ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO BÀI TẬP LỚN CUỐI KÌ

MÔN: LẬP TRÌNH XỬ LÍ TÍN HIỆU SỐ (DIGITAL SIGNAL PROCESSING)

DỰ ÁN:

XÂY DỰNG ỨNG DỤNG XỬ LÍ ẢNH BẰNG PYTHON

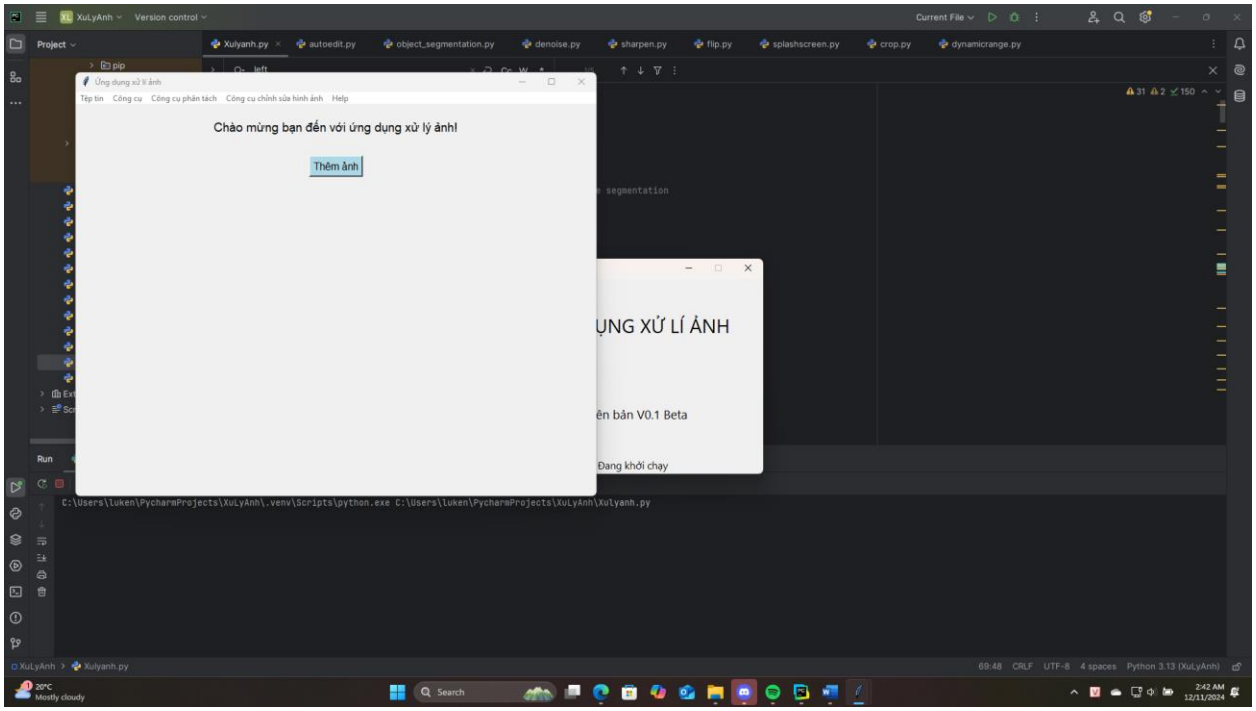
HỌ VÀ TÊN: NGUYỄN MINH HOÀNG
MÃ SINH VIÊN: 21021591
LỚP KHÓA HỌC: ELT3246_59

