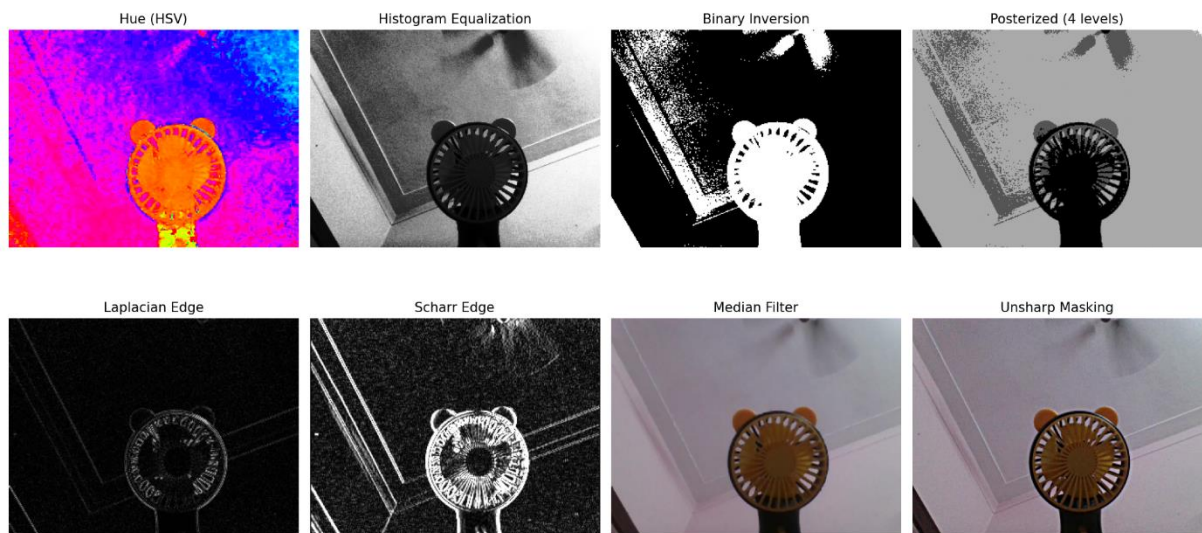


Assignment 2

What I have learnt while solving assignment

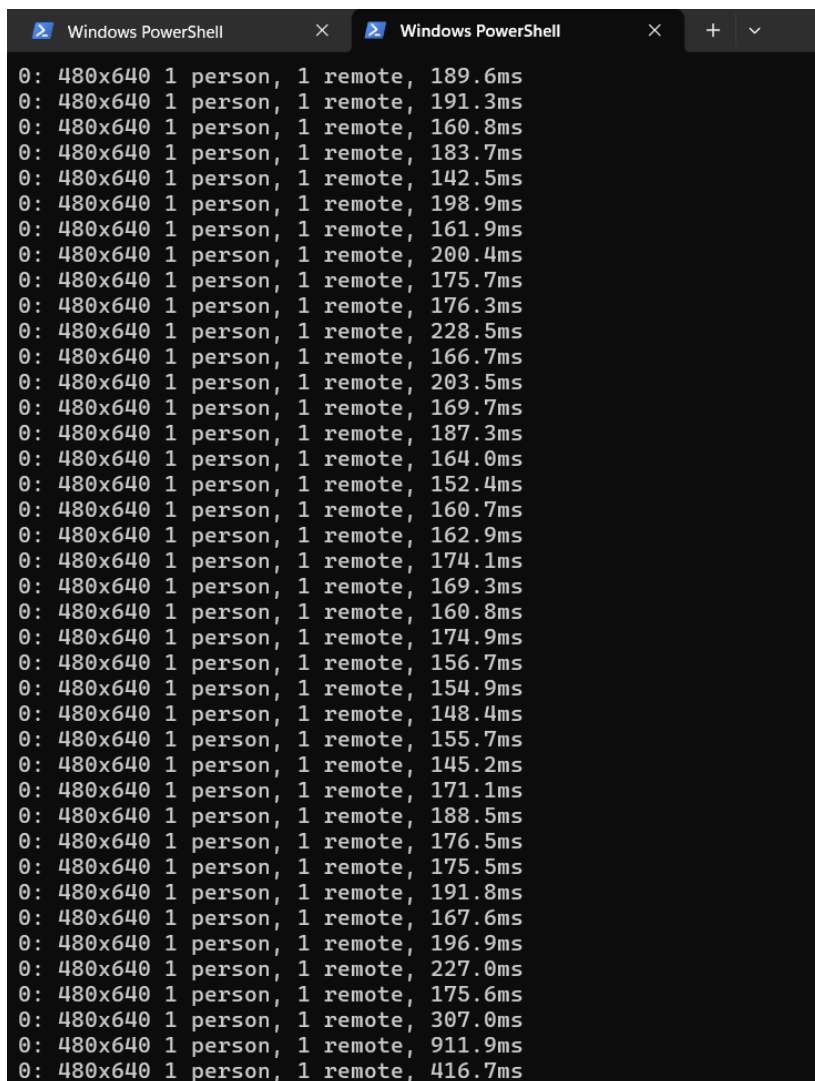
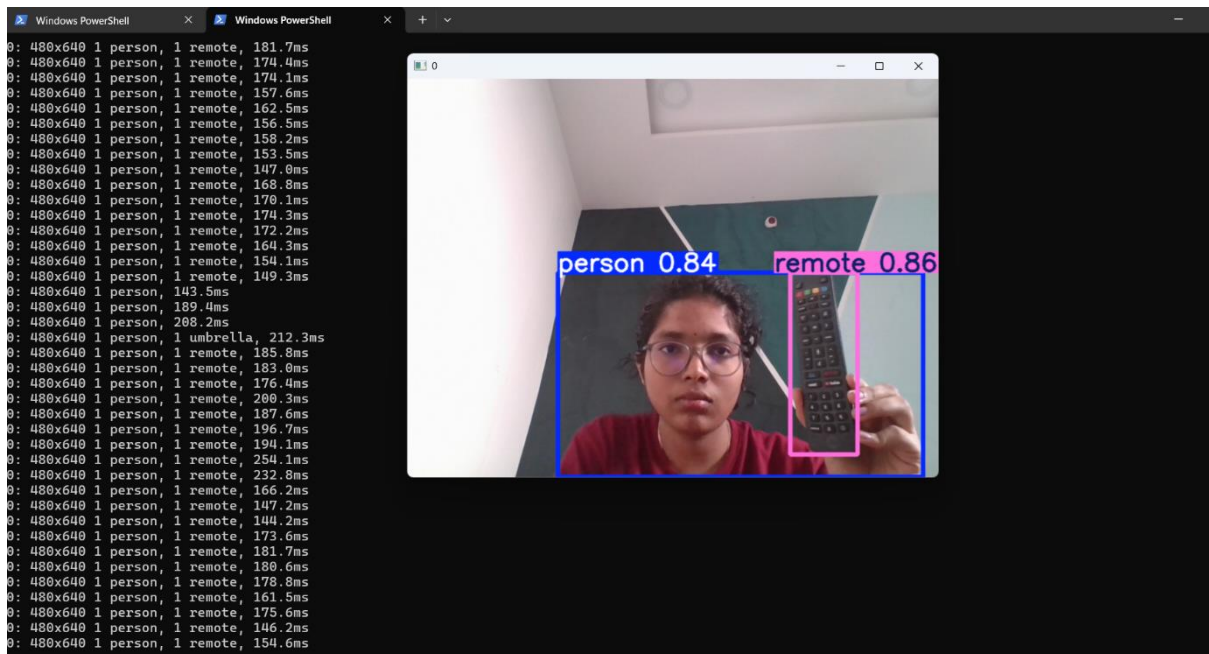
Working on this assignment gave me a hands-on understanding of image processing and computer vision. I learned how to manipulate images using OpenCV, apply various filters and transformations, and implement object detection using YOLOv5. The flag identification task was especially interesting because it required me to think about colour spaces and real-world variations in images. Summarizing the YOLO research paper also helped me grasp the evolution and strengths of modern object detection models.

