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

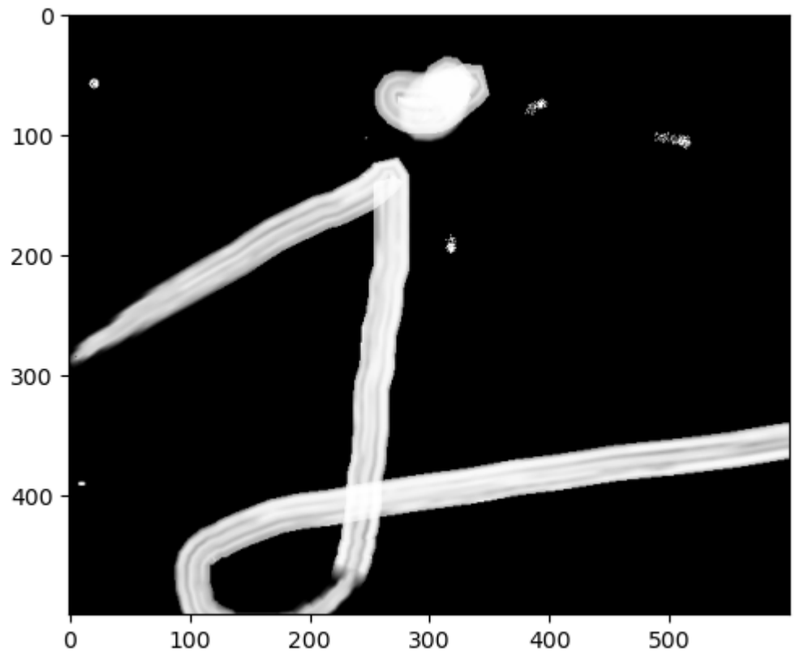
Bắt đầu lập trình hoặc tạo mã bằng trí tuệ nhân tạo (AI).

```
kernel = np.ones((5,5),np.uint8)
print(kernel)
```

