## ****. Image Processing Techniques****

Image processing involves manipulating images to improve quality, extract information, or apply effects. Common techniques include:

* **Filtering** (blur, sharpen, edge detection)
* **Thresholding** (convert to binary, e.g., black & white)
* **Morphological operations** (dilation, erosion, used in object detection)
* **Histogram equalization** (improves contrast)
* **Noise reduction** (remove unwanted grain/speckles)
* **Geometric transformations** (rotation, scaling, cropping, warping)
* **Color adjustments** (grayscale, brightness, saturation, hue changes)
* **Image compression** (reduce file size while preserving quality)

## ****2. Blur and Sharpening****

Both **blurring** and **sharpening** are done using **convolution filters (kernels)** applied to the image.

### 🔹 ****Blurring****

* Used to smooth an image, reduce noise, or create a soft effect.
* Achieved with **low-pass filters** (average or Gaussian filters).
* Example (3×3 average blur kernel):

This replaces each pixel with the average of its neighbors.

### 🔹 ****Sharpening****

* Enhances edges and fine details.
* Achieved with **high-pass filters**.
* Example sharpening kernel:

This increases contrast at edges.

✅ Libraries like **OpenCV** or **PIL** in Python make this very easy.

## ****3. Image Compression****

Compression reduces file size by encoding the image more efficiently.

### 🔹 ****Lossless compression****

* No data lost; image can be perfectly reconstructed.
* Formats: **PNG, BMP, TIFF**.
* Uses algorithms like **Run-Length Encoding (RLE)**, **Huffman coding**, **LZW**.

### 🔹 ****Lossy compression****

* Some information discarded to save space.
* Format: **JPEG, HEIC, WebP**.
* JPEG works by:
  1. Converting image to frequency space (DCT – Discrete Cosine Transform).
  2. Removing less important frequencies (details human eyes don’t notice).
  3. Quantizing and compressing coefficients.

HEIC (used in iPhones) uses a more modern codec (HEVC/H.265), offering **better compression than JPEG** for the same quality.

## ****4. Image Format Conversion (JPG → PNG, HEIC → PNG, etc.)****

* When you convert between formats, you’re **re-encoding the image** using a different compression algorithm.
* Example:
  + **JPG → PNG**: Expands the compressed data and saves it losslessly (file may get bigger).
  + **PNG → JPG**: Applies lossy compression, smaller size but some quality loss.
  + **HEIC → PNG**: Decodes HEIC (using HEVC codec), then encodes into PNG.

✅ Tools: OpenCV, PIL (Python), ImageMagick, Photoshop, even built-in OS converters.

# Image Processing Example with Python (OpenCV)

This example demonstrates how to:

1. Apply blurring techniques (Average, Gaussian)

2. Apply sharpening using a convolution kernel

3. Convert image formats (JPG, PNG, HEIC)

## Python Code Example:

import cv2  
import numpy as np  
  
# Load an image (change path to your image file)  
img = cv2.imread("input.jpg")  
  
# ---------- 1. BLURRING ----------  
# Average Blur  
blur\_avg = cv2.blur(img, (5, 5))  
  
# Gaussian Blur  
blur\_gaussian = cv2.GaussianBlur(img, (5, 5), 0)  
  
# ---------- 2. SHARPENING ----------  
# Define sharpening kernel  
sharpen\_kernel = np.array([[0, -1, 0],  
 [-1, 5, -1],  
 [0, -1, 0]])  
sharpened = cv2.filter2D(img, -1, sharpen\_kernel)  
  
# ---------- 3. FORMAT CONVERSION ----------  
# Save as PNG (lossless)  
cv2.imwrite("output.png", img)  
  
# Save as JPG (lossy)  
cv2.imwrite("output.jpg", img, [cv2.IMWRITE\_JPEG\_QUALITY, 90])  
  
# If you have HEIC images, you need an external library like pyheif to decode  
# Example (requires `pip install pyheif pillow`):  
# import pyheif  
# from PIL import Image  
# heif\_file = pyheif.read("input.heic")  
# img\_pil = Image.frombytes(  
# heif\_file.mode, heif\_file.size, heif\_file.data,  
# "raw", heif\_file.mode, heif\_file.stride  
# )  
# img\_pil.save("output.png") # Save as PNG  
  
# ---------- 4. SHOW RESULTS ----------  
cv2.imshow("Original", img)  
cv2.imshow("Blur - Average", blur\_avg)  
cv2.imshow("Blur - Gaussian", blur\_gaussian)  
cv2.imshow("Sharpened", sharpened)  
  
cv2.waitKey(0)  
cv2.destroyAllWindows()