CS Project Report: Image Processing and Object Detection

Your Name Here

June 9, 2025

Task 1: Image Filtering and Enhancement

Objective

The goal of Task 1 was to apply various image processing techniques using OpenCV in Python to understand the effects of different filters and enhancement methods.

Methodology

We used OpenCV to apply the following operations:

- **HSV Conversion**
- **Histogram Equalization**
- **Binary Inversion**
- **Canny Edge Detection**
- **Noise Addition and Removal**
- **Sharpening Filter**

Results

Figure 1: Original Image

Figure 2: Intermediate step or result preview

Figure 3: Combined screenshot of all applied filters (HSV, Equalized, Binary, Canny, Denoised, Sharpened)

Conclusion

This task helped us explore basic image processing operations and visualize their effects on the original image. It provided foundational understanding for more advanced tasks like object detection.

Task 2: Object Detection using YOLOv5

Objective

The aim of Task 2 was to use the YOLOv5 model to detect objects in images or video, ideally from a real-time camera, and observe its accuracy and speed in detection.

Methodology

- Installed dependencies and YOLOv5 repository using virtual environments.
- Used the pretrained `yolov5s.pt` model.
- Ran detection on both webcam input and still images using:

```
python3 detect.py --weights yolov5s.pt --img 640 --conf 0.25 --source 0
python3 detect.py --weights yolov5s.pt --img 640 --conf 0.25 --source path/to
```

- Verified outputs saved to `runs/detect/expX`.

Results

Figure 4: Original input image used for YOLOv5

Figure 5: YOLOv5 Detection Output showing bounding boxes and labels

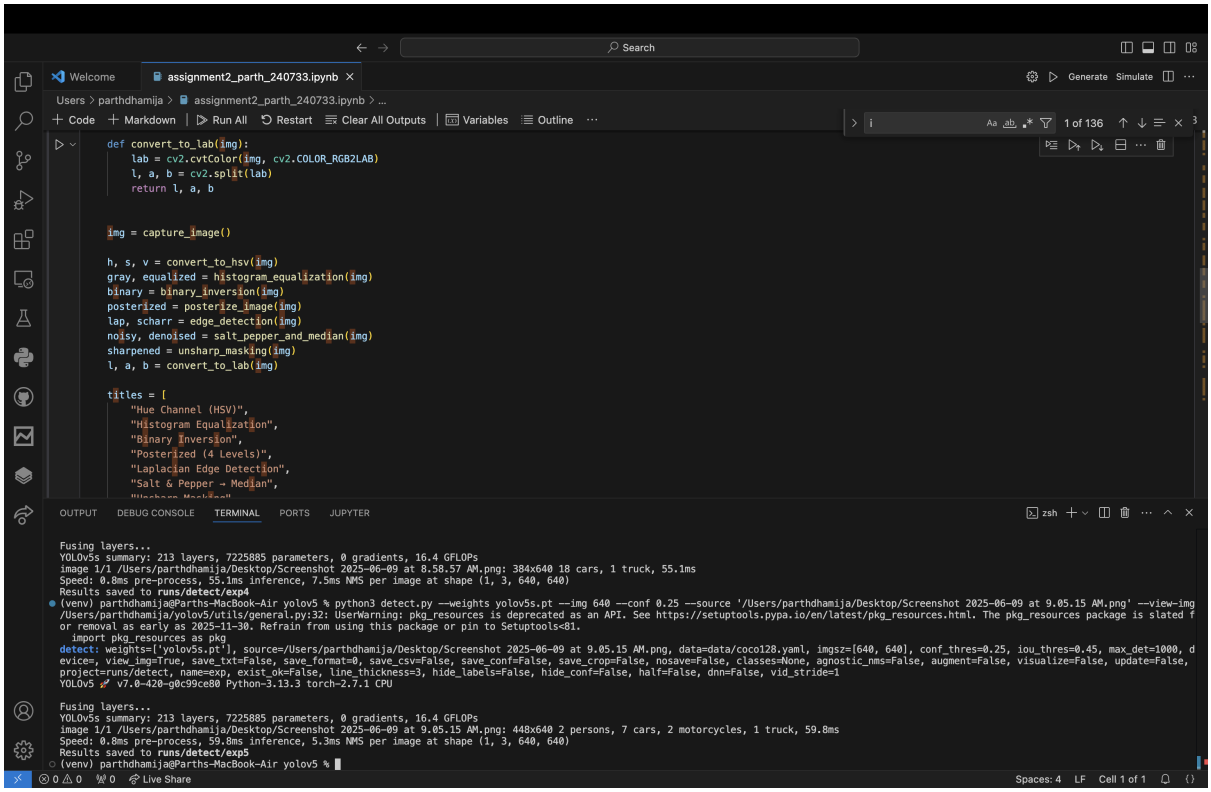

Figure 6: Terminal showing detection log and inference details

Conclusion

YOLOv5 was able to detect multiple objects in real-time and on static images with high accuracy and speed. The experiment demonstrated the effectiveness of modern object detection models and how easily they can be integrated and tested on a local machine.