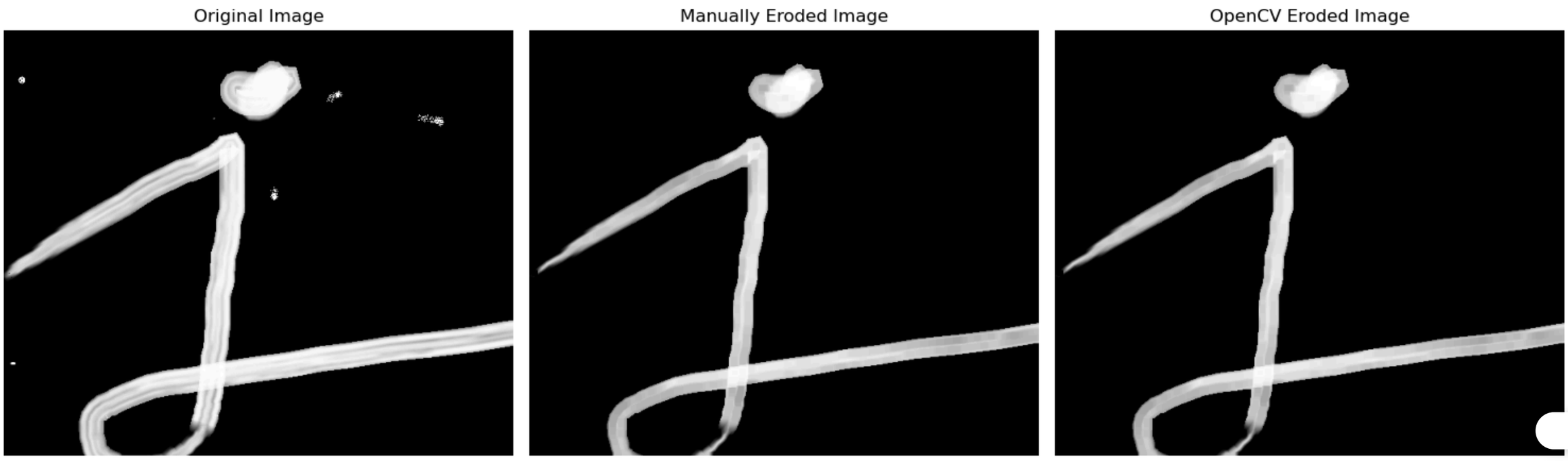
```
[[1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]]
```

```
import numpy as np

def erosion(img: np.ndarray, kernel: np.ndarray, iterations: int = 1) -> np.ndarray:


    img_h, img_w = img.shape

    k_h, k_w = kernel.shape


    pad_h = k_h // 2
    pad_w = k_w // 2

    padded_img = np.pad(img, ((pad_h, pad_h), (pad_w, pad_w)), mode='constant', constant_values=255)

    eroded_img = img.copy()

    for _ in range(iterations):
        temp_img = np.zeros_like(img)

        for i in range(img_h):
            for j in range(img_w):
                roi = padded_img[i:i + k_h, j:j + k_w]
                temp_img[i, j] = np.min(roi * kernel)

        eroded_img = temp_img
        padded_img = np.pad(eroded_img, ((pad_h, pad_h), (pad_w, pad_w)), mode='constant', constant_values=255)

    return eroded_img



def dilation(image: np.ndarray, kernel: np.ndarray, iterations: int = 1) -> np.ndarray:
    h, w = image.shape
    k_h, k_w = kernel.shape
    pad_h, pad_w = k_h // 2, k_w // 2

    padded_img = np.pad(image, ((pad_h, pad_w)), mode='constant', constant_values=0)
    temp_img = np.zeros_like(image)

    for i in range(h):
        for j in range(w):
            roi = padded_img[i:i + k_h, j:j + k_w]
            temp_img[i, j] = np.max(roi * kernel)  # Tính giá trị lớn nhất trong vùng ROI

    return temp_img


def opening(image: np.ndarray, kernel: np.ndarray, iterations: int = 1) -> np.ndarray:
    # Bước 1: Erosion
    eroded_image = erosion(image, kernel, iterations)

    # Bước 2: Dilation
    opened_image = dilation(eroded_image, kernel, iterations)

    return opened_image


def display_images(original, eroded_manual, eroded_cv2):
    plt.figure(figsize=(15, 5))

    plt.subplot(1, 3, 1)
    plt.imshow(original, cmap='gray')
    plt.title('Original Image')
    plt.axis('off')

    plt.subplot(1, 3, 2)
    plt.imshow(eroded_manual, cmap='gray')
    plt.title('Manually Eroded Image')
    plt.axis('off')

    plt.subplot(1, 3, 3)
    plt.imshow(eroded_cv2, cmap='gray')
    plt.title('OpenCV Eroded Image')
    plt.axis('off')

    plt.tight_layout()
    plt.show()


img = cv2.imread("SOI.png")
img_crop = img[0:500,300:900]
img_gray = cv2.cvtColor(src = img_crop,code =cv2.COLOR_BGR2GRAY)
plt.imshow(img_gray,cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x253382f5e50>
```

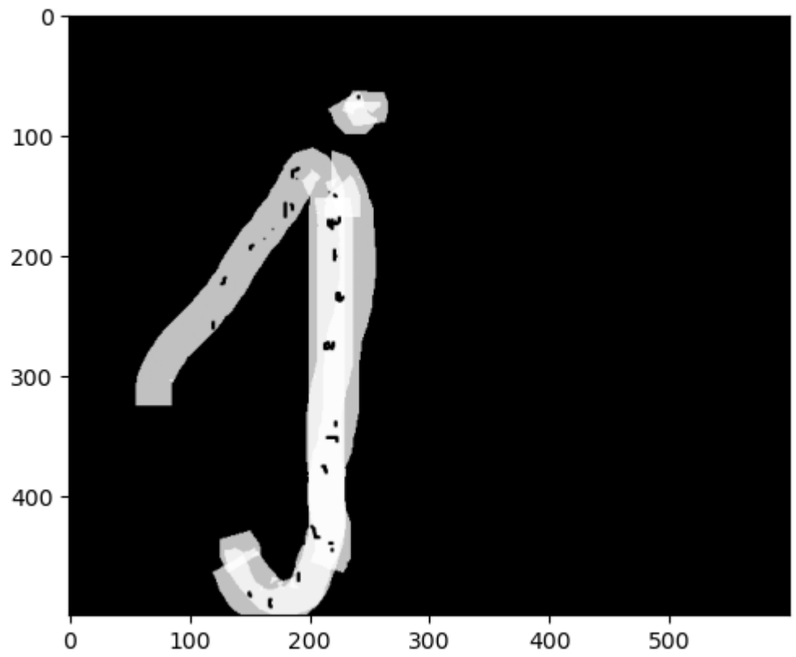



```
kernel_size = 3
kernel = np.ones((kernel_size, kernel_size), np.uint8)
eroded_manual = erosion(img_gray, kernel,iterations=3)


eroded_cv2 = cv2.erode(img_gray, kernel,iterations=3)

# Display both images
display_images(img_gray, eroded_manual, eroded_cv2)
```

