Computer Vision in Python - Essentials

1) What is Computer Vision?

Computer Vision lets computers understand images and video so we can take actions automatically. Typical results: labels, bounding boxes, masks, keypoints, text from images. Use cases:

- Safety and security (person detection, PPE checks)
- Retail and logistics (count items, low stock alerts, QR or barcodes)
- Documents (scan receipts or invoices, OCR)
- Manufacturing (find defects)
- Sports or fitness (pose based feedback)
- Media (background removal, simple AR)

2) Core ideas you need

- **Pixels and channels:** an image is height x width x channels (RGB or GRAY).
- **Color spaces:** RGB for display; HSV is handy for color thresholds.
- **Basic operations:** resize, crop, rotate; blur and edge detection.
- **Tasks this course uses:**

- **Classification** (single label for the image) - **Object detection** (boxes with class names) - **Segmentation** (per pixel mask) - **Tracking** (keep the same ID across frames) - **OCR** (read text from images) - **Pose** (keypoints like shoulders and knees for angles)

- **Metrics you will see:** IoU (box overlap), mAP (detection quality), FPS and latency (speed).
- **Inference loop:** read frame -> preprocess -> run model -> postprocess (NMS, thresholds) -> draw or emit events.
- **Responsible use:** get consent when recording; avoid storing faces or plates unless required.

3) Python libraries (what to import and why)

• **OpenCV** - image and video IO and drawing

import: import cv2 **why:** read cameras or videos, convert colors, draw boxes or text, basic filters.

• **NumPy** - numeric arrays for images

import: import numpy as np **why:** fast math on pixel arrays and simple image operations.

• **Pillow (PIL)** - lightweight image loading

import: from PIL import Image **why:** open or save images and quick format conversions.

• **Ultralytics YOLO** - pretrained detectors, segmenters, and pose

import: from ultralytics import YOLO **why:** run strong models with a few lines.

• **MediaPipe** - fast pose or hand or face landmarks

import: import mediapipe as mp **why:** quick pose or landmarks for fitness or gestures.

• **OCR**

Tesseract: import pytesseract **EasyOCR:** import easyocr **why:** text extraction from images without training.

• **Tracking helpers**

DeepSORT or ByteTrack (from a repo) usually provide a Tracker class. **why:** keep object IDs consistent frame to frame.

• **App frameworks**

Streamlit: import streamlit as st **Gradio:** import gradio as gr **why:** build quick UIs for your demo apps.

• **Optional deployment runtime**

ONNX Runtime: import onnxruntime as ort **why:** portable and fast inference on CPU or GPU.

4) Minimal workflows with code and short explanations

4.1 Open a camera, draw FPS, press ESC to quit (OpenCV)

```
import cv2, time

cap = cv2.VideoCapture(0)  # 0 = default webcam; or put a video path here

while True:
    ok, frame = cap.read()
    if not ok:
        break

    t0 = time.time()

    # Example processing: convert to grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Show an FPS estimate on the frame
    fps = 1.0 / max(1e-6, time.time() - t0)
    cv2.putText(gray, f"FPS: {fps:.1f}", (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 1.0, (255), 2)

    cv2.imshow("preview", gray)
    if cv2.waitKey(1) == 27:  # ESC key
        break

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

What this does: opens the webcam, converts each frame to grayscale, overlays FPS, and displays the stream. Press ESC to close.

4.2 Run a pretrained YOLO detector (Ultralytics) on webcam

```
from ultralytics import YOLO

# Choose a small model for speed; switch to yolov8n-seg.pt for segmentation or -pose.pt for pose
model = YOLO("yolov8n.pt")

# show=True opens a simple window; stream=True yields results per frame
for result in model.predict(source=0, show=True, stream=True, conf=0.25):
    # result contains .boxes (xyxy, conf, cls) and the plotted image via result.plot()
    pass
```

What this does: loads a pretrained YOLO model and runs it on the webcam, drawing boxes and class names automatically.

4.3 Basic OCR with Tesseract

```
import cv2, pytesseract

img = cv2.imread("receipt.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Increase contrast and binarize for better OCR
th = cv2.adaptiveThreshold(gray, 255,
                           cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                           cv2.THRESH_BINARY, 31, 10)

# Set language pack to English; install others as needed (e.g., 'ron' for Romanian)
text = pytesseract.image_to_string(th, lang="eng")
print(text)
```

What this does: reads an image, cleans it up, and extracts text. Works best on sharp, high contrast images.

4.4 Simple multi object tracking pipeline (YOLO + DeepSORT or ByteTrack style)

```
# Pseudocode - the tracker API may differ by repo
from ultralytics import YOLO
from tracker import Tracker  # e.g., a DeepSORT or ByteTrack wrapper you add to your project
import cv2

model = YOLO("yolov8n.pt")
tracker = Tracker()

for r in model.predict(source="video.mp4", stream=True, conf=0.3):
    # Convert YOLO detections to (xyxy, score, class) for the tracker
    dets = []
    for b in r.boxes:
        xyxy = b.xyxy[0].tolist()
        score = float(b.conf[0])
        cls = int(b.cls[0])
        dets.append((xyxy, score, cls))

    tracks = tracker.update(dets)  # returns objects with track_id and bbox

    # Draw IDs on the plotted image
    im = r.plot()
    for tr in tracks:
        x1, y1, x2, y2 = tr.bbox
        cv2.putText(im, f"ID {tr.track_id}", (int(x1), int(y1)-5),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,255,0), 2)

    # Optionally show: cv2.imshow("tracked", im); if cv2.waitKey(1) == 27: break
```

What this does: runs detection per frame, then hands boxes to a tracker to keep consistent IDs, and draws the ID labels.

5) Quick references

- **OpenCV search terms:** VideoCapture, cvtColor, putText, imshow
- **Ultralytics:** predict stream, yolov8n, seg, pose
- **OCR:** pytesseract image_to_string, EasyOCR Reader
- **Tracking:** DeepSORT Python, ByteTrack Python
- **App UIs:** streamlit camera, gradio image