pre-processing. First the original image is converted to grayscale image.

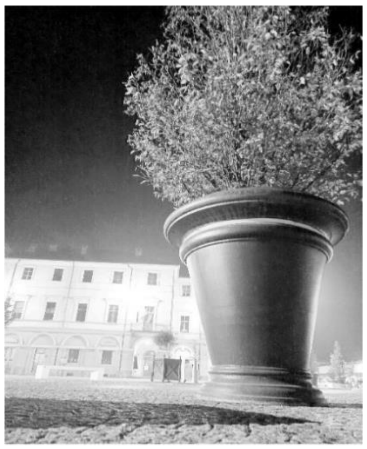


1. Original image

2. Grayscale image

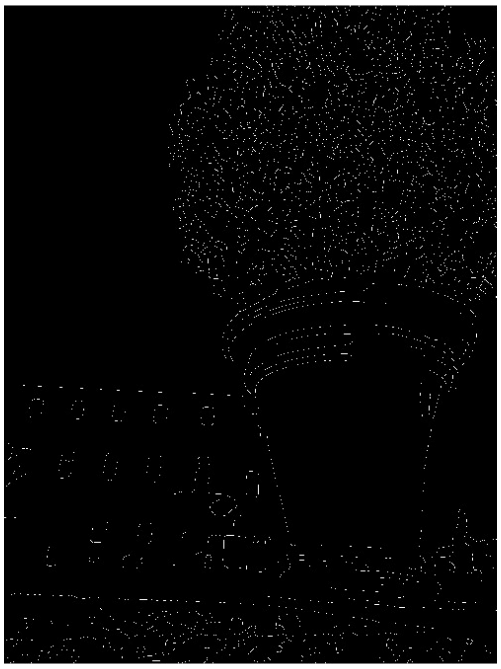
The grayscale image is dark and by looking at histogram of image, it's evident that pixel's intensities don't cover the whole range with uniform distribution and there is peak of pixels with low light intensity. An image intensity transformation called histogram equalization is applied to image and the output is shown in image 3. Histogram equalization maps histogram of input image to a uniform histogram of output image.