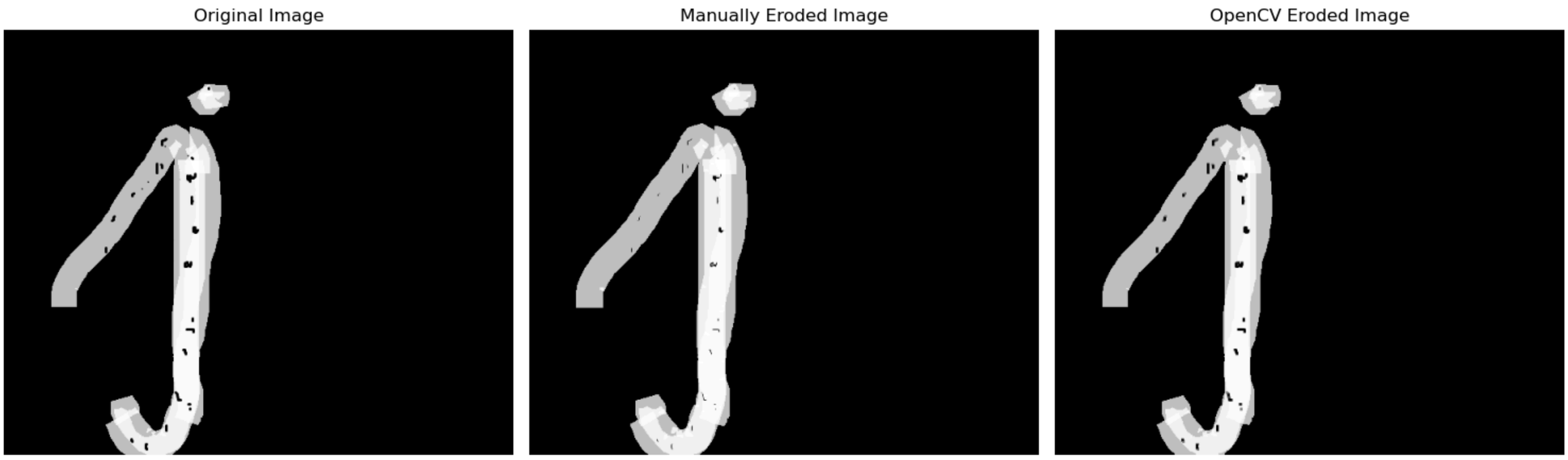



```
img = cv2.imread("a.png")
img_crop = img[0:500,300:900]
img_gray = cv2.cvtColor(src = img_crop,code =cv2.COLOR_BGR2GRAY)
plt.imshow(img_gray,cmap = 'gray')
```

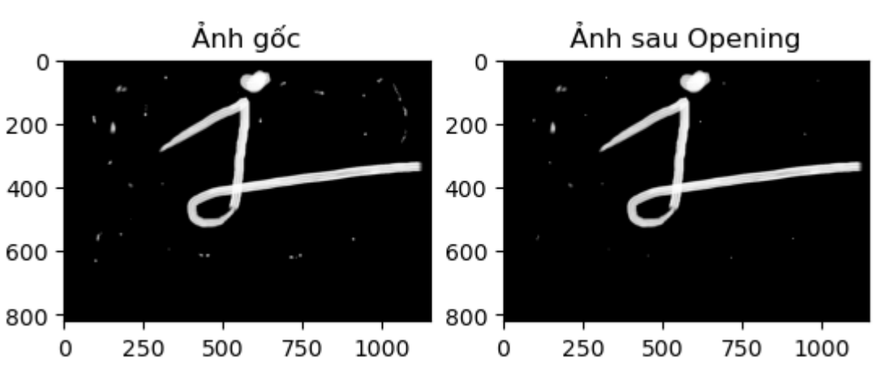
```
<matplotlib.image.AxesImage at 0x2533ab4d3d0>
```