ỨNG DỤNG XỬ LÍ HÌNH ẢNH

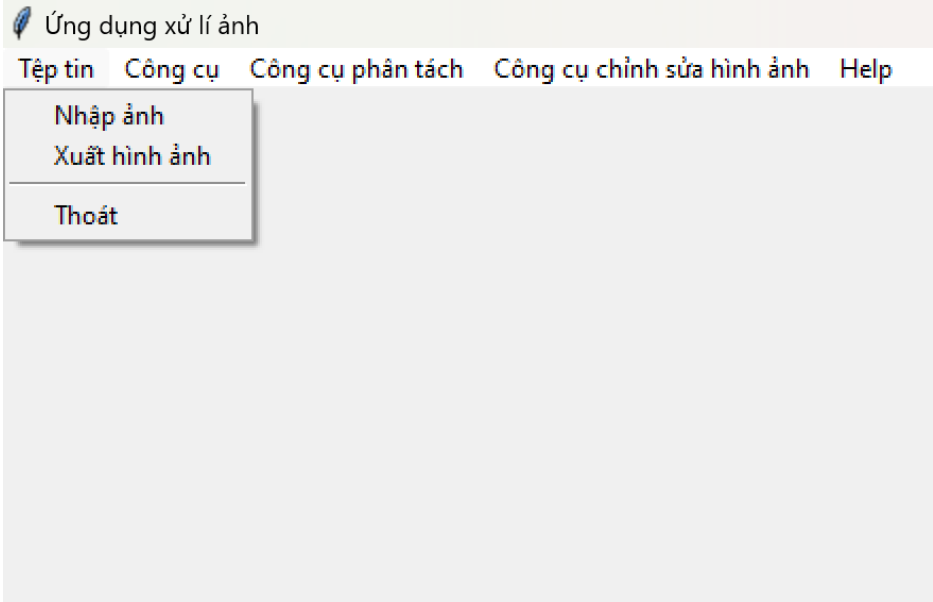
a. ***Phát triển***

- ***Phát triển trên nền Python3, sử dụng Pycharm Professional và hàm Tkinter và CV2.***

b. ***Giao diện chính:***



Giao diện chính của ứng dụng sẽ bao gồm một hình cửa sổ gốc, một nút thêm ảnh và các bộ công cụ ở phía trên.



a. Nhập hình ảnh

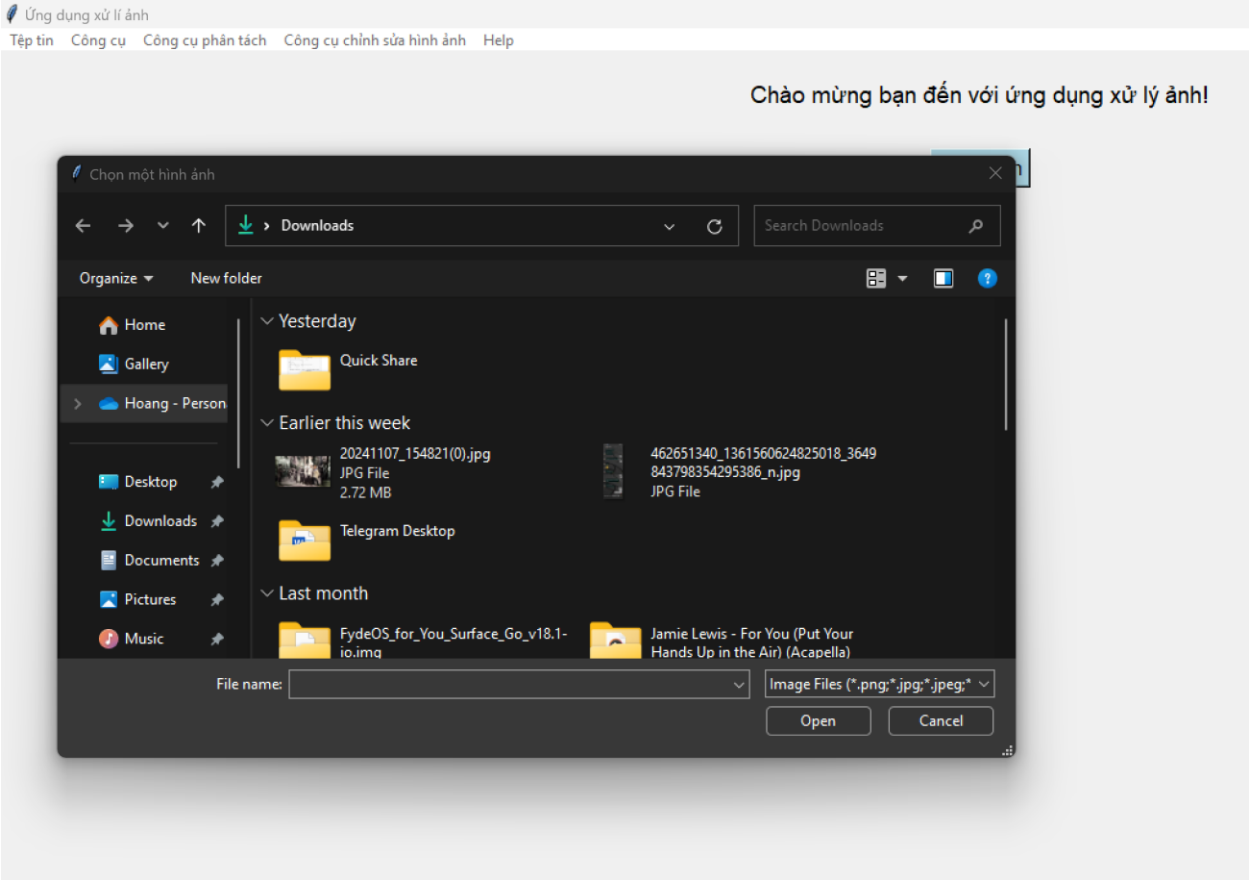
Phần tệp tin ở phía bên trên sẽ sử dụng nền tảng nhập file từ Explorer ở Windows thông qua thư viện PIL và thư viện thiết kế tkinter gồm:

```
import tkinter as tk
from tkinter import Menu, filedialog, messagebox, simpledialog
from PIL import Image, ImageTk
```

