📘 OpenCV Notes – Chapter 1: Reading Images, Videos, and Webcam

---

🔹 What is OpenCV?

OpenCV (Open Source Computer Vision Library) is a Python library used for real-time image processing and computer vision tasks like object detection, face recognition, and more.

🖼️ 1. Reading and Displaying an Image

✅ Concept:

• cv2.imread() is used to load an image
• cv2.imshow() displays the image in a window
• cv2.waitKey() keeps the window open for a specified time
• cv2.destroyAllWindows() closes all windows (used when needed)

🧠 Notes:

• cv2.waitKey(0) waits indefinitely until a key is pressed.
• It's useful to view static images.

✅ Code:

```
import cv2

print("Package imported")


# Load image from file

img = cv2.imread("Resources/lena.png")  # Provide correct path


# Show image in a window

cv2.imshow("Output", img)


# Wait until any key is pressed

cv2.waitKey(0)
```

🎥 2. Reading and Displaying a Video

✅ **Concept:**

- A video is a sequence of images (frames).
- cv2.VideoCapture() is used to read the video file.
- cap.read() reads one frame at a time.
- Use a while loop to display frames continuously.

🧠 **Notes:**

- success, img = cap.read() returns two values:
  - o success: True if a frame is read successfully.
  - o img: the frame image.
- cv2.waitKey(1) waits for 1 millisecond between frames.
- Press 'q' to quit the loop.

✅ **Code:**

```
import cv2


# Load video from file

cap = cv2.VideoCapture("Resources/test_video.mp4")


# Loop through each frame

while True:

    success, img = cap.read()  # Read the current frame

    cv2.imshow("Video", img)   # Display the frame


    # Exit loop when 'q' is pressed

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break
```

📸 3. Reading from a Webcam

✅ **Concept:**

- Use cv2.VideoCapture(0) to access the default webcam.
- Just like videos, webcam feeds are also a series of images.
- We can set webcam properties using cap.set().

🧠 **Notes:**

- cap.set(3, 640): Set width to 640 pixels.
- cap.set(4, 480): Set height to 480 pixels.
- cap.set(10, 100): Set brightness level.

✅ Code:

```
import cv2


# Start webcam capture (0 means default camera)

cap = cv2.VideoCapture(0)


# Set webcam properties

cap.set(3, 640)  # Width

cap.set(4, 480)  # Height

cap.set(10, 100) # Brightness


# Read and show frames continuously

while True:

    success, img = cap.read()

    cv2.imshow("Video", img)


    # Exit loop when 'q' is pressed

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

✅ Summary:

Function	Purpose
cv2.imread()	Read an image
cv2.imshow()	Display an image or frame
cv2.waitKey()	Pause the program for given ms or until key
cv2.VideoCapture()	Capture video from file or webcam
cap.read()	Read a single frame from video/webcam
cap.set(prop_id, val)	Set camera properties like width/height

---

📘 OpenCV Notes – Chapter 2: Basic Image Processing Functions

🛠️ Basic Functions in OpenCV

✅ Concepts:

• Convert to Grayscale
• Apply Gaussian Blur
• Edge Detection using Canny
• Dilation and Erosion

✅ Key Functions:

• cv2.cvtColor(): Convert color spaces (e.g., BGR to Grayscale)
• cv2.GaussianBlur(): Apply blur to reduce noise
• cv2.Canny(): Detect edges in the image
• cv2.dilate(): Thicken edges
• cv2.erode(): Thin edges

✅ Code:

```
import cv2

import numpy as np
```

```
img = cv2.imread("Resources/lena.png")
kernel = np.ones((5,5), np.uint8)  # Used for dilation/erosion

imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
imgBlur = cv2.GaussianBlur(imgGray, (7,7), 0)
imgCanny = cv2.Canny(img, 150, 200)
imgDilation = cv2.dilate(imgCanny, kernel, iterations=1)
imgEroded = cv2.erode(imgDilation, kernel, iterations=1)

cv2.imshow("Gray Image", imgGray)
cv2.imshow("Blur Image", imgBlur)
cv2.imshow("Canny Image", imgCanny)
cv2.imshow("Dilation Image", imgDilation)
cv2.imshow("Eroded Image", imgEroded)
cv2.waitKey(0)
```

📘 OpenCV Notes – Chapter 3: Resizing and Cropping

✅ Concepts:

- Resize an image using cv2.resize()
- Crop an image using slicing: img[y1:y2, x1:x2]

✅ Code:

```
import cv2
import numpy as np

img = cv2.imread("Resources/lambo.png")
```

```
print(img.shape)  # (height, width, channels)

imgResize = cv2.resize(img, (300, 200))  # Resize to 300x200
print(imgResize.shape)

imgCropped = img[0:200, 200:500]  # Crop height 0–200, width 200–500

cv2.imshow("Original", img)
cv2.imshow("Resized", imgResize)
cv2.imshow("Cropped", imgCropped)
cv2.waitKey(0)
```

📘 OpenCV Notes – Chapter 4: Drawing Shapes and Text

✅ Concepts:

• Draw basic shapes and add text using OpenCV functions.
• Create a blank image using NumPy.

✅ Key Functions:

• cv2.line() – Draw a straight line.
• cv2.rectangle() – Draw a rectangle.
• cv2.circle() – Draw a circle.
• cv2.putText() – Add text to an image.

✅ Code:

```
import cv2
import numpy as np

# Create blank image (Black background)
img = np.zeros((512, 512, 3), np.uint8)
```

```
# Fill background with blue color
img[:] = 255, 0, 0

# Draw a green diagonal line from top-left to bottom-right
cv2.line(img, (0, 0), (img.shape[1], img.shape[0]), (0, 255, 0), 3)
# (0, 0) – Start point (top-left)
# (img.shape[1], img.shape[0]) – End point (bottom-right)
# (0, 255, 0) – Green color in BGR
# 3 – Thickness in pixels

# Draw a red rectangle from top-left to (250, 350)
cv2.rectangle(img, (0, 0), (250, 350), (0, 0, 255), 2)
# (0, 0) – Top-left corner
# (250, 350) – Bottom-right corner
# (0, 0, 255) – Red color
# 2 – Border thickness

# Draw a yellow circle at (400, 50) with radius 30
cv2.circle(img, (400, 50), 30, (255, 255, 0), 5)
# (400, 50) – Center of circle
# 30 – Radius
# (255, 255, 0) – Yellow color
# 5 – Thickness of circle outline

# Write "OpenCV" text at position (300, 200)
cv2.putText(img, "OpenCV", (300, 200), cv2.FONT_HERSHEY_COMPLEX,
1, (0, 150, 0), 3)
# (300, 200) – Bottom-left corner of text
```

# FONT_HERSHEY_COMPLEX – Font type

# 1 – Font scale

# (0, 150, 0) – Greenish color

# 3 – Thickness

cv2.imshow("Shapes", img)

cv2.waitKey(0)

✅ Summary of Drawing Functions:

Function	Description
cv2.line(img, pt1, pt2, color, thickness)	Draw line from pt1 to pt2
cv2.rectangle(img, pt1, pt2, color, thickness)	Draw rectangle with corners pt1 & pt2
cv2.circle(img, center, radius, color, thickness)	Draw circle with given center and radius
cv2.putText(img, text, org, font, scale, color, thickness)	Put text on the image

📘 OpenCV Notes – Chapter 5: Perspective Warp

🔸 What is Perspective Transformation?

A perspective transform allows you to view an object from a different angle, correcting distortions. For example, when a card or page is viewed from an angle, it appears distorted. Perspective warp adjusts the image so it looks like a top-down view.

✅ Key Functions:

- cv2.getPerspectiveTransform(pts1, pts2)
  - Maps 4 points in the original image (pts1) to 4 points in the desired output (pts2).
- cv2.warpPerspective(img, matrix, (width, height))

  - Applies the transformation using the matrix and outputs the warped image.

✅ **Code:**

```
import cv2
import numpy as np

img = cv2.imread("Resources/cards.jpg")

width, height = 250, 350

# Points from the original image (corners of the card)
pts1 = np.float32([[111, 219], [287, 188], [154, 482], [352, 440]])
# Corresponding points in the new image
pts2 = np.float32([[0, 0], [width, 0], [0, height], [width, height]])