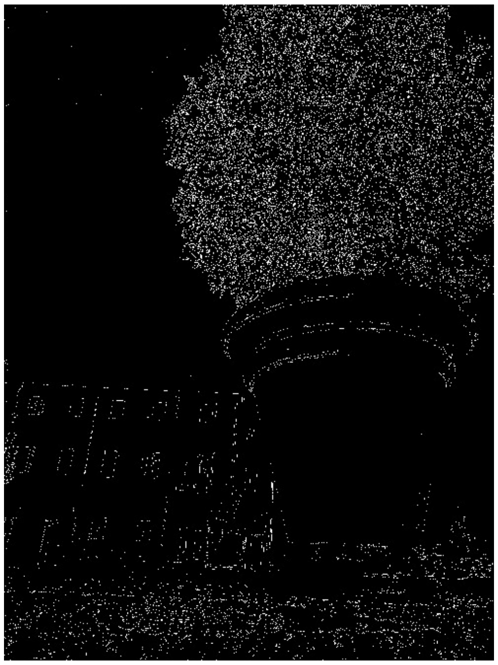


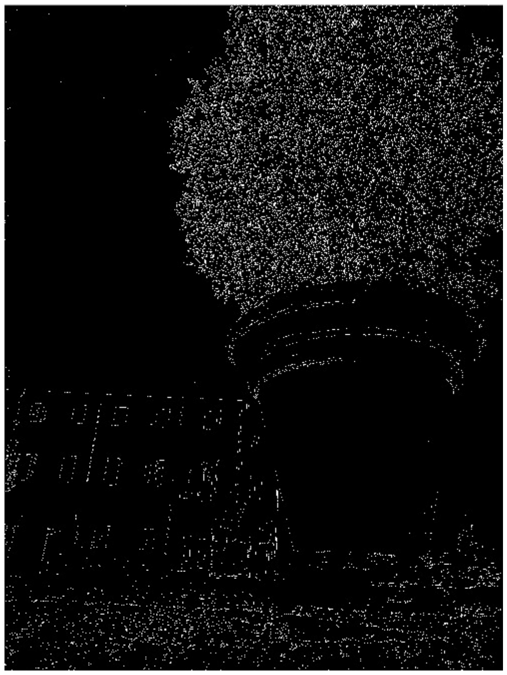


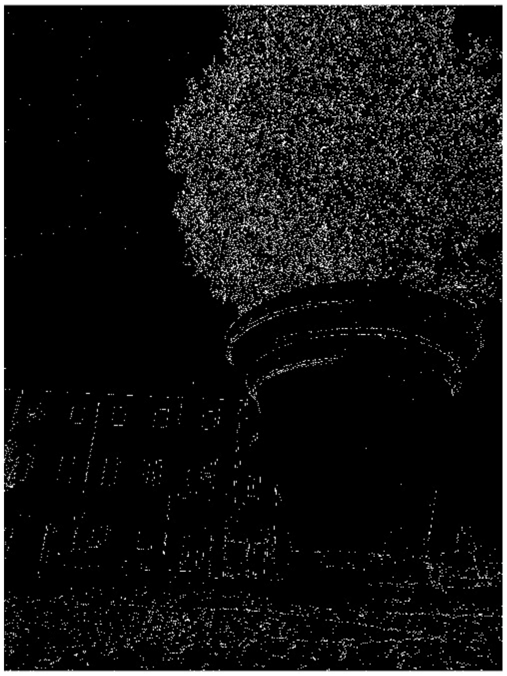
3. Histogram equalized image

Now, the image provides more brightness and more spread histogram, so it's more acceptable for edge detection. There exist several edge detections algorithms which all of them are based on differentiating filters. Derivatives are used to highlight intensity discontinuities in an image and to deemphasize regions with slowly varying intensity levels. Here, the outputs of four different well-known edge detection algorithms are compared.


4. Canny edge detection


5. Sobel edge detection

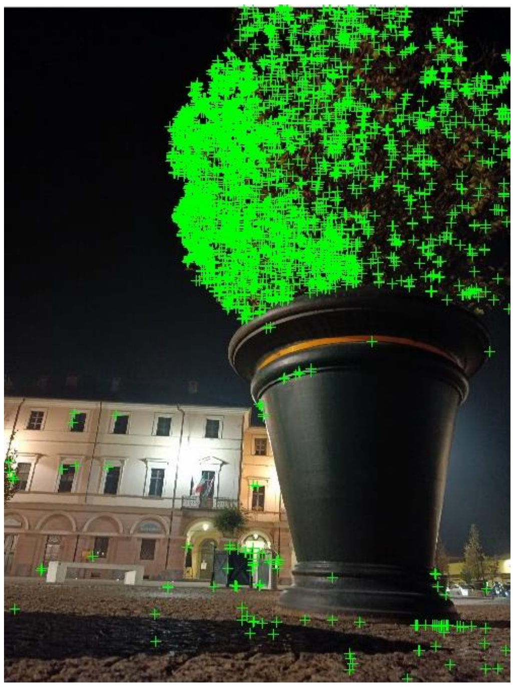

6. Prewitt edge detection


7. Roberts edge detection

After comparison of these four outputs, we can conclude that in this image the Canny edge detection provides better results, so I decided to continue with the output of Canny edge detection.

## A.2 Corner Detection

A point is interesting when the image content around there is dissimilar from the neighboring ones. Corners usually can be considered as key points. The Harris corner detector is a corner detection operator that is commonly used in computer vision algorithms to extract corners and infer features of an image. Harris corner detector takes the differential of the corner score into account with reference to direction directly.



8. image corners

If the points on the top of the flower pot are ignored, the remaining points provide a good outcome from the corner detection algorithm.

## A.3 Line and Ellipse Detection

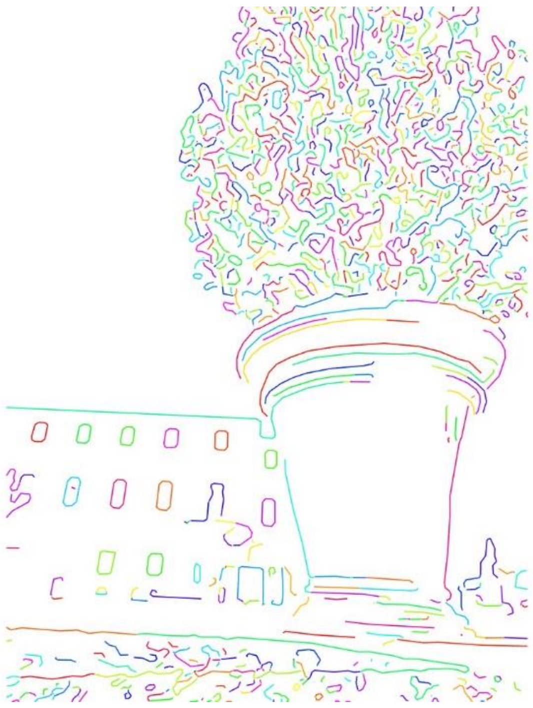
The output binary edge image in part A.2 is exposed to 3 steps:

1. Function edgelink which provides a cell array of edgelists. This function links edge points together into lists of coordinate pairs. Where an edge junction is encountered the list is terminated and a separate list is generated for each of the branches.