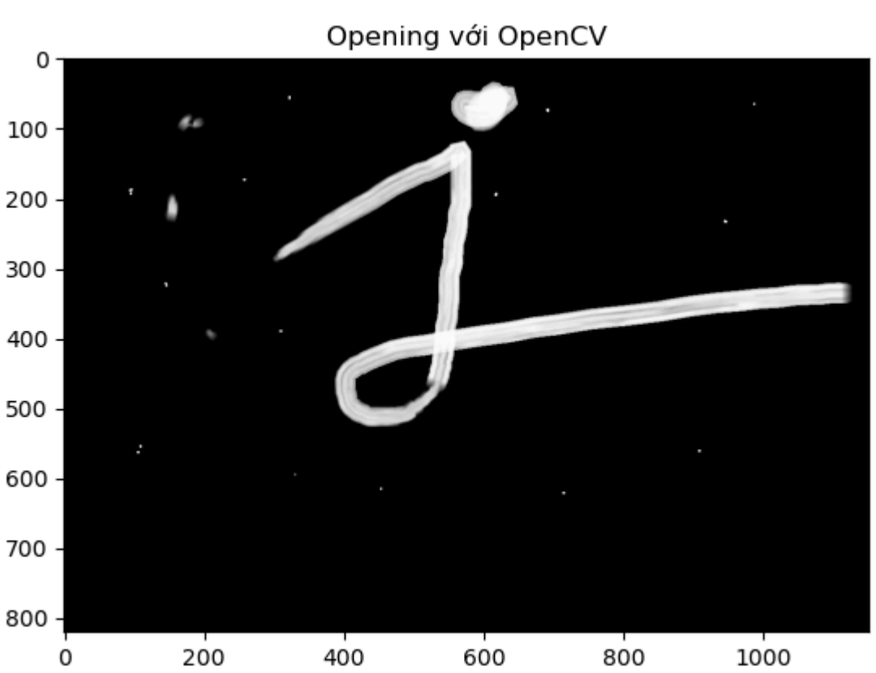

```
kernel_size = 3
kernel = np.ones((kernel_size, kernel_size), np.uint8)
dilation_manual = dilation(img_gray, kernel,iterations=3)


closed_cv2 = cv2.morphologyEx(img_gray, cv2.MORPH_CLOSE, kernel)

# Display both images
display_images(img_gray, dilation_manual, closed_cv2)
```




```
image = cv2.imread('soi.png', cv2.IMREAD_GRAYSCALE)

# Tạo kernel 3x3
kernel = np.ones((3, 3), dtype=np.uint8)

# Thực hiện Opening với thuật toán tự cài đặt
opened_image = opening(image, kernel)

# Hiển thị kết quả
plt.subplot(1, 2, 1)
plt.imshow(image, cmap='gray')
plt.title('Ảnh gốc')

plt.subplot(1, 2, 2)
plt.imshow(opened_image, cmap='gray')
plt.title('Ảnh sau Opening')

plt.show()
```




```
opened_cv = cv2.morphologyEx(image, cv2.MORPH_OPEN, kernel)

# Hiển thị kết quả OpenCV
plt.imshow(opened_cv, cmap='gray')
plt.title('Opening với OpenCV')
plt.show()
```




```
file_paths = ['miniPrj/a.png', 'miniPrj/b.png', 'miniPrj/c.png', 'miniPrj/d.png']

# Danh sách chứa các ảnh đã đọc
images = []

# Đọc tất cả các file ảnh
for file in file_paths:
    img = cv2.imread(file, cv2.IMREAD_COLOR)  # Đọc ảnh với chế độ màu (BGR)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # Chuyển từ BGR sang RGB để hiển thị đúng màu
    images.append(img)  # Thêm ảnh vào danh sách

# Hiển thị tất cả các ảnh đã đọc
plt.figure(figsize=(10,10))

for i, img in enumerate(images):
    plt.subplot(2, 2, i+1)  # Tạo grid 2x2 để hiển thị 4 ảnh
    plt.imshow(img)
    plt.title(f'Image {i+1}')
    plt.axis('off')

plt.tight_layout()
plt.show()
```

