Computer Vision Assignment 2

Sourish Das

Task 1: Image Processing Functions

Code

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

def capture_image(index=0):
    cap = cv2.VideoCapture(index)
    if not cap.isOpened():
        raise IOError("Cannot open webcam")
    ret, frame = cap.read()
    cap.release()
    if not ret:
        raise IOError("Failed to capture image")
    return cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

def convert_to_hsv(image):
    hsv = cv2.cvtColor(image, cv2.COLOR_RGB2HSV)
    h, s, v = cv2.split(hsv)
    return hsv, h, s, v

def equalize_histogram(image):
    gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    eq = cv2.equalizeHist(gray)
    return gray, eq

def binary_inversion(image, threshold=127):
    gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    _, binary = cv2.threshold(gray, threshold, 255, cv2.
       THRESH_BINARY_INV)
    return binary

def posterize_gray(image):
    gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    poster = (gray // 64) * 64
    return poster

def edge_detection(image):
    gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    laplacian = cv2.Laplacian(gray, cv2.CV_64F)
    scharr = cv2.Scharr(gray, cv2.CV_64F, 1, 0)
    return np.uint8(np.abs(laplacian)), np.uint8(np.abs(scharr))

def denoise_median(image):
    noisy = image.copy()
    h, w, _ = noisy.shape
    num_noise = int(0.01 * h * w)
    for _ in range(num_noise):
        x, y = np.random.randint(0, w), np.random.randint(0, h)
        noisy[y, x] = np.random.choice([0, 255], 3)
    denoised = cv2.medianBlur(noisy, 3)
    return noisy, denoised

def unsharp_mask(image, amount=1.5):
    blurred = cv2.GaussianBlur(image, (9, 9), 10)
    sharpened = cv2.addWeighted(image, 1 + amount, blurred, -amount, 0)
    return sharpened

def convert_to_lab(image):
    lab = cv2.cvtColor(image, cv2.COLOR_RGB2LAB)
    l, a, b = cv2.split(lab)
    return lab, l, a, b
```

Sample Output Plot

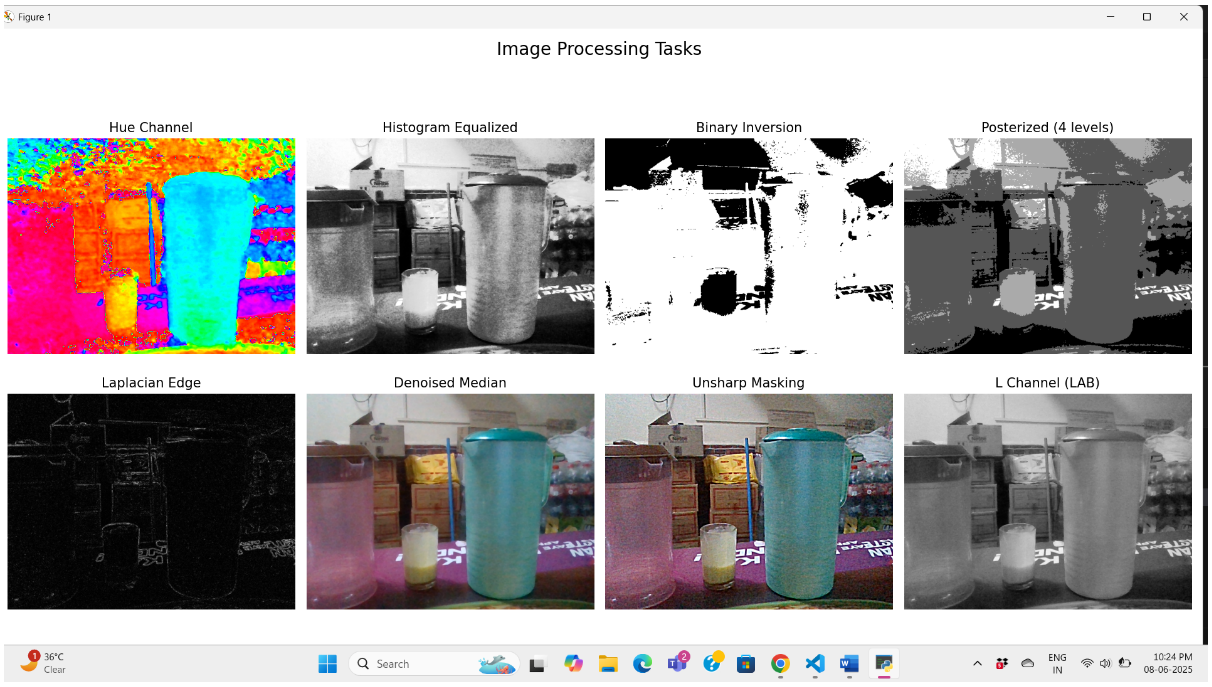


Figure 1: Results of various image processing operations such as HSV transformation, histogram equalization, edge detection, etc.

Task 3: Flag Recognition (Indonesia vs Poland)

Code