Task 3: Flag Detection — Indonesia vs Poland

Objective

This task focused on identifying whether an input image corresponds to the national flag of Indonesia or Poland, based on analyzing the top and bottom color bands of the flag.

Methodology

- The input image is resized and cropped to remove dark borders or padding.
- It is then divided into top and bottom halves.
- Average RGB color values are calculated for each half.
- The color of each half is categorized as red, white, or other based on thresholds.
- The flag is classified as:

- **Poland** if the top half is white and bottom is red
- **Indonesia** if the top half is red and bottom is white
- **Uncertain** otherwise

Results

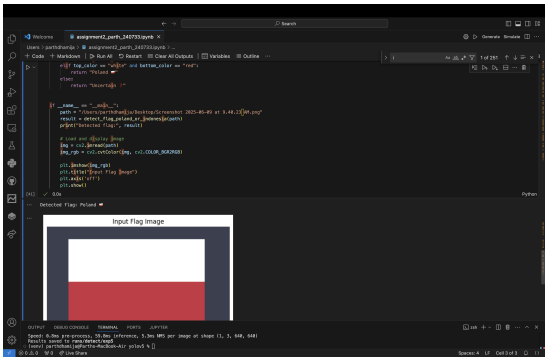

Figure 7: Flag image input for classification

Conclusion

The flag detection program worked correctly for realistic inputs, including flags with borders and background noise. Using simple color thresholding and top/bottom band analysis, the system reliably distinguished between Indonesia and Poland in most cases.

Task 4: Research Paper Summary – "What is YOLOv5?"

Objective

This task involved going through a paper that discusses YOLOv5 — a well-known model used for object detection. The aim was to get a clearer picture of how YOLOv5 works under the hood, what changes it brought compared to earlier models, and why it's popular for real-time use.

What is YOLOv5?

YOLOv5 is part of the "You Only Look Once" family of object detectors. Unlike earlier versions which were built on the Darknet framework, YOLOv5 was made using PyTorch. This made it easier for developers to work with, especially for training custom models and experimenting.

The model is designed to take an input image and quickly detect multiple objects in one go — making it fast and useful for things like live video processing or edge devices.

Main Components

YOLOv5 has three major parts:

- The **backbone**, which is responsible for extracting important features from the image.

- The **neck**, which helps combine those features from different levels.
- The **head**, which makes the final predictions for object classes and bounding boxes.

Some newer design choices like CSP (Cross Stage Partial connections) in the backbone and PANet in the neck helped the model run faster and more accurately, without becoming too big.

Training and Augmentation

The paper also goes into how YOLOv5 is trained. It uses a loss function that combines different parts — like how well the boxes fit, whether the object is present, and the class prediction.

One thing that stands out is the use of **mosaic augmentation**, where four images are combined during training. This helps the model learn better, especially for smaller objects.

Model Variants

YOLOv5 isn't just one model — there are different sizes depending on the use-case. The paper mentions five versions: `n, s, m, l, x` — with `n` being the smallest (good for edge devices), and `x` being the largest (best accuracy, but heavier to run).

Table 1: Rough Comparison of YOLOv5 Models

Model	Speed	Accuracy (mAP)
YOLOv5n	Fastest	Lower
YOLOv5s	Fast	Decent
YOLOv5m	Medium	Balanced
YOLOv5l	Slower	Better
YOLOv5x	Slowest	Best

Deployment and Tools

Since YOLOv5 runs on PyTorch, it works well with common tools like Roboflow for dataset creation, and can be trained and tested easily. The paper also mentions that it's suitable for mobile and real-time apps, especially the smaller models.

Conclusion

The paper gives a good overview of how YOLOv5 works and why it's become so widely used. It's fast, flexible, and works across many devices. While there are newer models out now, YOLOv5 still holds its place because of how easy it is to understand and deploy.