22AIE313 Computer Vision & Image Understanding (2-1-3-4)

Dr. Aiswarya S Kumar
Dept. of Computer Science & Engineering (AI)
Amrita School of Computing

Camera Calibration

The image processing or computer vision field frequently uses the term “camera calibration.”

It is the process of determining specific camera parameters in order to complete operations with specified performance measurements.

This involves obtaining all the necessary information such as parameters or coefficients of the camera to establish an accurate relationship between a 3D point in the real world and its corresponding 2D projection in the image captured by the calibrated camera.

Camera Calibration

Types of Camera Calibration:

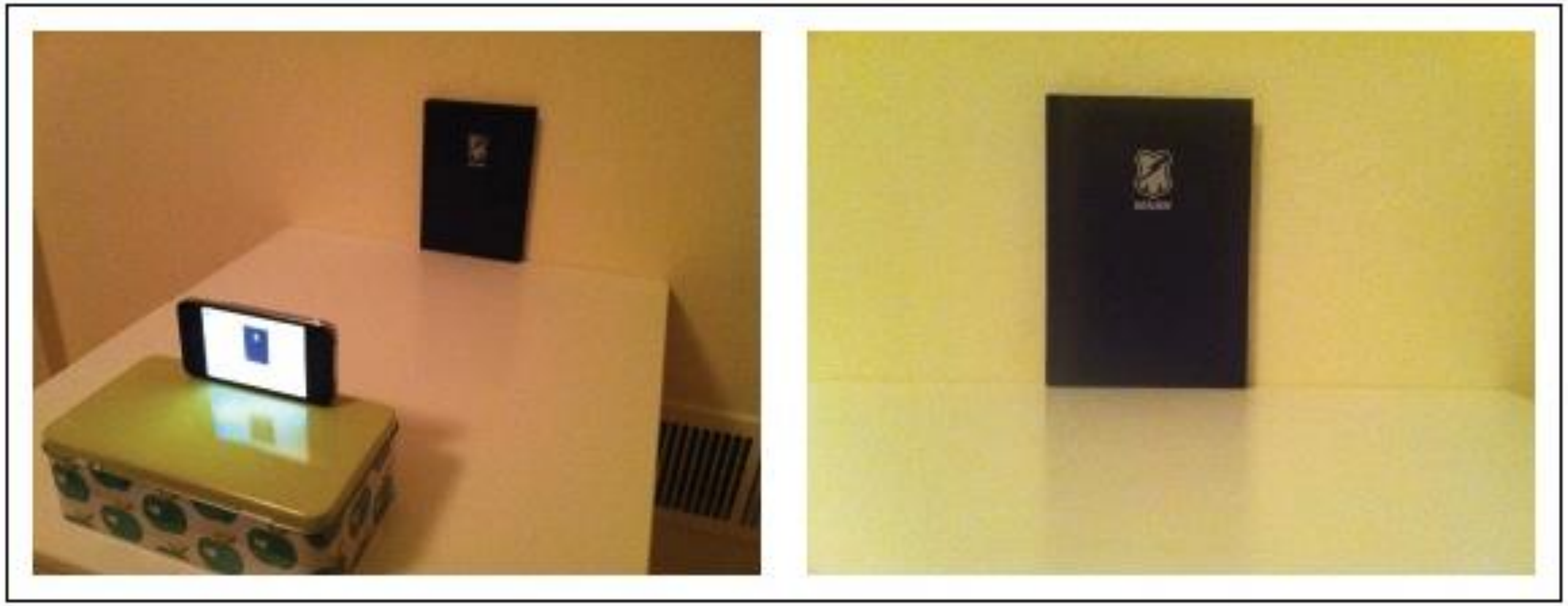
1. Intrinsic or Internal Parameters

It allows mapping between pixel coordinates and camera coordinates in the image frame.
E.g. optical center, focal length, and radial distortion coefficients of the lens.