2. Function lineseg that fit line segments to the edgelists in seglist array. This function takes each array of edgepoints in edgelist, finds the size and position of the maximum deviation from the line

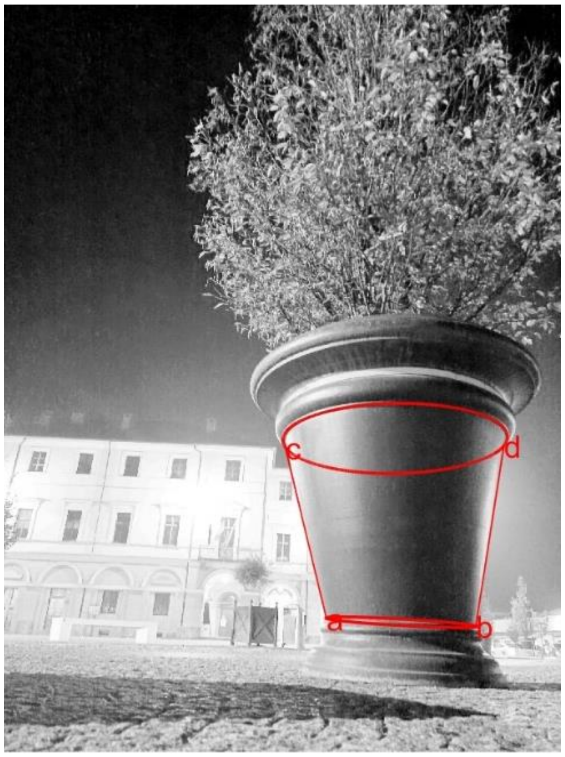
that joins the endpoints, if the maximum deviation exceeds the allowable tolerance the edge is shortened to the point of maximum deviation and the test is repeated. In this manner each edge is broken down to line segments, each of which adhere to the original data with the specified tolerance.

3. function drawedgelist which draw the fitted line segments stored in seglist



9. Colored edge segments

Figure 9 demonstrates the edges of the picture highlighted and it may be suitable image to select the points manually for fitting the desired ellipses and lines. In addition, there exist functions that can apply fitting procedure automatically. But still there is huge error in fitting and the output segments doesn’t provide good accuracy. Therefore, I decided to select points manually from histogram equalized image to increase accuracy and better results. The ellipses are fitted through 5 points and the points are selected in the way that first and last point for each conic are the endpoints of side lines.



10. image of desired lines and ellipses

## Part 2. Geometry and Calibration

**Q1.** From $C1$, $C2$ find the horizon (vanishing) line $h$ of the plane orthogonal to the cone axis.

**Theory:**

From theory we know that all circumferences cross the line at infinity $l_\infty$ at the same two points I and J. $C_1$ and $C_2$ are the images of circumferences. Therefore, they intersect the image of the line at infinity at points $I'$ and $J'$ which are the images of circular points. The line that crossing these two points is the horizon. By intersecting $C_1$ and $C_2$ we can find $I'$ and $J'$. We have 4 unknowns, so we need these 2 equations which are listed below:

$$I'^T C_1 I' = 0$$

$$I'^T C_2 I' = 0$$

Each of above equations leads to two constraints which are real=0 and imaginary=0 parts. We define $I'$ , $J'$ , $C_1$ and $C_2$ as following:

$$I' = \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \qquad J' = \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \qquad C_1 = \begin{bmatrix} a_1 & b_1/2 & d_1/2 \\ b_1/2 & c_1 & e_1/2 \\ d_1/2 & e_1/2 & f_1 \end{bmatrix} \qquad C_2 = \begin{bmatrix} a_2 & b_2/2 & d_2/2 \\ b_2/2 & c_2 & e_2/2 \\ d_2/2 & e_2/2 & f_2 \end{bmatrix}$$

Then these equations are derived:

$$a_1x_1^2 + b_1x_1y_1 + c_1y_1^2 + d_1x_1 + e_1y_1 + f_1 = 0$$

$$a_1x_2^2 + b_1x_2y_2 + c_1y_2^2 + d_1x_2 + e_1y_2 + f_1 = 0$$

By solving these equations, we can obtain $I'$ and $J'$, so the horizon is as below:

$$h = I' \times J'$$

**MATLAB:**

The two quadratic equations are solved by solve function.

```
eq1 = a1*x^2 + b1*x*y + c1*y^2 + d1*x + e1*y + f1;
eq2 = a2*x^2 + b2*x*y + c2*y^2 + d2*x + e2*y + f2;

eqns = [eq1 ==0, eq2 ==0];
S = solve(eqns, [x,y]);
```