# Perspective transformation matrix
matrix = cv2.getPerspectiveTransform(pts1, pts2)
imgOutput = cv2.warpPerspective(img, matrix, (width, height))

cv2.imshow("Image", img)
cv2.imshow("Output", imgOutput)
cv2.waitKey(0)
```

📘 OpenCV Notes – Chapter 6 : Joining Images

🔸 **Problem:**

- Different image sizes and channels can break stacking.

✅ **Solution:**

- Create a utility function stackImages() to handle resizing and color conversion.
- Ensures all images fit into a stackable format.

✅ **Code:**

```
import cv2
import numpy as np


def stackImages(scale, imgArray):
    rows = len(imgArray)
    cols = len(imgArray[0])
    rowsAvailable = isinstance(imgArray[0], list)
    width = imgArray[0][0].shape[1]
    height = imgArray[0][0].shape[0]
    if rowsAvailable:
        for x in range(rows):
            for y in range(cols):
                if imgArray[x][y].shape[:2] == imgArray[0][0].shape[:2]:
                    imgArray[x][y] = cv2.resize(imgArray[x][y], (0, 0), None, scale,
scale)
                else:
                    imgArray[x][y] = cv2.resize(imgArray[x][y], (width, height), None,
scale, scale)
                if len(imgArray[x][y].shape) == 2:
                    imgArray[x][y] = cv2.cvtColor(imgArray[x][y],
cv2.COLOR_GRAY2BGR)
        hor = [np.hstack(row) for row in imgArray]
        ver = np.vstack(hor)
    else:
        for x in range(rows):
            if imgArray[x].shape[:2] == imgArray[0].shape[:2]:
                imgArray[x] = cv2.resize(imgArray[x], (0, 0), None, scale, scale)
            else:
                imgArray[x] = cv2.resize(imgArray[x], (width, height), None, scale,
scale)
            if len(imgArray[x].shape) == 2:
                imgArray[x] = cv2.cvtColor(imgArray[x],
cv2.COLOR_GRAY2BGR)
        ver = np.hstack(imgArray)
    return ver


img = cv2.imread('Resources/lena.png')
imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
imgStack = stackImages(0.5, ([img, imgGray, img], [img, img, img]))


cv2.imshow("ImageStack", imgStack)
cv2.waitKey(0)
```

📘 OpenCV Notes – Chapter 7: Colour Detection with HSV

🔸 Why HSV?

HSV (Hue, Saturation, Value) is better for color filtering than BGR. It separates image intensity (brightness) from color information.

✅ Tools Used:

- cv2.cvtColor(img, cv2.COLOR_BGR2HSV): Convert BGR image to HSV

- cv2.inRange(imgHSV, lower, upper): Create mask for color range
- cv2.bitwise_and(img, img, mask=mask): Apply mask to keep only the target color
- cv2.createTrackbar(...): Create GUI sliders to tune HSV values live

✅ **Code:**

```
import cv2
import numpy as np


def empty(a):
    pass


cv2.namedWindow("TrackBars")
cv2.resizeWindow("TrackBars", 640, 240)
cv2.createTrackbar("Hue Min", "TrackBars", 0, 179, empty)
cv2.createTrackbar("Hue Max", "TrackBars", 19, 179, empty)
cv2.createTrackbar("Sat Min", "TrackBars", 110, 255, empty)
cv2.createTrackbar("Sat Max", "TrackBars", 240, 255, empty)
cv2.createTrackbar("Val Min", "TrackBars", 153, 255, empty)
cv2.createTrackbar("Val Max", "TrackBars", 255, 255, empty)


while True:
    img = cv2.imread('Resources/lambo.png')
    imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)


    h_min = cv2.getTrackbarPos("Hue Min", "TrackBars")
    h_max = cv2.getTrackbarPos("Hue Max", "TrackBars")
    s_min = cv2.getTrackbarPos("Sat Min", "TrackBars")
    s_max = cv2.getTrackbarPos("Sat Max", "TrackBars")
    v_min = cv2.getTrackbarPos("Val Min", "TrackBars")
    v_max = cv2.getTrackbarPos("Val Max", "TrackBars")

