Mathematics in Image Processing — Complete Tutorial with Python

This tutorial explains **the math behind image processing** in a **clear, step-by-step way**, then shows **how to solve and implement it using Python libraries**.

It is designed for: - BS / MS students - Teachers - Final-year & AI/ML projects - Interview & viva preparation

---

1. Why Mathematics is Important in Image Processing

An image is **not a picture** for a computer — it is a **matrix of numbers**.

All image processing operations are based on: - Linear Algebra - Calculus - Probability & Statistics - Signal Processing - Geometry

Python libraries **hide the math**, but to build **correct systems**, you must understand it.

---

2. Image as a Matrix (Linear Algebra)

Mathematical Representation

- Grayscale image → 2D matrix
- Color image → 3D matrix (Height × Width × Channels)

$$I(x, y) \in [0, 255]$$

Python Implementation

```python
import cv2
import numpy as np

img = cv2.imread('image.jpg', 0)  # grayscale
print(img.shape)
print(img)
```

Each pixel is an **integer value**.

---

3. Matrix Operations on Images

(a) Image Addition

$$I = A + B$$

```
result = cv2.add(img, img)
```

(b) Image Subtraction

$$I = A - B$$

```
result = cv2.subtract(img, img)
```

(c) Scalar Multiplication (Brightness)

$$I' = k \times I$$

```
bright = cv2.convertScaleAbs(img, alpha=1.5, beta=0)
```

---

4. Image Normalization

Mathematical Formula

$$I_{norm} = \frac{I - I_{min}}{I_{max} - I_{min}}$$

Python Code

```
normalized = cv2.normalize(img, None, 0, 255, cv2.NORM_MINMAX)
```

---

5. Convolution (MOST IMPORTANT TOPIC)

Mathematical Formula

$$G(x, y) = \sum_i \sum_j I(x + i, y + j) \cdot K(i, j)$$

Example Kernel (Blur)

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1 1 & 1 & 1 1 & 1 & 1\end{bmatrix}$$

Python Implementation

```
kernel = np.ones((3,3), np.float32) / 9
blur = cv2.filter2D(img, -1, kernel)
```

---

6. Gaussian Filter (Statistics + Calculus)

Gaussian Formula

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

Python

```
gaussian = cv2.GaussianBlur(img, (5,5), 1.0)
```

---

7. Derivatives & Edge Detection (Calculus)

Edges are detected using **first derivative**.

Gradient

$$\nabla I = [\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}]$$

Sobel Operator

```
sobel_x = cv2.Sobel(img, cv2.CV_64F, 1, 0)
sobel_y = cv2.Sobel(img, cv2.CV_64F, 0, 1)
```

---

8. Gradient Magnitude

$$G = \sqrt{G_x^2 + G_y^2}$$

```
magnitude = np.sqrt(sobel_x**2 + sobel_y**2)
magnitude = np.uint8(magnitude)
```

---

9. Laplacian (Second Derivative)

Formula

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Python

```
laplacian = cv2.Laplacian(img, cv2.CV_64F)
```

---

10. Thresholding (Decision Theory)

Mathematical Rule

$$I(x,y) = \begin{cases}255 & I > T \ 0 & otherwise\end{cases}$$

Python

```
_, thresh = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)
```

---

11. Probability in Image Processing

Histogram (PDF)

$$p(r_k) = \frac{n_k}{N}$$

```
hist = cv2.calcHist([img],[0],None,[256],[0,256])
```

---

12. Histogram Equalization

Formula

$$s_k = (L-1)\sum_{j=0}^{k} p(r_j)$$

Python

```
equalized = cv2.equalizeHist(img)
```

---

13. Morphological Operations (Set Theory)

Dilation

$$A \oplus B$$

```
kernel = np.ones((5,5), np.uint8)
dilate = cv2.dilate(thresh, kernel)
```

Erosion

$$A \ominus B$$

```
erode = cv2.erode(thresh, kernel)
```

---

14. Geometric Transformations (Geometry)

Scaling Matrix

$$\begin{bmatrix} s_x & 0 & 0 0 & s_y & 0 \end{bmatrix}$$

Python

```
scaled = cv2.resize(img, None, fx=0.5, fy=0.5)
```

---

15. Rotation Matrix

$$\begin{bmatrix}\cos\theta & -\sin\theta\ \sin\theta & \cos\theta\end{bmatrix}$$

```
(h, w) = img.shape
M = cv2.getRotationMatrix2D((w//2, h//2), 45, 1)
rotated = cv2.warpAffine(img, M, (w, h))
```

---

16. Distance Metrics (Object Detection)

Euclidean Distance

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

```
from math import sqrt
d = sqrt((x1-x2)**2 + (y1-y2)**2)
```

---

17. Linear Algebra in Feature Detection

- Eigenvalues
- Eigenvectors
- SVD

Used in: - PCA - Harris Corner - SIFT

```
eigenvalues, eigenvectors = np.linalg.eig(img.astype(float))
```

---

18. Optimization (Canny Edge)

Canny uses: - Gradient - Non-maximum suppression - Double threshold - Hysteresis

(All math-based decisions)

```
edges = cv2.Canny(img, 100, 200)
```

---

19. How Python Libraries Hide the Math

Math Concept	OpenCV Function
Convolution	filter2D
Derivative	Sobel
Probability	calcHist
Geometry	warpAffine
Optimization	Canny

---

20. Practice Problems (Very Important)

1. Implement convolution **without OpenCV** using NumPy
2. Write Sobel filter manually
3. Compute histogram using loops
4. Implement thresholding from scratch
5. Rotate image using pure matrix math

---

21. Exam / Viva Focus Topics

- Convolution formula
- Sobel vs Laplacian
- Gaussian math
- Histogram equalization
- Morphology equations

---

22. Learning Path for You

Math → Image Processing → Computer Vision → CNN → Medical Imaging

Perfect for your **Bone Fracture Detection** and **AI Engineer goal**.

---

23. Next I Can Provide

✔️ Numerical solved problems (step-by-step) ✔️Manual math implementation (no OpenCV) ✔️MCQs & Viva questions ✔️University exam notes ✔️PDF / PPT slides

Just tell me ♀️