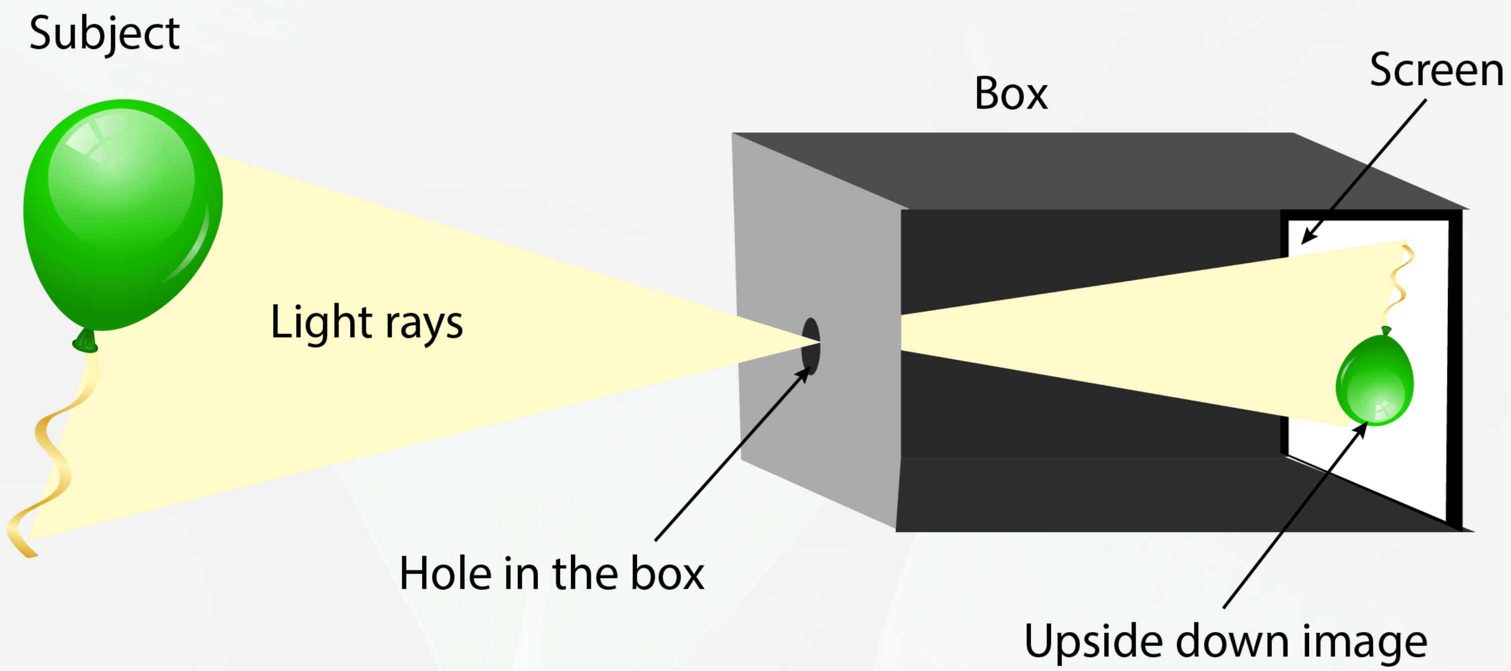
2. Extrinsic or External Parameters

It describes the orientation and location of the camera. This refers to the rotation and translation of the camera with respect to some world coordinate system.