    lower = np.array([h_min, s_min, v_min])
    upper = np.array([h_max, s_max, v_max])
    mask = cv2.inRange(imgHSV, lower, upper)
    imgResult = cv2.bitwise_and(img, img, mask=mask)

    imgStack = stackImages(0.6, ([img, imgHSV], [mask, imgResult]))
    cv2.imshow("Stacked Images", imgStack)
    cv2.waitKey(1)
```

✅ **Summary:**

Function	**Purpose**
cv2.getPerspectiveTransform()	Computes transformation matrix from 4 source to 4 destination points
cv2.warpPerspective()	Applies the transformation to warp the image
cv2.cvtColor()	Convert image to different color space (like HSV)
cv2.inRange()	Create a mask for a specific color range
cv2.bitwise_and()	Apply mask to extract the filtered part
cv2.createTrackbar()	Creates interactive sliders for tuning
cv2.resize()	Resizes image to desired size
np.hstack, np.vstack	Joins multiple images horizontally or vertically

📘 OpenCV Notes – Chapter 8: Contour/Shape Detection

📊 **Concept**:

- Contours are useful for shape analysis and object detection.
- A contour is a curve joining all continuous points along a boundary with the same color or intensity.
- Steps:
    1. Convert to grayscale.
    2. Apply Gaussian Blur to reduce noise.
    3. Use cv2.Canny() for edge detection.
    4. Use cv2.findContours() to detect contours.
    5. Optionally use cv2.drawContours() to draw contours.

🧠 **Functions & Arguments Explained**:

- cv2.findContours(image, mode, method):
    - RETR_EXTERNAL: Retrieves only the extreme outer contours.
    - CHAIN_APPROX_NONE: Stores all the contour points.
- cv2.contourArea(cnt): Calculates area of a contour.
- cv2.arcLength(cnt, True): Finds perimeter.
- cv2.approxPolyDP(cnt, 0.02*peri, True): Approximates shape of contour.
- cv2.boundingRect(approx): Returns (x, y, width, height) of bounding box.
- cv2.putText(img, text, org, font, scale, color, thickness):
    - org = (x+(w//2)-10,y+(h//2)-10) places label at center.

📝 **Shape Classification**:

- 3 Corners ➔ Triangle
- 4 Corners + Aspect Ratio ➔ Square or Rectangle
- 4 Corners ➔ Circle

✅ **Code Summary**:

# Image preprocessing

imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

```
imgBlur = cv2.GaussianBlur(imgGray,(7,7),1)
imgCanny = cv2.Canny(imgBlur,50,50)


# Get contours and classify shapes
getContours(imgCanny)
```

🌐 **Final Visualization**:

- Uses stackImages() to compare stages: original, gray, blur, canny, contour-drawn, blank

📘 OpenCV Notes – Chapter 9: Face Detection

🕵️ **Concept**:

- OpenCV provides pre-trained Haar Cascade classifiers for face, eye, full-body detection, etc.
- Uses .xml files to identify features of faces.

🧠 **Functions & Arguments Explained**:

- cv2.CascadeClassifier(path): Loads the Haar Cascade classifier from file.
- detectMultiScale(image, scaleFactor, minNeighbors):
    - scaleFactor: How much the image size is reduced at each scale.
    - minNeighbors: How many neighbors each rectangle should have to retain it.
- cv2.rectangle(img, pt1, pt2, color, thickness):
    - pt1: Top-left (x, y)
    - pt2: Bottom-right (x+w, y+h)
    - color: (255, 0, 0) for blue
    - thickness: Border thickness of rectangle

✅ **Code Summary**:

```
faceCascade =
cv2.CascadeClassifier("Resources/haarcascade_frontalface_default.xml")
```

```
imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = faceCascade.detectMultiScale(imgGray, 1.1, 4)


for (x,y,w,h) in faces:
   cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
```

🚀 **Uses**:

- Can be used in real-time video feed as well
- Also extendable to eye, smile, or license plate detection using different .xml files

---