Task-3

https://github.com/CODER-7777/GanForge/blob/vivekananda_m/Task-3

Reading and Resize the Image

```
image = cv2.imread(image_path)
image = cv2.resize(image, (400, 300))
```

- Reads the image from disk.
- Resizes it to a fixed dimension (400×300) to standardize input and reduce computation.

Blurring to Remove Noise

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

- Applies Gaussian Blur to smooth out small irregularities like wrinkles or noise.
- Helps in consistent color segmentation.

Converting to HSV Color Space

```
hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)
```

- HSV (Hue, Saturation, Value) is more stronger than RGB for color detection.
  - **Hue**: actual color
  - **Saturation**: purity of the color
  - **Value**: brightness

Creating Color Masks

Red Mask

```
red_lower1 = np.array([0, 70, 50])
red_upper1 = np.array([10, 255, 255])
```

```
red_lower2 = np.array([160, 70, 50])
red_upper2 = np.array([180, 255, 255])
```

- Red hue appears at two ends of the HSV scale: near 0 and near 180.
- So we create **two red ranges** to capture both.

```
red_mask = cv2.inRange(hsv, red_lower1, red_upper1) | cv2.inRange(hsv, red_
```

- Combines both red ranges using bitwise O

White Mask

```
white_lower = np.array([0, 0, 200])
white_upper = np.array([180, 50, 255])
white_mask = cv2.inRange(hsv, white_lower, white_upper)
```

- White has **low saturation** and **high value** in HSV.
- This range captures bright, low-saturation regions.

Find Largest Red/White Region

```
combined_mask = cv2.bitwise_or(red_mask, white_mask)
contours, _ = cv2.findContours(combined_mask, cv2.RETR_EXTERNAL, cv2.C
```

- Merges both red and white masks.
- Finds **external contours** (continuous regions of same color).
- Assumes that the largest such region is likely the flag.

```
if not contours:
    return "Unable to detect flag region
```

- Handles failure case if no contours are detected.

```


largest_contour = max(contours, key=cv2.contourArea)
x, y, w, h = cv2.boundingRect(largest_contour)
```

```
flag_roi = image[y:y+h, x:x+w]
```

- Picks the largest contour.
- Extracts its bounding rectangle.
- Crops that part from the image → likely to contain the flag.

6. Resize and Segment the Flag

```
flag_roi = cv2.resize(flag_roi, (200, 100))
flag_hsv = cv2.cvtColor(flag_roi, cv2.COLOR_BGR2HSV)

top_half = flag_hsv[:50, :]
bottom_half = flag_hsv[50:, :]
```

- Standardizes the flag region to 200×100 pixels.
- Converts cropped flag to HSV.
- Splits into **top half (0–50 rows)** and **bottom half (50–100 rows)**.

Measure Red and White Ratios

Red Ratio Function

```
def red_ratio(region):
    red1 = cv2.inRange(region, red_lower1, red_upper1)
    red2 = cv2.inRange(region, red_lower2, red_upper2)
    return np.sum(red1 | red2) / 255
```

- Applies both red masks to the region (top/bottom).
- Counts how many pixels are red (non-zero in the mask).
- Returns a total count (divided by 255 because masks are binary images).

White Ratio Function

```
def white_ratio(region):
    white = cv2.inRange(region, white_lower, white_upper)
    return np.sum(white) / 255
```

Extracting Color Proportions

```
top_red = red_ratio(top_half)
top_white = white_ratio(top_half)
bottom_red = red_ratio(bottom_half)
bottom_white = white_ratio(bottom_half)
```

- Measures **how much red/white** is in **top** and **bottom halves**.

Final :

```
if top_red > bottom_red and bottom_white > top_white:
    return "Indonesia"
elif top_white > bottom_white and bottom_red > top_red:
    return "Poland"
else:
    return "Unable to classify confidently"
```