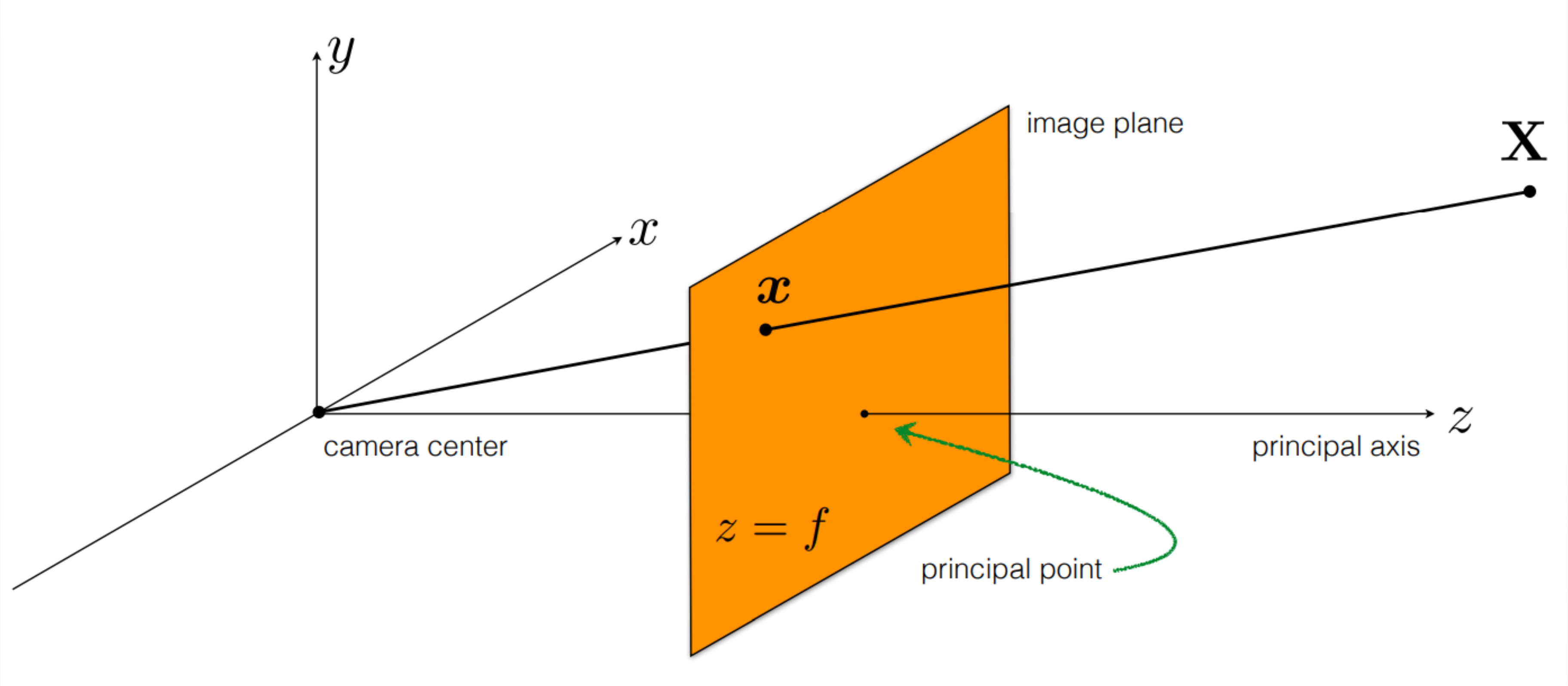
Camera Calibration



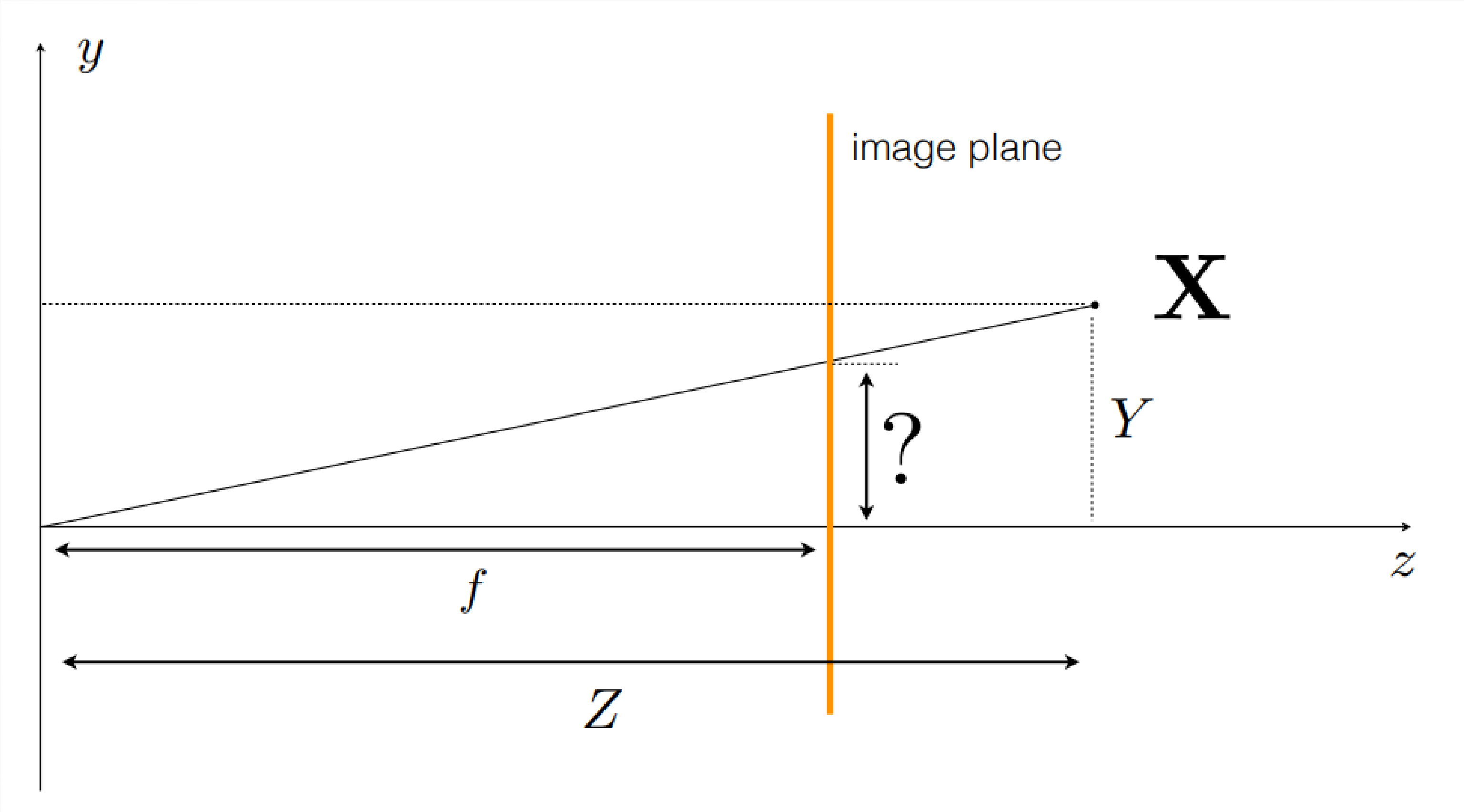
The pinhole camera



The pinhole camera and mapping



3D to 2D mapping



3D to 2D mapping

$$[X \quad Y \quad Z]^\top \mapsto [fX/Z \quad fY/Z]^\top$$

(X,Y,Z) are the coordinates of a 3D point in the camera coordinate system.

(x,y) are the corresponding 2D coordinates on the image plane.

f is the focal length of the camera, which determines how far the image plane is from the pinhole (optical center).

The division by Z captures perspective projection, meaning objects further from the camera appear smaller.

Camera matrix

A camera is a mapping between the 3D world and a 2D image.

$$\boldsymbol{x} = \mathbf{P}\mathbf{X}$$

2D image point | camera matrix | 3D world point

$$\boldsymbol{x} = \mathbf{P}\mathbf{X}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

homogeneous image 3 x 1 | Camera matrix 3 x 4 | homogeneous world point 4 x 1

Camera matrix

Camera matrix can be decomposed into two matrices

$$\mathbf{P} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc:c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right]$$

(3 x 3) (3 x 4)