```
import cv2

def checkFlag(image_path):
    image = cv2.imread(image_path)
    if image is None:
        raise ValueError("Image not read.")
    image = cv2.resize(image, (300, 200))
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    top_half = hsv[:100, :]
    bottom_half = hsv[100:, :]

    def is_red(region):
        mask1 = cv2.inRange(region, (0, 70, 50), (10, 255, 255))
        mask2 = cv2.inRange(region, (170, 70, 50), (180, 255, 255))
        red_pixels = cv2.countNonZero(mask1 | mask2)
        return red_pixels / region.size > 0.2

    def is_white(region):
        mask = cv2.inRange(region, (0, 0, 200), (180, 50, 255))
        white_pixels = cv2.countNonZero(mask)
        return white_pixels / region.size > 0.2

    if is_red(top_half) and is_white(bottom_half):
        return "Indonesia"
    elif is_white(top_half) and is_red(bottom_half):
        return "Poland"
    else:
        return "Uncertain"
```

Logic Explanation

The input image is resized to a uniform size. It's converted to HSV to simplify color detection under varying lighting. The flag is divided into two regions: top and bottom. Based on color ratios using HSV masks, the function determines whether the flag is Indonesia (red over white), Poland (white over red), or uncertain.

Task 4: YOLOv5 Research Summary

1. Introduction and Objectives

YOLOv5 is a real-time object detection system developed by Ultralytics. It improves on YOLOv3 by implementing in PyTorch and optimizing training/inference speed. Key objectives include:

- Comparing different YOLOv5 variants (n, s, m, l, x)
- Analyzing architectural improvements like CSPDarknet, PANet
- Transition from Darknet to PyTorch

2. Evolution and Development

- April–June 2020: Progressive feature releases
- YOLOv5's popularity stems from its Python-first design and easy customization

3. Architecture and Training

- **Backbone:** CSPDarknet for feature extraction
- **Neck:** PANet for feature aggregation
- **Head:** Outputs bounding boxes and class labels