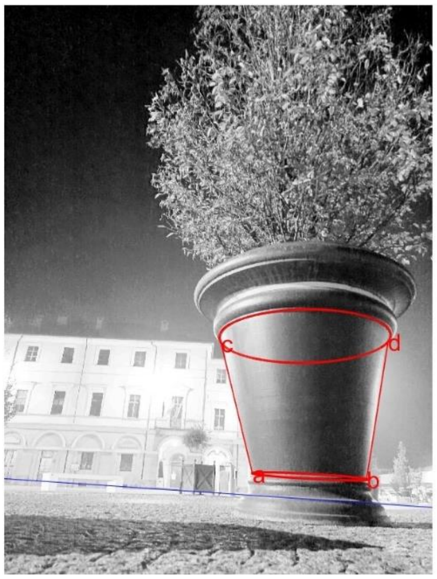
The above solve function results 4 solutions. It will be proved later that s3 and s4 are acceptable.

```
s1 = [double(S.x(1));double(S.y(1));1];
s2 = [double(S.x(2));double(S.y(2));1];
s3 = [double(S.x(3));double(S.y(3));1];
s4 = [double(S.x(4));double(S.y(4));1];
```

```
II =                          JJ =                           horizon =

   1.0e+03 *                     1.0e+03 *                           2.0235e-05
                                                                    -3.0095e-04
   1.2223 - 3.3694i              1.2223 + 3.3694i                    1
   3.4050 - 0.2266i              3.4050 + 0.2266i
   0.0010 + 0.0000i              0.0010 + 0.0000i
```

To draw a line given in homogenous coordinates at least two points on the line are needed. The hline function first specify that the line is horizontal or vertical approximately. If the line is vertical, the function obtains 2 points by intersecting the line with the line of first and last rows of the image. If it's horizontal, the function obtains 2 points by intersection of the line and the lines of first and last column of the image. The blue line which is depicted in image 11 is the horizon line.

```
hline(horizon);
```

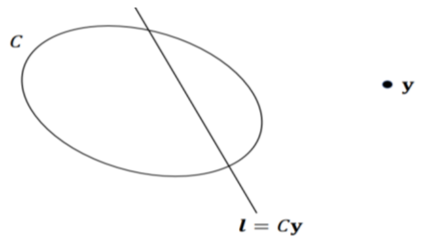


11. depicted Horizon

**Q2.** From $l1$, $l2$, $C1$, $C2$ find the image projection $a$ of the cone axis.

**Theory:**

Given a point $\mathbf{y}$ and a conic $C$ in the plane, the line $\boldsymbol{l} = C\mathbf{y}$ is called the polar line of point $\mathbf{y}$ with respect to the conic $C$.



It can be proven that polar line of center of a circumference is the line at infinity. Here the line at infinity and the circumferences are given, therefore:

$$l_\infty = Cy$$

$$y = C^{-1} l_\infty$$

and in this problem:

$$C1_{center} = C_1^{-1} h$$

$$C2_{center} = C_2^{-1} h$$

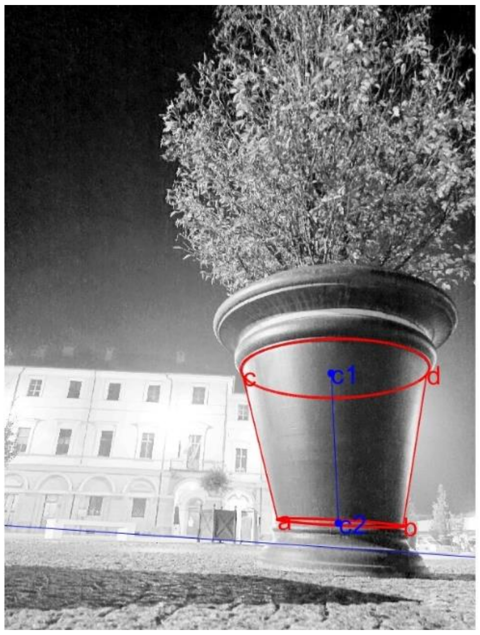
The cone axis is the line between two centers.

$$a = lc1c2 = C1_{center} \times C2_{center}$$

MATLAB:

```
c1_center=inv(C1)*horizon;
c2_center=inv(C2)*horizon;
lc1c2=hcross(c1_center,c2_center);

lc1c2 =

     -5.0755e-04
      2.4203e-05
      1
```

The line between points c1 and c2 shows the image projection of cone axis a.



12. Horizon and cone axis

**Q3**. From $l1$, $l2$, $C1$, $C2$ (and possibly $h$ and $a$), find the calibration matrix $K$.

**Theory:**

The image is taken by a natural camera and calibration matrix has only three unknown parameters namely the focal distance $f$ and the two coordinates $Uo$, $Vo$ of the principal point, therefore we need at least three constraints.

