AI385

Lab 3

Image formation with OpenCV

Learning Objectives:

- *Access and manipulate individual pixel values.*
- *Resize, crop, rotate, and flip images.*
- *Convert images into different color spaces.*
- *Extract and visualize individual color channels.*

Introduction:

Image formation is a fundamental concept in computer vision and digital imaging. It describes how images are captured, processed, and represented in a computer. For machine learning and deep learning applications, understanding image formation is essential for building robust models used in ***autonomous driving, medical imaging, face recognition, and satellite image analysis****.*

Exercise 1: Accessing and Modifying Pixel Values

Application: In deep learning, accessing and modifying pixels is useful in preprocessing images for segmentation, anomaly detection, and image enhancement.

```
import cv2
import numpy as np

# Load an image
img = cv2.imread('image.jpg')
# Get pixel value at (100,100)
pixel = img[100, 100]
print("Original Pixel Value at (100,100):", pixel)

# Modify pixel value at (100,100)
img[100, 100] = [255, 0, 0]  # Change to blue
cv2.imshow('Modified Image', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Exercise 2: Image Transformations (Rotating, Cropping, Resizing, Flipping)

Application: Image transformations help in data augmentation for training robust machine learning models.

```
# Load an image
img = cv2.imread('image.jpg')
# Rotate the image by 45 degrees
(h, w) = img.shape[:2]
center = (w // 2, h // 2)
M = cv2.getRotationMatrix2D(center, 45, 1.0)
rotated_img = cv2.warpAffine(img, M, (w, h))
cv2.imshow('Rotated Image', rotated_img)
cv2.waitKey(0)
# Cropping an image
cropped_img = img[50:300, 50:300]  # Crop a region from (50,50) to (300,300)
cv2.imshow('Cropped Image', cropped_img)
cv2.waitKey(0)
# Resizing an image
resized_img = cv2.resize(img, (300, 300))
cv2.imshow('Resized Image', resized_img)
cv2.waitKey(0)
# Flipping an image
flipped_img = cv2.flip(img, 1)  # Flip horizontally
cv2.imshow('Flipped Image', flipped_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Exercise 3: Working with Color Spaces

Application: Different color spaces are used for object tracking, segmentation, and classification.

a) Convert to different color spaces (HSV, LAB)

```
import cv2
import numpy as np

# Load an image
img = cv2.imread('image.jpg')
# Convert into different color space
hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
lab_img = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
cv2.imshow('HSV Image', hsv_img)
cv2.imshow('LAB Image', lab_img)
cv2.waitKey(0)
```

b) Display each color channel separately

```
# Display indivisual Color channels# Split color channels
b, g, r = cv2.split(img)
cv2.imshow('Red Channel', r)
cv2.imshow('Green Channel', g)
cv2.imshow('Blue Channel', b)
cv2.waitKey(0)
```

Bonus Task: Using AI Assistance for Video Color Space Processing

Objective:

Simulate a real-world scenario where you want to process a video in different color spaces but need AI assistance to write and refine the code. Your goal is to:

- Prompt AI to generate code that captures video from a file or webcam.
- Convert each frame to different color spaces (Grayscale, HSV, LAB).
- Display the video stream in each color space.
- Stop the video stream when the 'q' key is pressed.

Expected Outcome:

A program that successfully:

✅ Opens a video file or webcam stream.
✅ Converts each frame into **Grayscale, HSV, and LAB** color spaces.
✅ Displays three live video feeds side by side.
✅ Stops execution upon pressing **'q'**.

References

- OpenCV: Basic Image Processing
- OpenCV: Image Transformations
- OpenCV: Color Spaces