TASK-1

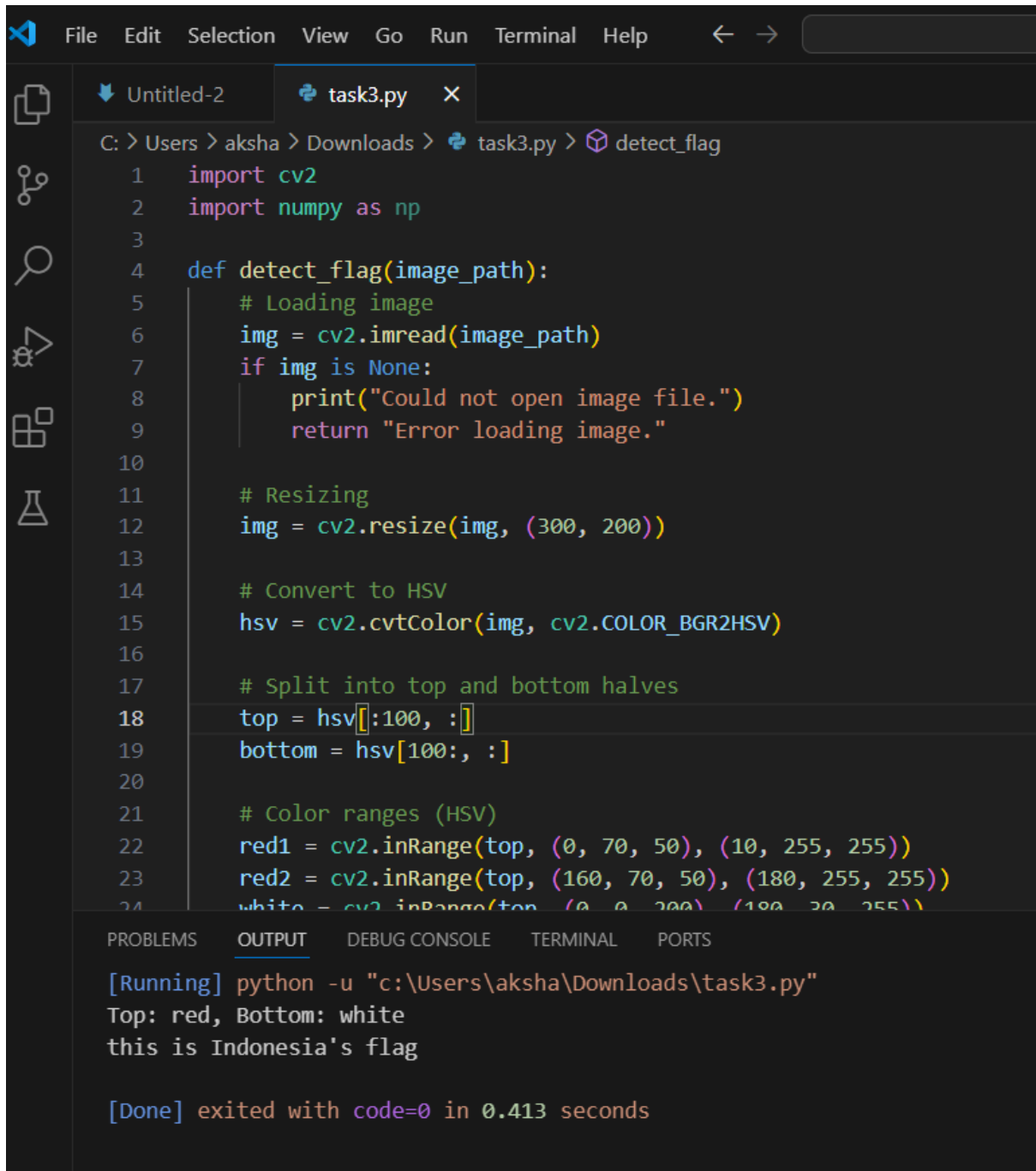


TASK-2





TASK-3



The image shows a Visual Studio Code editor window with a Python script named `task3.py` open. The script is located at `C:\Users\aksha\Downloads\task3.py` and contains a function `detect_flag` that processes an image to identify the Indonesian flag. The script uses `cv2` for image processing and `numpy` for array operations. It loads the image, resizes it to 300x200 pixels, converts it to HSV, and then splits it into top and bottom halves. It then uses `cv2.inRange` to identify red and white regions. The output of the script is displayed in the terminal window at the bottom.

```
C: > Users > aksha > Downloads > task3.py > detect_flag
1  import cv2
2  import numpy as np
3
4  def detect_flag(image_path):
5      # Loading image
6      img = cv2.imread(image_path)
7      if img is None:
8          print("Could not open image file.")
9          return "Error loading image."
10
11     # Resizing
12     img = cv2.resize(img, (300, 200))
13
14     # Convert to HSV
15     hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
16
17     # Split into top and bottom halves
18     top = hsv[:100, :]
19     bottom = hsv[100:, :]
20
21     # Color ranges (HSV)
22     red1 = cv2.inRange(top, (0, 70, 50), (10, 255, 255))
23     red2 = cv2.inRange(top, (160, 70, 50), (180, 255, 255))
24     white = cv2.inRange(bottom, (0, 0, 200), (180, 20, 255))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[Running] python -u "c:\Users\aksha\Downloads\task3.py"
Top: red, Bottom: white
this is Indonesia's flag

[Done] exited with code=0 in 0.413 seconds
```

Example image :



CODE :

```