The image of absolute conic contains image of circular points:

$$\mathrm{I}'^{\mathrm{T}}\omega\mathrm{I}' = 0$$

But this equation is not used directly, and these changes are applied:

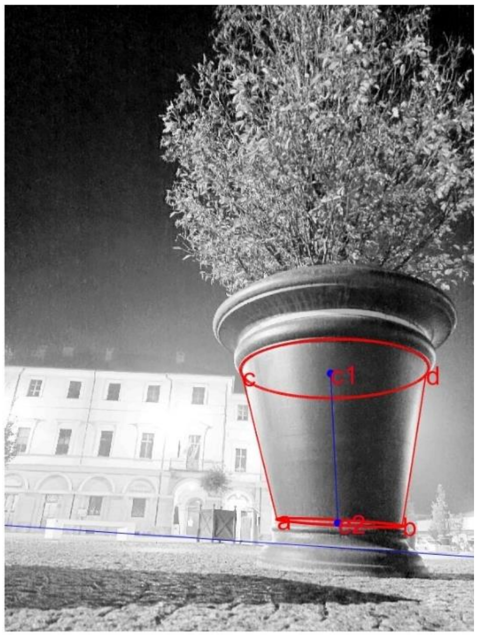
$$\mathrm{I}' = \mathrm{H}_{\mathrm{R}}^{-1}\mathrm{I} = \mathrm{H}_{\mathrm{R}}^{-1}\begin{bmatrix}1\\ \mathrm{i}\\ 0\end{bmatrix} = [\mathrm{h}_1 \quad \mathrm{h}_2 \quad \mathrm{h}_3]\begin{bmatrix}1\\ \mathrm{i}\\ 0\end{bmatrix} = \mathrm{h}_1 + \mathrm{i}\mathrm{h}_2 \ , \qquad (\mathrm{h}_1 + \mathrm{i}\mathrm{h}_2)^{\mathrm{T}}\omega(\mathrm{h}_1 + \mathrm{i}\mathrm{h}_2) = 0$$

$$\mathrm{h}_1^{\mathrm{T}}\omega\mathrm{h}_2 = 0 \qquad (1)$$

$$\mathrm{h}_1^{\mathrm{T}}\omega\mathrm{h}_1 - \mathrm{h}_2^{\mathrm{T}}\omega\mathrm{h}_2 = 0 \qquad (2)$$

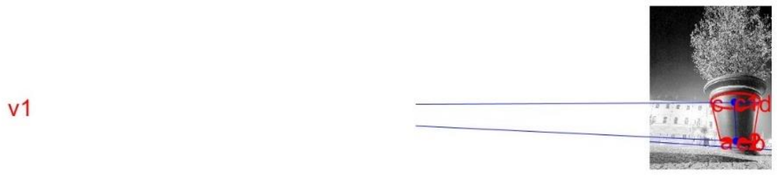
Each right cone has the feature of axial symmetry and cone axis is the symmetry axis. When an image is taken from a right cone, the symmetry is converted into planar symmetry. It means the points on contour generator which are at same cross section are symmetric points. The symmetry plane is the back-projection of the image of cone axis. The corresponding symmetric point of each point on contour generator is a point with same distance at other side of symmetry plane and the line between two symmetric points is perpendicular to symmetry plane.

We name the endpoints of $l_1$ and $l_2$ as it is shown is image 13.



13. image of horizon, cone axis and contour lines

Points $(a, b)$ and $(c, d)$ are symmetric points and we can conclude that lines $l_{ab}$ and $l_{cd}$ are parallel since they both are orthogonal to same plane. Therefore, they intersect at vanishing point $V_1$. Here, we can verify that the vanishing point $V_1$ in a line at horizon.



14. vanishing point V1

The symmetry plane which mentioned before, is a plane that contains camera optical center O and cone axis. So, we can say the direction of vanishing point $V_1$ is orthogonal to any direction from point O to cone axis. It is worth to mention that all points on a viewing ray through O share the same image. Consider a point at infinity along this line:

$$X = \begin{bmatrix} d \\ 0 \end{bmatrix}$$

The corresponding image point is:

$$u = \begin{bmatrix} M & m \end{bmatrix} \begin{bmatrix} d \\ 0 \end{bmatrix} = Md$$

Then the corresponding direction to the point is:

$$d = M^{-1}u$$

So, we can use the points on the image of the cone axis as below:

$$d_1 = M^{-1}o_1$$

$$d_2 = M^{-1}o_2$$

$$d_3 = M^{-1}v_1$$

Which $O_1$ and $O_2$ are the centers of conics. Directions ($d_3$,$d_1$) and ($d_3$,$d_2$) are perpendicular so if we put these directions in this formula:

$$\cos(\vartheta) = \frac{d_1^T d_2}{\sqrt{(d_1^T d_1)(d_2^T d_2)}} = 0$$

these constraints can be concluded:

$$v_1^T \omega o_1 = 0 \qquad (3)$$

$$v_1^T \omega o_2 = 0 \qquad (4)$$

So, we have enough constraints to compute the calibration matrix.

