Objective

- The objective of this assignment is to develop a solution using image processing techniques to partially de-annotate an image, given an original image and a fully annotated image. The solution should be able to de-annotate the image partially without re-annotating the original image. The assignment will also provide a set of sample images containing original, fully annotated, and partially annotated images for testing and evaluation purposes.The solution should be able to identify and remove specific annotations from the fully annotated image while preserving the underlying information of the original image. The partially de-annotated image should not contain any unwanted annotations or distortions that may affect its interpretation. Additionally, the solution should be able to generalize to different types of images and annotations, and should be efficient and scalable for processing large datasets.

Importing necessary libraries

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

Data

https://github.com/akaiketech/internship-assignment-cv/tree/main/Dataset

Solution

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




def get_output_image(original_image_path: str,
fully_annotated_image_path: str,
                     partially_annotated_image_path: str):



    original_image = cv2.imread(original_image_path)
    fully_annotated_image = cv2.imread(fully_annotated_image_path)


    hsv_image = cv2.cvtColor(fully_annotated_image, cv2.COLOR_BGR2HSV)
    lower_red = np.array([0, 100, 100], dtype=np.uint8)
    upper_red = np.array([10, 255, 255], dtype=np.uint8)
    mask = cv2.inRange(hsv_image, lower_red, upper_red)
    result = original_image.copy()
    result[np.where(mask == 255)] = fully_annotated_image[np.where(mask
== 255)]


    cv2.imwrite(partially_annotated_image_path, result)


    fig, axs = plt.subplots(1, 3, figsize=(12, 4))
    axs[0].imshow(cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB))
    axs[0].set_title("Original Image")
    axs[0].axis('off')
    axs[1].imshow(cv2.cvtColor(fully_annotated_image,
cv2.COLOR_BGR2RGB))
    axs[1].set_title("Fully Annotated Image")
    axs[1].axis('off')
    axs[2].imshow(cv2.cvtColor(result, cv2.COLOR_BGR2RGB))
    axs[2].set_title("Partially Annotated Image")
    axs[2].axis('off')
    plt.tight_layout()
    plt.show()


# In[3]:


# Set the path to the dataset folder
dataset_folder = 'H:\Dataset'

# Process each folder in the dataset
for folder_name in os.listdir(dataset_folder):
    folder_path = os.path.join(dataset_folder, folder_name)

    original_image_path =
os.path.join(folder_path,'original_image.jpg')
    fully_annotated_image_path =
os.path.join(folder_path,'fully_annotated_image.jpg')
    partially_annotated_image_path =
os.path.join(folder_path,'partially_annotated_image.jpg')

    if os.path.isfile(original_image_path) and
os.path.isfile(fully_annotated_image_path):

        if os.path.isfile(partially_annotated_image_path):

            get_output_image(original_image_path,
fully_annotated_image_path, partially_annotated_image_path)
        else:

            partially_annotated_image_path = os.path.join(folder_path,
'partially_annotated_image_generated.jpg')
            get_output_image(original_image_path,
fully_annotated_image_path, partially_annotated_image_path)
```