Training includes augmentation (Mosaic), mixed precision (FP16), and CIoU loss.

4. Key Innovations

- **CSP Backbone:** Feature split and merge to reduce computation
- **PANet:** Improved multi-scale detection
- **Anchor Boxes:** Improved via clustering and evolution
- **PyTorch Framework:** Enhances accessibility

5. Deployment and Tools

- Supports tools like Roboflow, ClearML, LabelImg
- YOLO TXT + YAML format for annotations

6. Conclusion

YOLOv5 provides an excellent tradeoff between speed, accuracy, and flexibility, making it suitable for real-time and embedded systems.

Task 2 Outputs and Visuals

Original Image Captured via Webcam

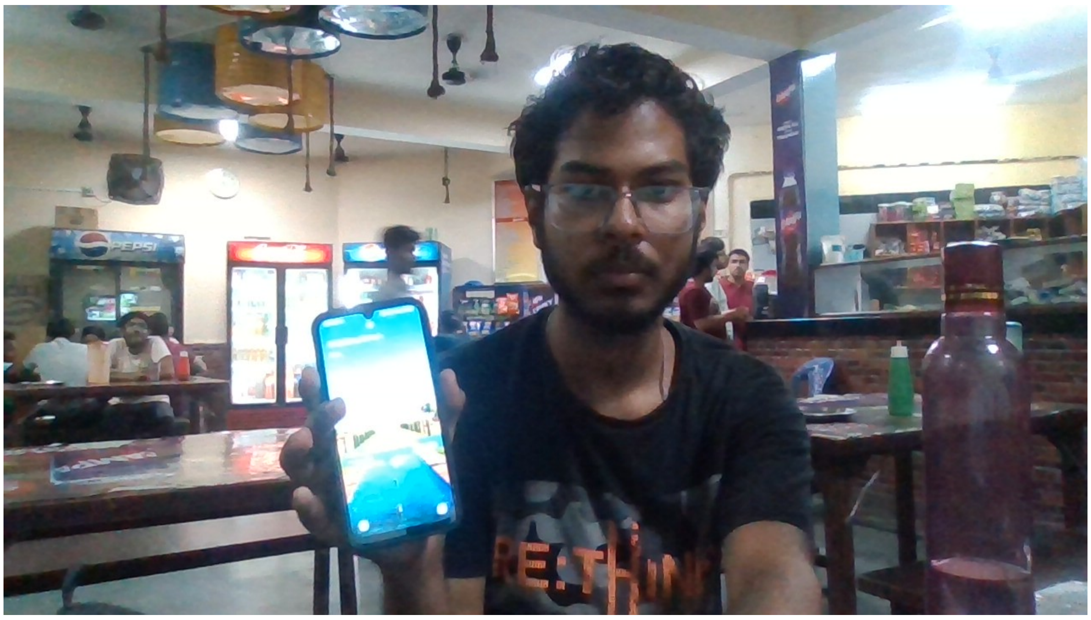


Figure 2: Original captured image used for object detection.

YOLOv5 Detection Result

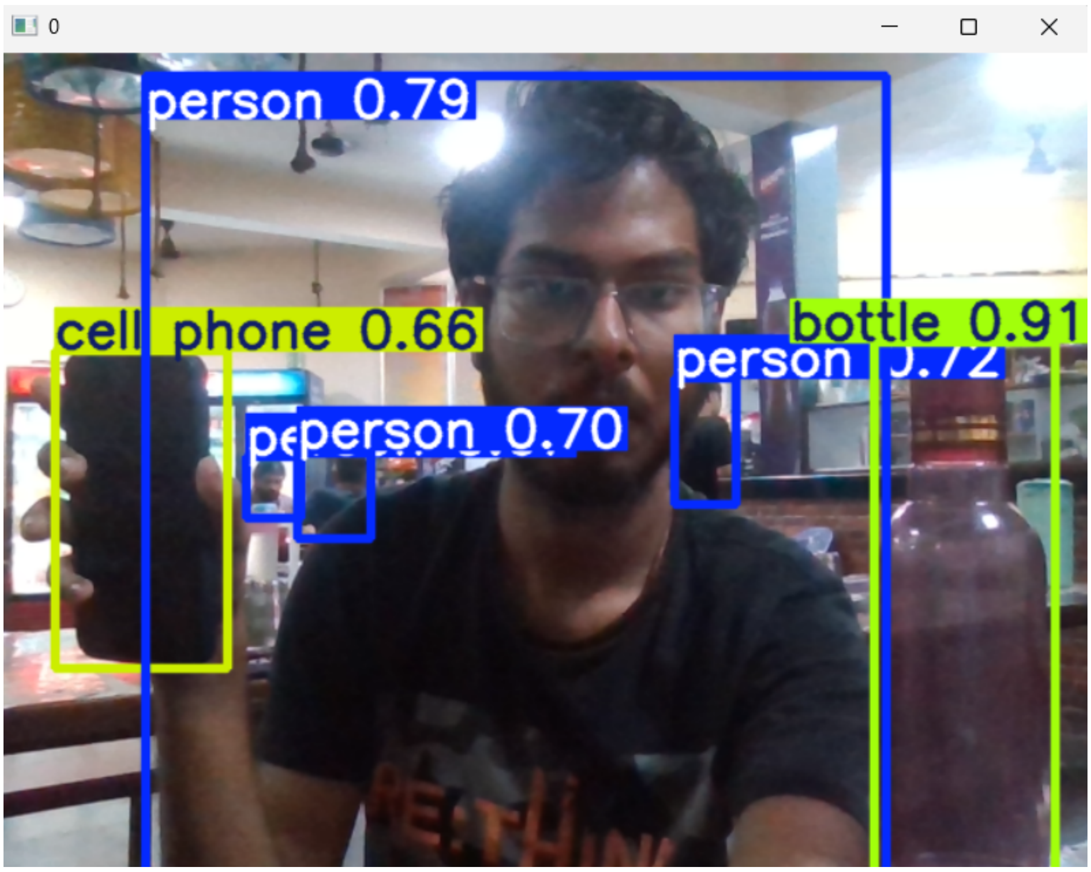
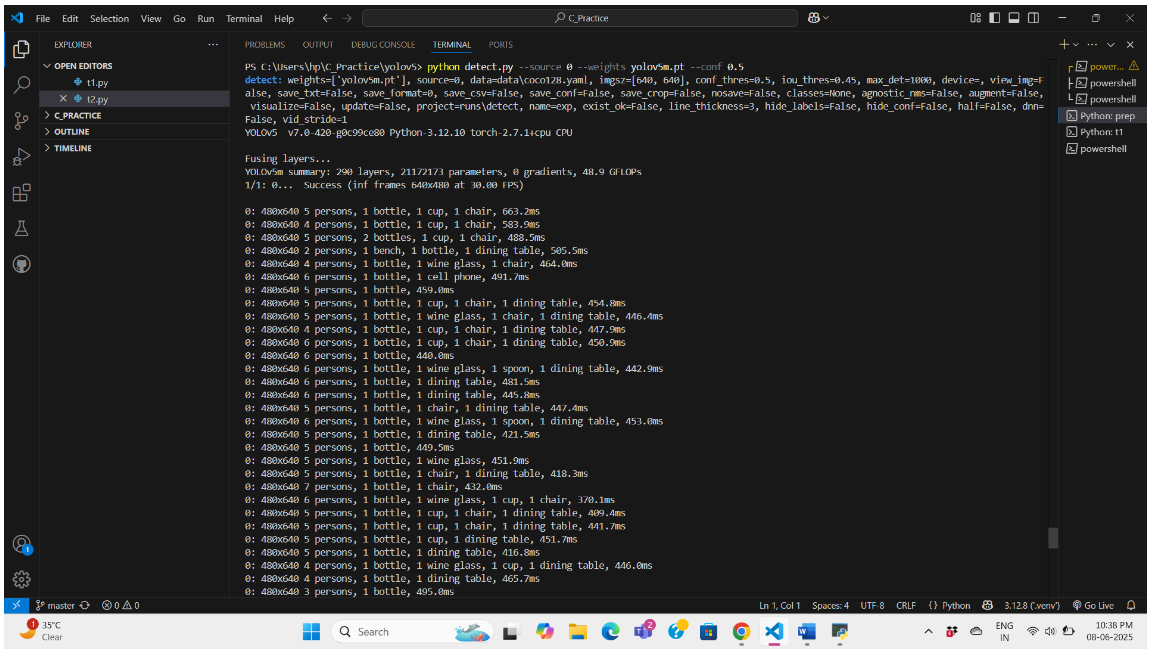


Figure 3: YOLOv5 applied on the webcam image; detects people, phone, and bottle.

YOLOv5 Inference Terminal Output