$$\mathbf{P} = \mathbf{K}[\mathbf{I}|\mathbf{0}]$$

Camera matrix

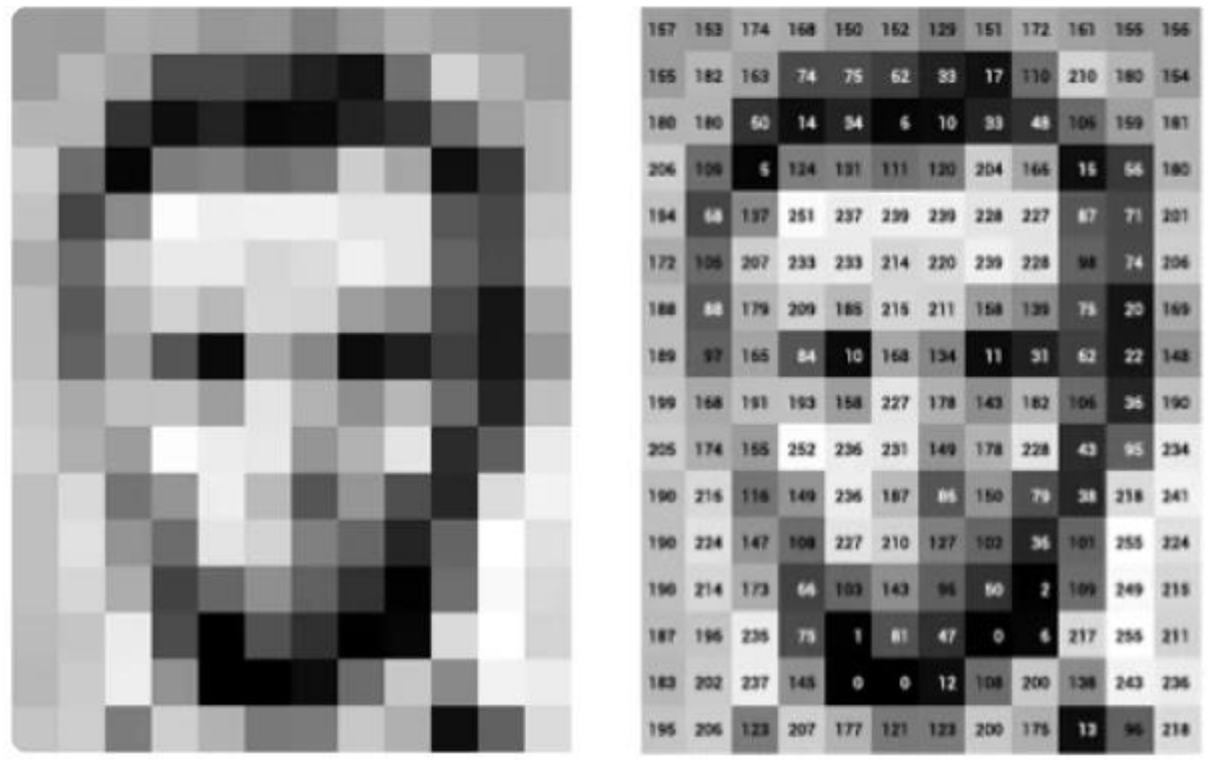
$$\mathbf{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

It contains intrinsic camera parameters such as focal length and principal point.

Focal length, f is the distance from the camera center to the image plane.

Principal point is the point where the optical axis (a straight line from the camera center perpendicular to the image plane) intersects the image plane.

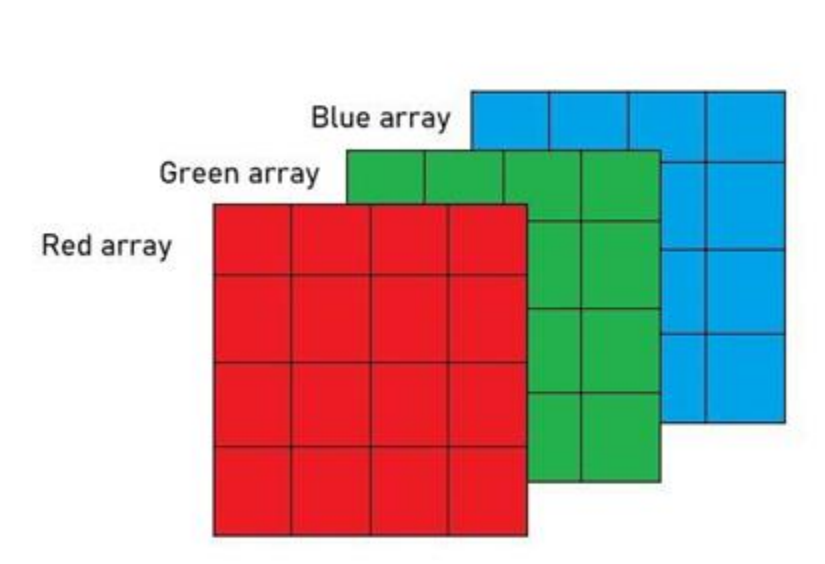
Image Representation

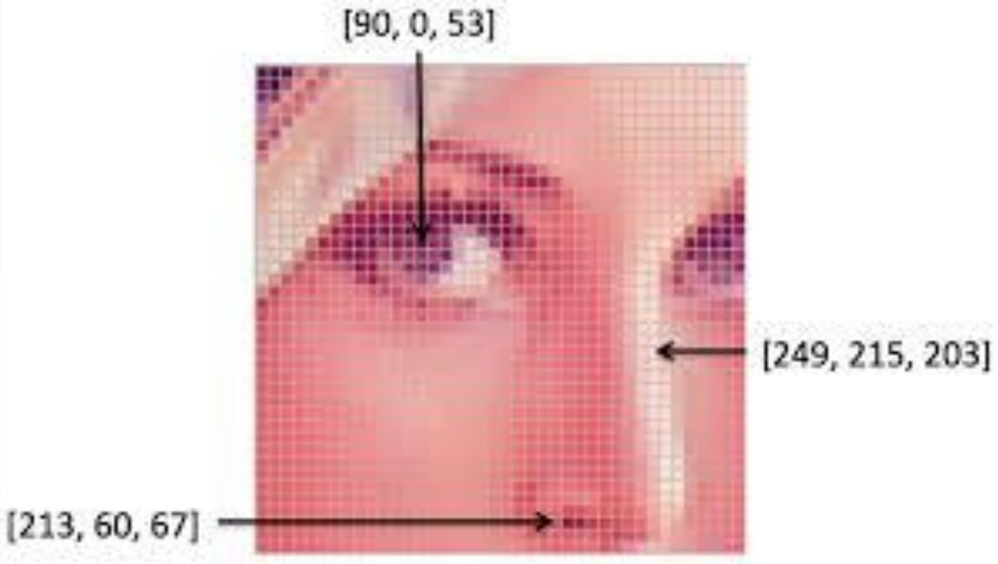


157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

RGB Representation

Arrays stacked over each other to form a Digital Image.