import cv2
import numpy as np

def detect_flag(image_path):
    # Loading image
    img = cv2.imread(image_path)
    if img is None:
        print("Could not open image file.")
        return "Error loading image."

    # Resizing
    img = cv2.resize(img, (300, 200))

    # Convert to HSV
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    # Split into top and bottom halves
    top = hsv[:100, :]
    bottom = hsv[100:, :]

    # Color ranges (HSV)
    red1 = cv2.inRange(top, (0, 70, 50), (10, 255, 255))
    red2 = cv2.inRange(top, (160, 70, 50), (180, 255, 255))
    white = cv2.inRange(top, (0, 0, 200), (180, 30, 255))
    top_red = cv2.countNonZero(red1) + cv2.countNonZero(red2)
    top_white = cv2.countNonZero(white)

    red1_b = cv2.inRange(bottom, (0, 70, 50), (10, 255, 255))
    red2_b = cv2.inRange(bottom, (160, 70, 50), (180, 255, 255))
    white_b = cv2.inRange(bottom, (0, 0, 200), (180, 30, 255))
    bottom_red = cv2.countNonZero(red1_b) + cv2.countNonZero(red2_b)
    bottom_white = cv2.countNonZero(white_b)

    # Figure out which color is dominant in each half
    top_color = 'red' if top_red > top_white else 'white'
    bottom_color = 'red' if bottom_red > bottom_white else 'white'

    # Debug prints
    print(f"Top: {top_color}, Bottom: {bottom_color}")

    if top_color == 'red' and bottom_color == 'white':
        return "this is Indonesia's flag "
    elif top_color == 'white' and bottom_color == 'red':
        return "this is Poland's flag"
    else:
        # TODO: Add more flags later
```

```
return "Sorry, I couldn't recognize this flag."
```

```
# Example usage  
result = detect_flag("flag1")  
print(result)
```