In case of square pixels w (image of absolute conic) has the form:

$$\omega = \begin{bmatrix} \omega_1 & 0 & \omega_2 \\ 0 & \omega_1 & \omega_3 \\ \omega_2 & \omega_3 & \omega_4 \end{bmatrix}$$

Here, constraints (2), (3) and (4) are used to estimate calibration matrix. which each of them generates linear equations in the elements of $\omega$ . These equations are stacked together to form an equation $A\omega = 0$ , where $A$ is a $3 \times 4$ matrix.

The vector $\omega$ is obtained as the null vector of $A$ , and this determines $\omega$. The matrix $K$ is obtained from $\omega = (KK^T)^{-1}$ by Cholesky factorization, followed by inversion.

**MATLAB:**

The rectifying homography can be estimated by SVD decomposition of degenerate dual conic to the image of circular points.

```
imDCCP = II*JJ' + JJ*II';
imDCCP = imDCCP./norm(imDCCP);
[U,D,V] = svd(imDCCP);
D(3,3) = 1;
A = U*sqrt(D);
H_R=inv(A);
```

This matrix H_R is the shape reconstruction homography which converts the taken image to an image similar to planar face. But in the formula of constraints the inversion of the matrix H_R is required.

```
H=inv(H_R);
```

The vanishing point V1 is computed as below:

```
lab=hcross(a,b);
lcd=hcross(c,d);
v1=hcross(lab,lcd);
```

Now, the matrix A of coefficients is built:

```
v2=c1_center;
v3=c2_center;
A=[v1(1)*v3(1)+v1(2)*v3(2)   v1(1)*v3(3)+v3(1)*v1(3)
v1(2)*v3(3)+v3(2)*v1(3)   v1(3)*v3(3); H(1)^2+H(2)^2-H(4)^2-H(5)^2
2*H(1)*H(3)-2*H(4)*H(6)  2*H(2)*H(3)-2*H(5)*H(6)  H(3)^2-H(6)^2;
v1(1)*v2(1)+v1(2)*v2(2)      v1(1)*v2(3)+v2(1)*v1(3)
v1(2)*v2(3)+v2(2)*v1(3)  v1(3)*v2(3)];
```

The right null space of matrix A gives the w matrix elements:

```
B=null(A);
w=[B(1) 0 B(2);
    0   B(1) B(3);
    B(2) B(3) B(4)];
```

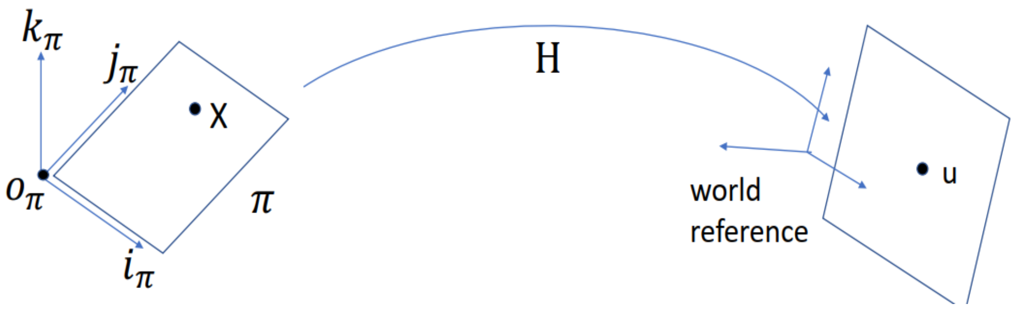
And finally the calibration matrix K is:

```
K=inv(chol(w));

K =

   1.0e+03 *

    2.8553         0    1.5786
         0    2.8553    1.6373
         0         0    0.0010
```

Q4. From $h$ and $K$, determine the orientation of the cone axis with respect to the camera reference.