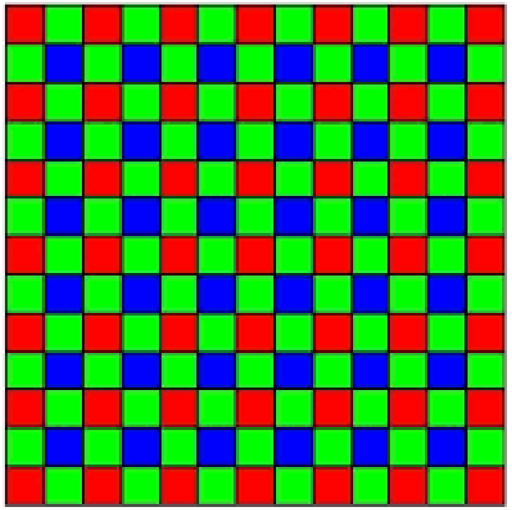




Bayer filter

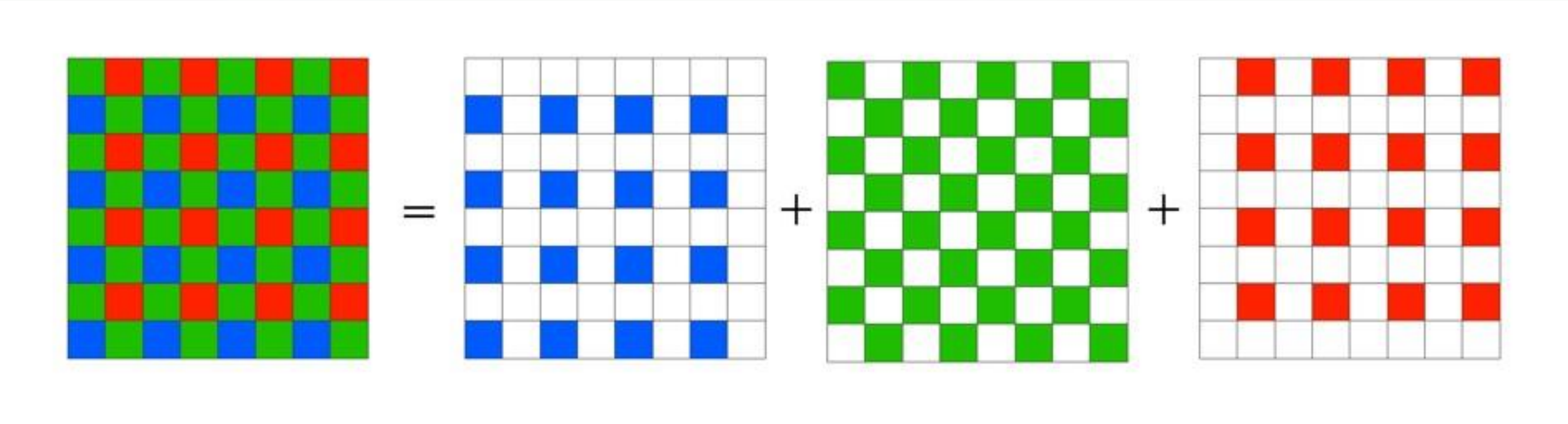
A digital camera separates Red (R), Green (G), and Blue (B) components using Bayer filter (a color filter array) placed over the image sensor.

Each pixel only captures **one** color (R, G, or B).