LOGIC HOW THE CODE WORKS

1.	Image Preparation
	<ul style="list-style-type: none">• First loads & resizes the image to 300x200 pixels (makes processing easier)• Converts colors to HSV format (better for color detection than regular RGB)
2.	Split Analysis
	<ul style="list-style-type: none">• Cuts the image into top and bottom halves (Assumption: Flags have two horizontal color bands)
3.	Color Detection
	For Red:
	<ul style="list-style-type: none">• Checks for two red ranges (0-10° and 160-180° in HSV) (Red appears at both ends of the hue spectrum)• Counts all red pixels
	For White:
	<ul style="list-style-type: none">• Looks for high brightness (200-255) with low saturation (0-30)• Counts all white pixels
4.	Decision Making
	<ul style="list-style-type: none">• Compares which color (red/white) has more pixels in each half• Top red + bottom white = Indonesia• Top white + bottom red = Poland

TASK-4

YOLOv5 Research Paper Summary (Simple, Human-like Language)

Introduction to YOLO and YOLOv5

YOLO, which stands for "You Only Look Once," is a well-known method in computer vision for finding and identifying objects in images or videos. Unlike older methods that look at an image multiple times, YOLO checks the whole image just once and quickly predicts where objects are and what they are¹. YOLOv5 is one of the latest versions, and it's popular because it's fast, accurate, and easy to use.

Why YOLOv5 is Important

YOLOv5 is special because it works well on different devices, from powerful computers to small gadgets like phones or cameras. It's used in many real-world tasks, like checking products in factories or monitoring traffic, because it can spot objects in real time¹.

How YOLOv5 Evolved

YOLOv5 was developed after YOLOv3 and YOLOv4, but it was the first to be built using PyTorch, a popular deep learning library. This made it easier for more people to use and improve the model. The developers, Ultralytics, kept adding new features to make YOLOv5 faster and more accurate, like better ways to handle data and smarter model parts¹.

Key Features and Innovations in YOLOv5

- **Architecture:** YOLOv5 is made up of three main parts:
 - *Backbone:* This part takes the image and pulls out important features using a special method called CSP (Cross-Stage Partial), which makes the model faster and less complicated.
 - *Neck:* This section combines features from different layers using PA-Net, helping the model understand objects at different sizes.
 - *Head:* This final part predicts where objects are and what they are¹.
- **Training Tricks:**
 - *Data Augmentation:* YOLOv5 uses tricks like changing image colors, sizes, and mixing four images into one (mosaic augmentation). This helps the model learn better, especially for small objects.
 - *Loss Function:* The model uses a special formula to measure how well it's doing, combining scores for object location, type, and confidence. It uses something called CIoU and BCE for these calculations.
 - *16-Bit Precision:* YOLOv5 can run using less computer memory, making it faster on some hardware¹.
- **Anchor Boxes:** These are like templates that help the model guess where objects might be in an image. YOLOv5 uses smart ways to choose these templates, making it work better for different types of images¹.

How to Use YOLOv5

- **Annotation:** YOLOv5 uses a simple text format to label objects in images. Tools like Roboflow, LabelImg, and CVAT help create these labels easily.
- **Integration:** YOLOv5 works with many platforms for easier training, tracking, and deployment, like Roboflow for labeling and Weights & Biases for monitoring training progress¹.

Why YOLOv5 Stands Out

- *Flexible:* You can pick a model size based on your device and needs.
- *Fast and Accurate:* It's great for real-time tasks where speed matters.
- *Easy to Use:* Thanks to PyTorch, more people can train and use the model for their own projects.

- *Good for Small Objects:* Mosaic augmentation and other tricks help detect even tiny objects in images¹.

Conclusion

YOLOv5 is a powerful tool for object detection. It's fast, accurate, and easy to use on many devices. The improvements in its design and training methods make it a top choice for both research and real-world applications, from factories to smart cameras. As computer vision keeps growing, YOLOv5 will likely remain a strong foundation for future work in this area¹.

GIT REPOSITORY LINKS

[Guthikonda-Akshaya/GanForge at assignment-2](#)