**Theory:**

Consider a point on a known planar object and camera reference on the world reference. These relations can be obtained:



$$\mathbf{X}_\pi = \begin{bmatrix} x \\ y \\ 0 \\ w \end{bmatrix} \qquad \mathbf{X}_\mathbf{w} = \begin{bmatrix} \mathrm{i}_\pi & \mathrm{j}_\pi & \mathrm{k}_\pi & \mathrm{o}_\pi \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \\ w \end{bmatrix}$$

$$\mathbf{X}_\mathbf{w} = \begin{bmatrix} \mathrm{i}_\pi & \mathrm{j}_\pi & \mathrm{k}_\pi & \mathrm{o}_\pi \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \\ w \end{bmatrix} = \begin{bmatrix} \mathrm{i}_\pi & \mathrm{j}_\pi & \mathrm{o}_\pi \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} \mathrm{i}_\pi & \mathrm{j}_\pi & \mathrm{o}_\pi \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}_\pi$$

camera reference on the world reference

$$\mathrm{P} = [\mathbf{KR} \quad \mathbf{KRt}] = [\mathbf{K} \quad \mathbf{0}]$$

$$\mathbf{u} = \mathrm{P}\mathbf{X}_\mathbf{w} = [\mathbf{K} \quad 0] \begin{bmatrix} \mathrm{i}_\pi & \mathrm{j}_\pi & \mathrm{o}_\pi \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}_\pi = \mathbf{K}[\mathrm{i}_\pi \quad \mathrm{j}_\pi \quad \mathrm{o}_\pi]\mathbf{x}_\pi$$

plane $\pi$ to image homography $\mathrm{H} = \mathbf{K}[\mathrm{i}_\pi \quad \mathrm{j}_\pi \quad \mathrm{o}_\pi]$

$$\boldsymbol{Q} = [\boldsymbol{i}_\pi \quad \boldsymbol{j}_\pi \quad \boldsymbol{o}_\pi] = \boldsymbol{K}^{-1}\boldsymbol{H}$$

The matrices K and H are given from previous points; therefore, matrix Q can be computed, and the Roto-translation matrix is:

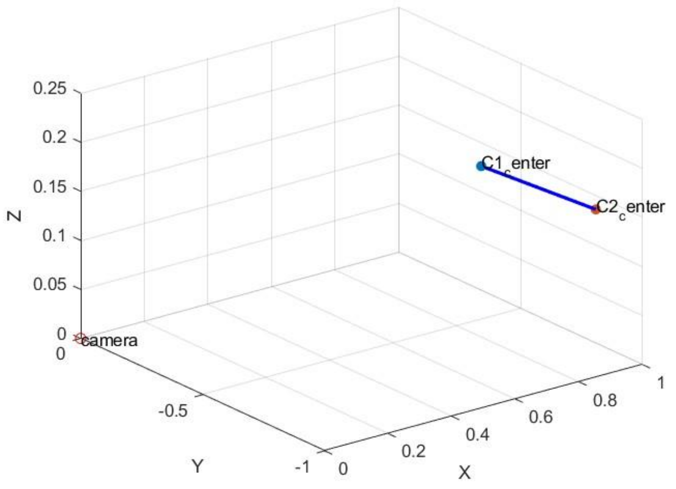
$$\mathbf{R\_t} = \begin{bmatrix} \boldsymbol{i}_\pi & \boldsymbol{j}_\pi & \boldsymbol{k}_\pi & \boldsymbol{o}_\pi \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \qquad \boldsymbol{k}_\pi = \boldsymbol{i}_\pi \times \boldsymbol{j}_\pi$$

To obtain the cone axis coordinate in camera reference frame, the two centers of ellipses are considered as known planar points. It is trivial that the line between these two points in camera reference (world reference) frame is the cone axis.

$$C1_{center\_3D} = R\_t * \begin{bmatrix} C1_{center}(1) \\ C1_{center}(2) \\ 0 \\ C1_{center}(3) \end{bmatrix} \qquad C2_{center\_3D} = R\_t * \begin{bmatrix} C2_{center}(1) \\ C2_{center}(2) \\ 0 \\ C2_{center}(3) \end{bmatrix}$$

**MATLAB:**

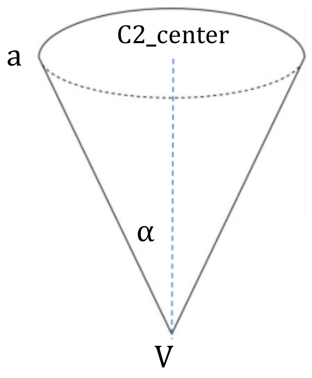
```
Q=inv(K)*H;
i_pi=Q(:,1);
j_pi=Q(:,2);
O_pi=Q(:,3);
k_pi=cross(i_pi,j_pi);
R_t=[i_pi j_pi k_pi O_pi;
     0     0    0     1]

c1_3d=R_t*[c1_center(1);c1_center(2);0;c1_center(3)]
c2_3d=R_t*[c2_center(1);c2_center(2);0;c2_center(3)]

c1_3d =

    0.5099
   -0.9771
    0.2428
    1.0000

c2_3d =

    0.8534
   -0.9991
    0.1709
    1.0000
```



5. How would you use $K$, $h$, the axis orientation and the image $V$ of the cone vertex in order to compute the cone semi-aperture angle α?