Bayer filter

Bayer filter

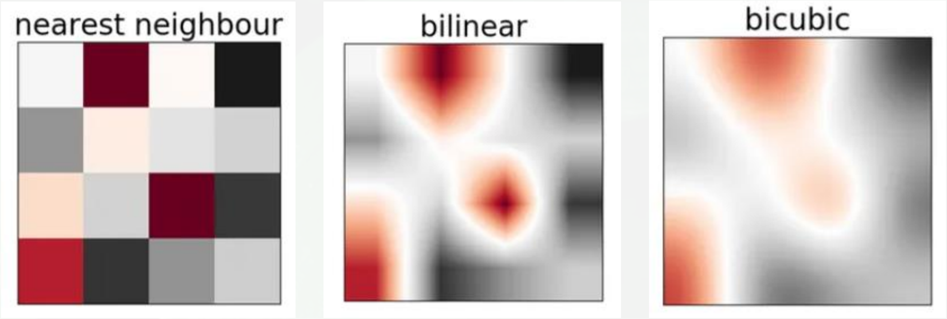


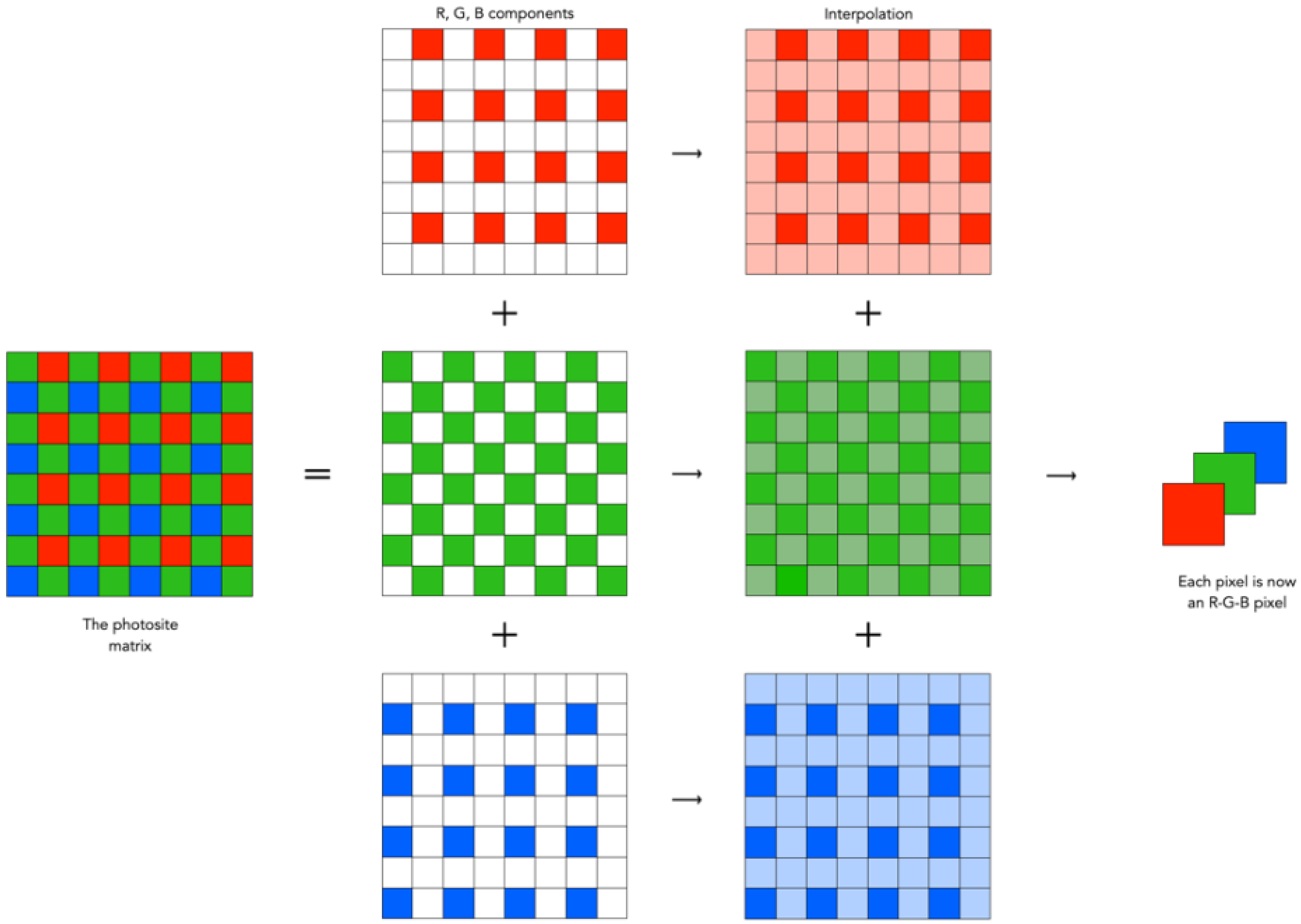
Interpolation

Interpolation is a mathematical technique used to estimate unknown values between known data points.

Types of interpolation:

- Nearest-neighbour interpolation:
- Bilinear interpolation:
- Bicubic interpolation:




R, G, B components
Interpolation
The photosite matrix
Each pixel is now an R-G-B pixel

Edge Detection

Edges: Edges are abrupt changes in intensity, discontinuity in image brightness or contrast; usually edges occur on the boundary of two regions.

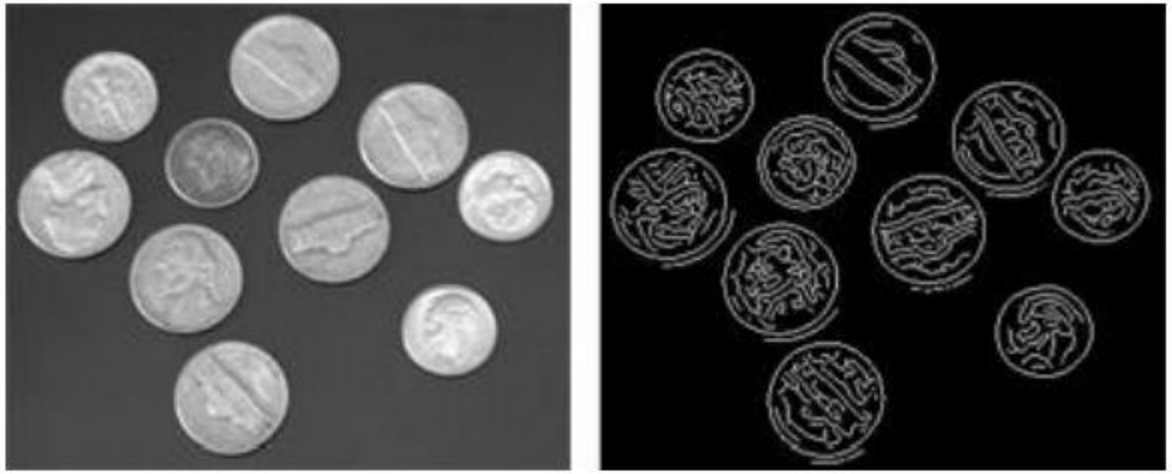


Figure: Original image (left) and edge (right)

Edge Detection

Edge detection: Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness.

Why we use edge detection?

- Reduce unnecessary information in the image while preserving the structure of the image.
- Extract important features of an image such as corners, lines, and curves.
- Edges provide strong visual clues that can help the recognition process.

Edge Detection

Here are some of the masks for edge detection.

- Prewitt Operator
- Sobel Operator
- Robinson Compass Masks
- Kirsch Compass Masks
- Laplacian Operator

Edge Detection

Prewitt Operator: By using Prewitt operator we can detects only horizontal and vertical edges.

