Chapter 21: OpenCV

Introduction

In the world of Artificial Intelligence, visual understanding is a powerful capability. Computers are now being taught to "see" and "interpret" the world around them, just like humans. This area of AI is called **Computer Vision**. One of the most widely used libraries for computer vision is **OpenCV** – Open Source Computer Vision Library.

In this chapter, you will explore the basics of OpenCV, understand how images are processed in a computer, and learn how AI applications like face detection, object tracking, and gesture recognition are built using OpenCV.

21.1 What is OpenCV?

OpenCV stands for **Open Source Computer Vision Library**. It is a **Python-compatible** and **C++-based** open-source library that provides tools to:

- Read and write images
- Perform image processing
- Detect faces and objects
- Analyze visual data in real-time

It is widely used in:

- Robotics
- Surveillance systems
- Augmented Reality (AR)
- Healthcare diagnostics (e.g., detecting X-ray abnormalities)

21.2 Installing OpenCV in Python

To use OpenCV in Python, it must be installed using **pip**:

```
pip install opencv-python
```

Once installed, it is imported using:

```
import cv2
```

21.3 Working with Images

21.3.1 Reading an Image

To read an image:

```
import cv2
image = cv2.imread('example.jpg')
cv2.imshow('Display Image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- `imread()` – loads the image
- `imshow()` – displays it in a window
- `waitKey(0)` – waits for a key press
- `destroyAllWindows()` – closes the window

21.3.2 Image as a Matrix

Images are stored as **matrices of pixels**:

- **Grayscale images** → 2D arrays (Height × Width)
- **Color images** → 3D arrays (Height × Width × 3 channels - BGR)

21.4 Image Processing with OpenCV

OpenCV allows many image manipulation techniques:

21.4.1 Converting to Grayscale

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

21.4.2 Resizing an Image

```
resized = cv2.resize(image, (300, 200))
```

21.4.3 Blurring an Image

```
blurred = cv2.GaussianBlur(image, (5, 5), 0)
```

21.4.4 Drawing on Images

```
cv2.rectangle(image, (50, 50), (200, 200), (0, 255, 0), 2)
cv2.circle(image, (150, 150), 50, (255, 0, 0), 3)
```

21.5 Face Detection Using OpenCV

OpenCV comes with a **pre-trained face detection model** using Haar Cascades.

21.5.1 Load Haar Cascade Classifier

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

21.5.2 Detect Faces in an Image

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

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

21.5.3 Display Detected Faces

```
cv2.imshow('Detected Faces', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

21.6 Using Webcam with OpenCV

To capture live video from a webcam:

```
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    cv2.imshow('Webcam Feed', frame)

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

cap.release()
cv2.destroyAllWindows()
```

This is used in:

- Real-time face detection
- Gesture recognition
- AI-powered video applications

21.7 Applications of OpenCV

Field	Application Example
Healthcare	Detecting tumors in scans
Automotive	Lane detection in self-driving cars
Retail	Customer movement tracking

Field	Application Example
Security	Intruder detection using CCTV
Education	Gesture-based interaction systems

Summary

OpenCV is a powerful and widely used open-source library for image and video processing. It forms the foundation of computer vision applications in AI. In this chapter, you learned to:

- Install and use OpenCV in Python
- Read, display, and manipulate images
- Convert color images to grayscale and apply filters
- Draw shapes and text on images
- Detect faces in static images
- Capture and process video in real-time

This knowledge is the first step toward building intelligent systems that can understand and interact with the visual world—just like humans do.