**Theory:**

The cone semi-aperture angle is a measure of the angle between the central axis of a cone and the line connecting the vertex of the cone to the point on the circular base.



The point V in the image can be obtained as cross product of contour lines ac and bd.

$$\mathrm{V} = \mathrm{ac} \times bd$$

By referring to point 4, the points a and V can be localized in the camera reference frame.

$$a_{3D} = R\_t * \begin{bmatrix} a(1) \\ a(2) \\ 0 \\ a(3) \end{bmatrix}$$

$$V_{3D} = R\_t * \begin{bmatrix} V(1) \\ V(2) \\ 0 \\ V(3) \end{bmatrix}$$

So, we have 3 points in 3D homogeneous coordinates in following form:

$$a_{3D} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ w_1 \end{bmatrix} \qquad V_{3D} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ w_2 \end{bmatrix} \qquad C2_{center\_3D} = \begin{bmatrix} c2_1 \\ c2_2 \\ c2_3 \\ w_3 \end{bmatrix}$$

These 3 points are normalized by their 4th element and as result we have 3 points in 3D cartesian coordinates.

$$a_{3Dc} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \qquad V_{3Dc} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \qquad C2_{center\_3Dc} = \begin{bmatrix} c2_1 \\ c2_2 \\ c2_3 \end{bmatrix}$$

Then the two 3D vectors $\overline{a_{3Dc}V_{3Dc}}$ and $\overline{C2_{center\_3Dc}V_{3Dc}}$ can be obtained. Once we have this two vectors, the angle between them can be computed as following:

$$A\,.B = |A||B|\cos{(\theta)}$$

$$A = \overline{a_{3Dc}V_{3Dc}} = \begin{bmatrix} v_1 - a_1 \\ v_2 - a_2 \\ v_3 - a_3 \end{bmatrix}$$

$$B = \overline{C2_{center\_3Dc}V_{3Dc}} = \begin{bmatrix} v_1 - c2_1 \\ v_2 - c2_2 \\ v_3 - c2_3 \end{bmatrix}$$

A and B are the vectors with angle theta between them.

**MATLAB:**

```
lac=hcross(a,c);
lbd=hcross(b,d);

V=hcross(lac,lbd);

V =

   1.0e+03 *

    2.2279
    5.3168
    0.0010
```

Obtaining points in 3D cartesian coordinates:

```
V_3d=R_t*[V(1);V(2);0;V(3)];
a_3d=R_t*[a(1);a(2);0;a(3)];
V_3d=V_3d./V_3d(4);
a_3d=a_3d./a_3d(4);

V_3dc=V_3d(1:3,:);
a_3dc=a_3d(1:3,:);
c2_3dc=c2_3d(1:3,:);
```

```
V_3dc =

    1.5773
   -1.0461
    0.0183

a_3dc =

    0.8039
   -0.9300
    0.2864


c2_3dc =

    0.8534
   -0.9991
    0.1709
```

Calculation of vectors:

```
vec_V_3dc_a_3dc=a_3dc-V_3dc;
vec_V_3dc_c2_3dc=c2_3dc-V_3dc;
A=vec_V_3dc_a_3dc;
B=vec_V_3dc_c2_3dc;
```

And finally, angle $\alpha$ is:

```
cos_theta=(dot(A,B)/(norm(A)*norm(B)));
alpha=rad2deg(acos(cos_theta))


alpha =

    8.4344
```

**Conclusion:**

The photo was taken in a poorly lit setting, making it difficult for feature detection algorithms to produce accurate results. To improve the outcome, a technique called Image histogram equalization is applied as a pre-processing step. Despite this, the image still struggles with detecting lines and ellipses due to the low light. Additionally, the corner detection algorithm produces many points that do not provide any meaningful information about the top of the pot. To minimize errors in the computation, the contour lines and ellipses are selected by hand.

The lines a and h and calibration matrix K are as following:

$$a = \begin{bmatrix} -5.0575 \times 10^{-4} \\ 2.4203 \times 10^{-5} \\ 1 \end{bmatrix} \qquad h = \begin{bmatrix} 2.0235 \times 10^{-5} \\ -3.0095 \times 10^{4} \\ 1 \end{bmatrix} \qquad K = \begin{bmatrix} 2855 & 0 & 1578 \\ 0 & 2855 & 1637 \\ 0 & 0 & 1 \end{bmatrix}$$

It can be concluded from the calibration matrix that the image is taken by a zero-skew natural camera. Furthermore, the semi-aperture angle is 8.43 degree.