```
PS C:\Users\hp\C_Practice\yolov5> python detect.py --source 0 --weights yolov5m.pt --conf 0.5
detect: weights=['yolov5m.pt'], source=0, data=data\coco128.yaml, imgsz=[640, 640], conf_thres=0.5, iou_thres=0.45, max_det=1000, device=, view_img=F
alse, save_txt=False, save_format=0, save_csv=False, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False,
 visualize=False, update=False, project=runs\detect, name=exp, exist_ok=False, line_thickness=3, hide_labels=False, hide_conf=False, half=False, dnn=
False, vid_stride=1
YOLOv5  v7.0-420-g0c99ce80 Python-3.12.10 torch-2.7.1+cpu CPU

Fusing layers...
YOLOv5m summary: 290 layers, 21172173 parameters, 0 gradients, 48.9 GFLOPs
1/1: 0...  Success (inf frames 640x480 at 30.00 FPS)

0: 480x640 5 persons, 1 bottle, 1 cup, 1 chair, 663.2ms
0: 480x640 4 persons, 1 bottle, 1 cup, 1 chair, 583.9ms
0: 480x640 5 persons, 2 bottles, 1 cup, 1 chair, 488.5ms
0: 480x640 2 persons, 1 bench, 1 bottle, 1 dining table, 505.5ms
0: 480x640 4 persons, 1 bottle, 1 wine glass, 1 chair, 464.0ms
0: 480x640 6 persons, 1 bottle, 1 cell phone, 491.7ms
0: 480x640 5 persons, 1 bottle, 459.0ms
0: 480x640 5 persons, 1 bottle, 1 cup, 1 chair, 1 dining table, 454.8ms
0: 480x640 5 persons, 1 bottle, 1 wine glass, 1 chair, 1 dining table, 446.4ms
0: 480x640 4 persons, 1 bottle, 1 cup, 1 chair, 1 dining table, 447.9ms
0: 480x640 6 persons, 1 bottle, 1 cup, 1 chair, 1 dining table, 450.9ms
0: 480x640 6 persons, 1 bottle, 440.0ms
0: 480x640 6 persons, 1 bottle, 1 wine glass, 1 spoon, 1 dining table, 442.9ms
0: 480x640 6 persons, 1 bottle, 1 dining table, 481.5ms
0: 480x640 6 persons, 1 bottle, 1 dining table, 445.8ms
0: 480x640 5 persons, 1 bottle, 1 chair, 1 dining table, 447.4ms
0: 480x640 6 persons, 1 bottle, 1 wine glass, 1 spoon, 1 dining table, 453.0ms
0: 480x640 5 persons, 1 bottle, 1 dining table, 421.5ms
0: 480x640 5 persons, 1 bottle, 449.5ms
0: 480x640 5 persons, 1 bottle, 1 wine glass, 451.9ms
0: 480x640 5 persons, 1 bottle, 1 chair, 1 dining table, 418.3ms
0: 480x640 7 persons, 1 bottle, 1 chair, 432.0ms
0: 480x640 6 persons, 1 bottle, 1 wine glass, 1 cup, 1 chair, 370.1ms
0: 480x640 5 persons, 1 bottle, 1 cup, 1 chair, 1 dining table, 409.4ms
0: 480x640 5 persons, 1 bottle, 1 cup, 1 chair, 1 dining table, 441.7ms
0: 480x640 5 persons, 1 bottle, 1 cup, 1 dining table, 451.7ms
0: 480x640 5 persons, 1 bottle, 1 dining table, 416.8ms
0: 480x640 4 persons, 1 bottle, 1 wine glass, 1 cup, 1 dining table, 446.0ms
0: 480x640 4 persons, 1 bottle, 1 dining table, 465.7ms
0: 480x640 3 persons, 1 bottle, 495.0ms
```

Figure 4: YOLOv5 terminal output showing detected objects across frames with inference time.