-1	0	1
-1	0	1
-1	0	1

For Vertical Edges

-1	-1	-1
0	0	0
1	1	1

For Horizontal Edges

Edge Detection using Prewitt

Original Image	Applying Vertical Mask	Applying Horizontal Mask

Edge Detection

Sobel Operator: The sobel operator is very similar to Prewitt operator. Like Prewitt operator sobel operator is also used to detect two kinds of edges in an image:

-1	0	1
-2	0	2
-1	0	1

For Vertical Edges

-1	-2	-1
0	0	0
1	2	1

For Horizontal Edges

Edge Detection

Robinson Compass Masks: A Robinson compass mask is a type of mask which is used for edge detection. It has eight orientations. It is also known as the direction mask. It extracts the edges with respect to its direction.

Following are its eight orientations:

North, North West, West, South West, South, South East, East, North East

Robinson Compass Masks

0	1	2
-1	0	1
-2	-1	0

North West Mask

-1	0	1
-2	0	2
-1	0	1

North Mask

-2	-1	0
-1	0	1
0	1	2

North East Mask

1	2	1
0	0	0
-1	-2	-1

West Mask

-1	-2	-1
0	0	0
1	2	1

East Mask

2	1	0
1	0	-1
0	-1	-2

South West Mask

1	0	-1
2	0	-2
1	0	-1

South Mask

0	-1	-2
1	0	-1
2	1	0

South East Mask

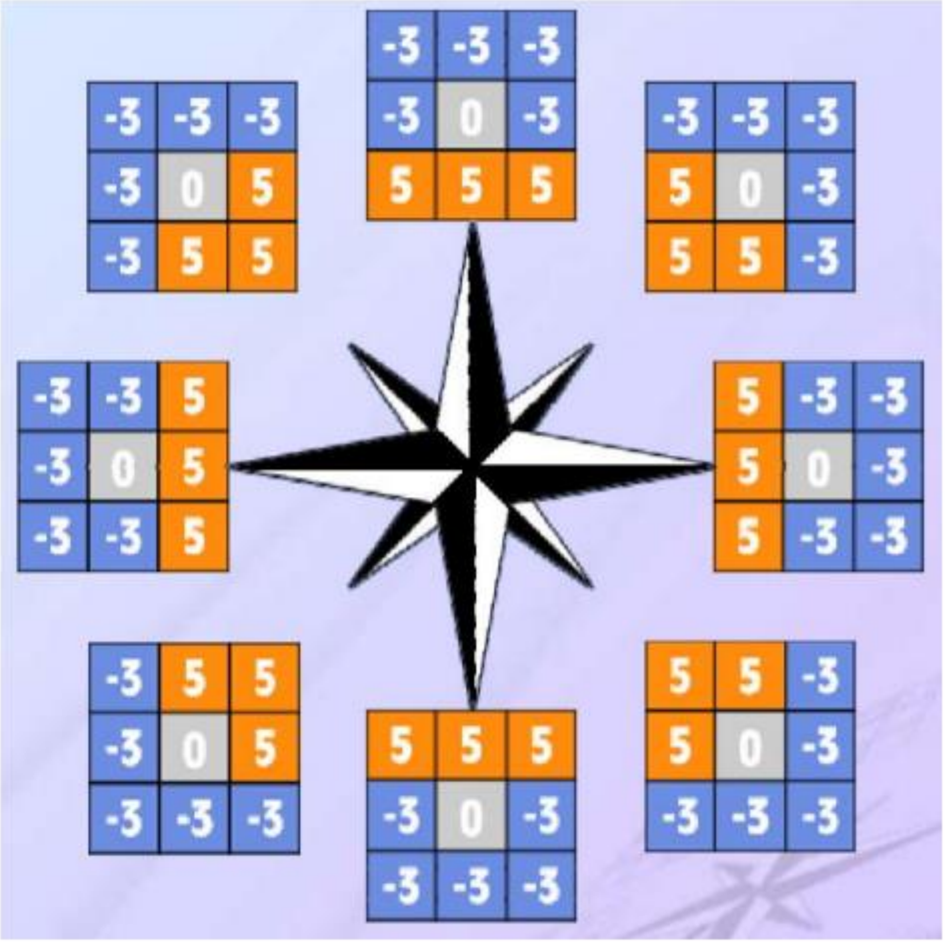
You can take any mask and you have to rotate it to find edges in all the above mentioned directions.

Edge Detection

Kirsch Compass Masks:

This is also like Robinson compass find edges in all the eight directions of a compass.

The only difference between Robinson and kirsch compass masks is that in Kirsch we have a standard mask but in Kirsch we change the mask according to our own requirements.


-3 -3 -3 / -3 0 5 / -3 5 5
-3 -3 -3 / -3 0 -3 / 5 5 5
-3 -3 -3 / 5 0 -3 / 5 5 -3
-3 -3 5 / -3 0 5 / -3 -3 5
5 -3 -3 / 5 0 -3 / 5 -3 -3
-3 5 5 / -3 0 5 / -3 -3 -3
5 5 5 / -3 0 -3 / -3 -3 -3
5 5 -3 / 5 0 -3 / -3 -3 -3

Edge Detection

Laplacian Operator:

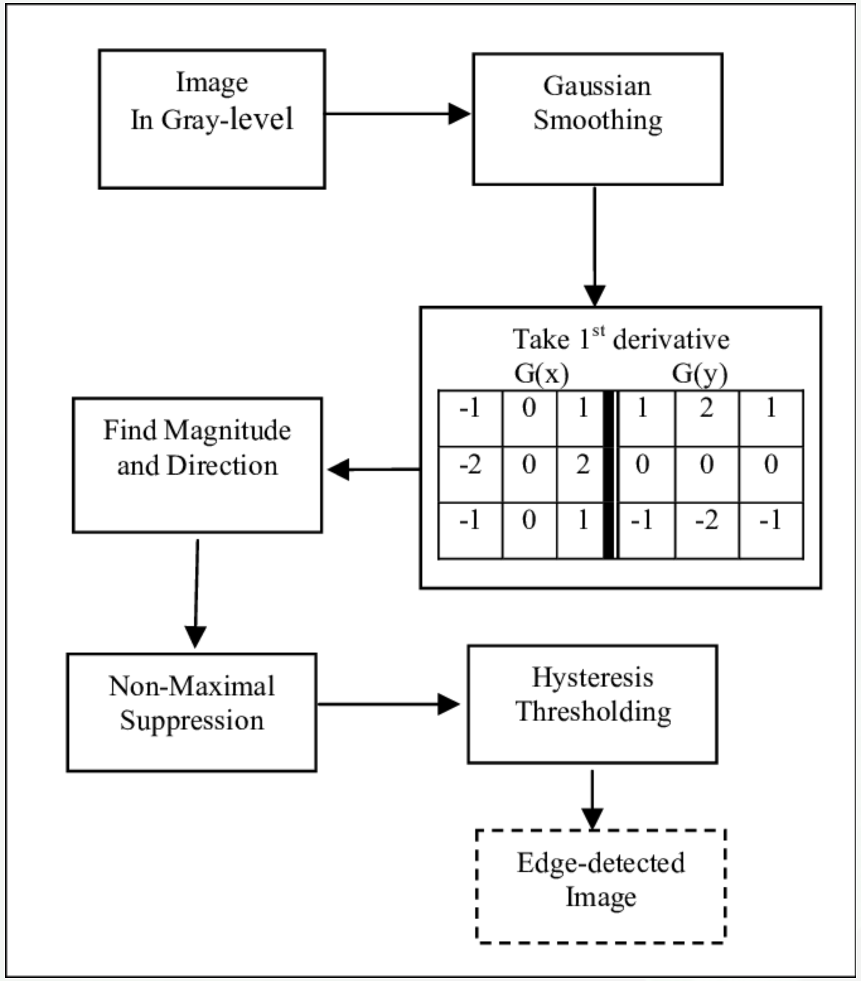
Laplacian is a second order derivative mask.

0	-1	0
-1	4	-1
0	-1	0

Original Image

Image After applying positive Laplacian operator

Image After applying Negative Laplacian operator



Canny's Edge Detection

The Canny edge detection algorithm is a multi-step process that detects edges in images.

It's a popular algorithm in computer vision and is known for its effectiveness in detecting edges with minimal false detection.

Canny's Edge Detection

<u>Gaussian smoothing</u>

Smoothing reduces noise and prevents false edges.

Gaussian kernel,

$$G(x, y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

(x, y) is the pixel coordinate

σ controls the amount of blur

Canny’s Edge Detection

1/16

1	2	1
2	4	2
1	2	1

3x3 Kernel

1/273

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

5x5 Kernel

1/1003

0	0	1	2	1	0	0
0	3	13	22	13	3	0
1	13	59	97	59	13	1
2	22	97	159	97	22	2
1	13	59	97	59	13	1
0	3	13	22	13	3	0
0	0	1	2	1	0	0

7x7 Kernel

Canny's Edge Detection

Finding gradients

Sobel finds the gradients in both horizontal and vertical direction. Since edges are perpendicular to the gradient direction, using these gradients we can find the edge gradient and direction for each pixel as:

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$

Magnitude

$$\mathbf{\Theta} = \operatorname{atan}\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

Direction

Canny's Edge Detection

<u>Non-maximum suppression</u>

This is an edge thinning technique. In this, for each pixel, we check if it is a local maximum in its neighborhood in the direction of gradient or not. If it is a local maximum it is retained as an edge pixel, otherwise suppressed.

For each pixel, the neighboring pixels are located in horizontal, vertical, and diagonal directions (0°, 45°, 90°, and 135°). Thus we need to round off the gradient direction at every pixel to one of these

After rounding, we will compare every pixel value against the two neighboring pixels in the gradient direction. If that pixel is a local maximum, it is retained as an edge pixel otherwise suppressed.

Canny’s Edge Detection

<u>Hysteresis Thresholding</u>

Canny uses the Hysteresis thresholding to solve the problem of “which edges are really edges and which are not” . Here, we set two thresholds T1 and T2.

Any edges with intensity greater than T2 are the sure edges.

Any edges with intensity less than T1 are sure to be non-edges.

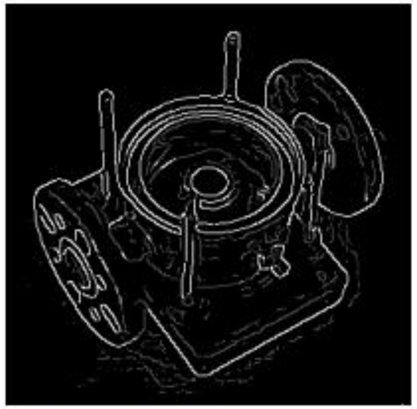
The edges between T1 and T2 are classified as edges only if they are connected to a sure edge otherwise discarded.

(a) Original

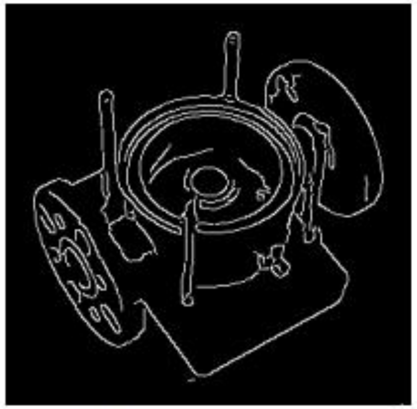
(b) Smoothed

(c) Gradient magnitudes

(d) Edges after non-maximum suppression



(e) Double thresholding



(f) Final output

Thank you