Image 1

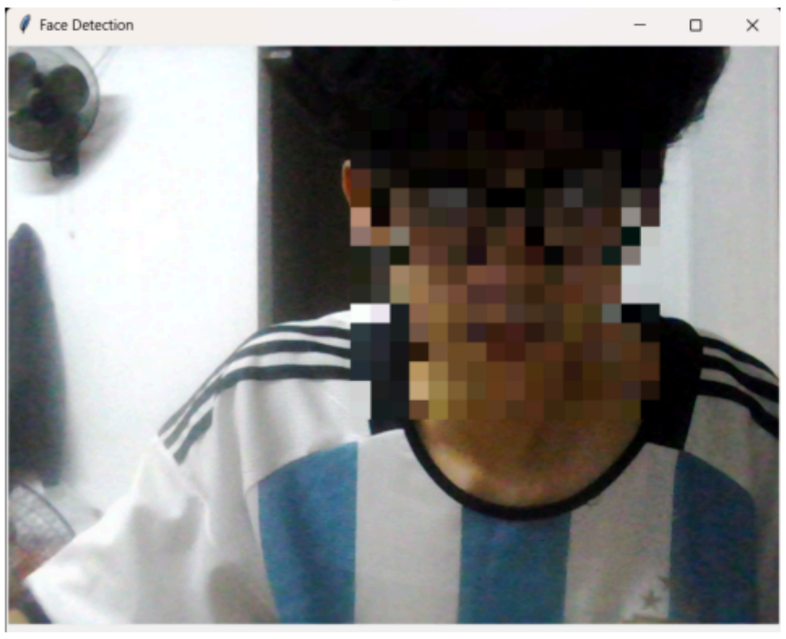


Image 2

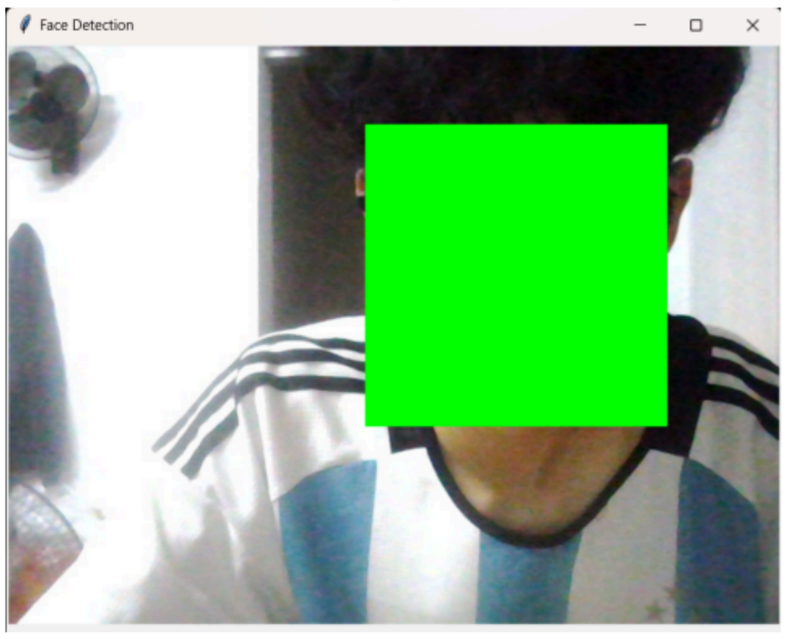


Image 3

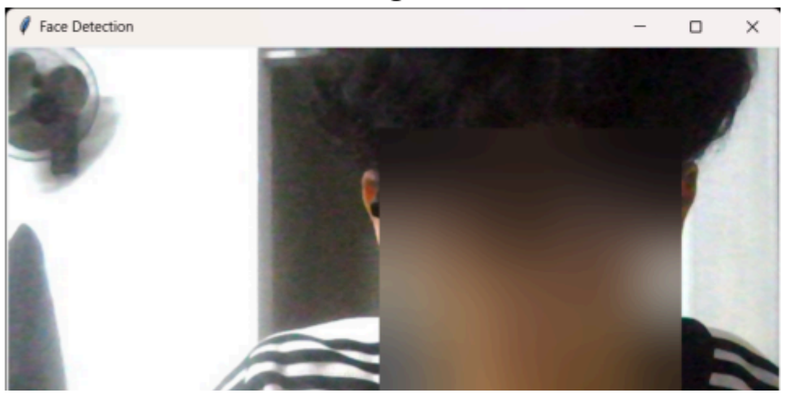


Image 4