Chúng ta thêm thư viện tkinter thông qua câu lệnh pip install tkinter và hàm nhập file sẽ là:

```
file_path = filedialog.askopenfilename(
    filetypes=[("Image Files", "*.png;*.jpg;*.jpeg;*.bmp;*.gif")],
    title="Chọn một hình ảnh"
)
```

Nó sẽ hiện ra như sau khi chúng ta bấm vào nút Thêm ảnh hoặc Tệp tin > Nhập ảnh:



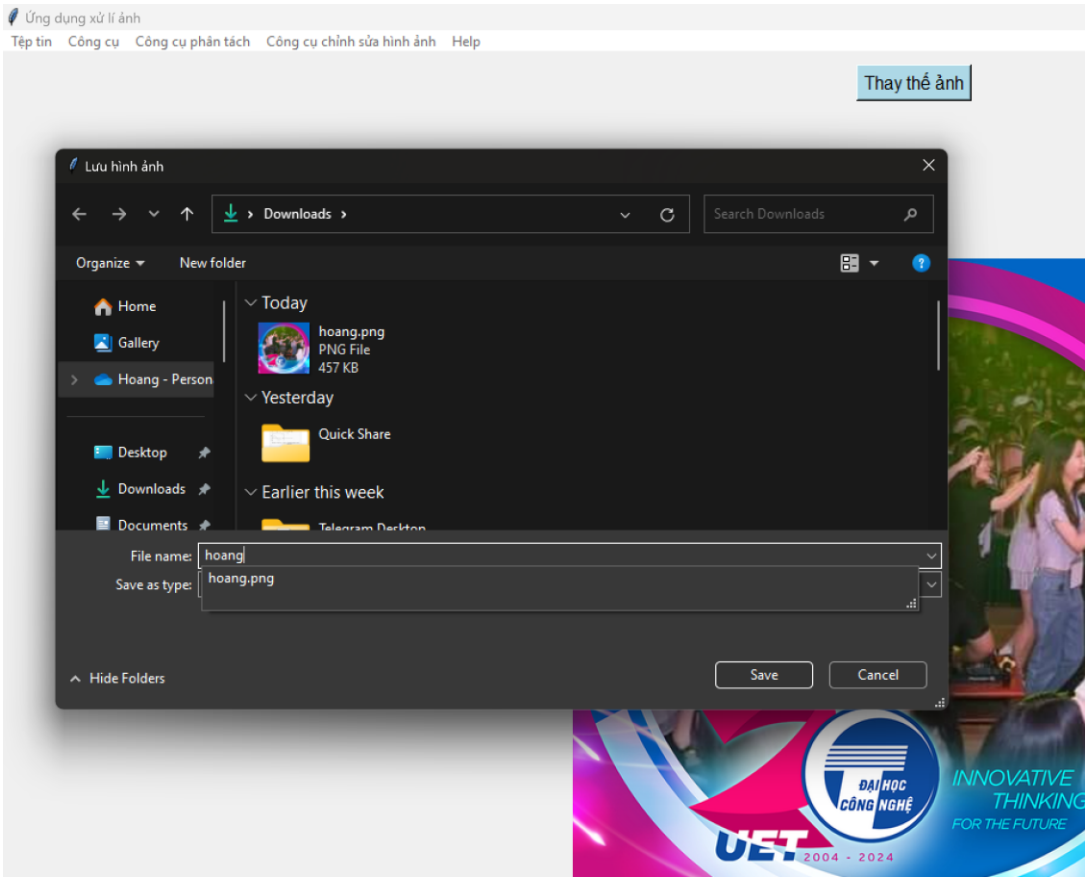
Khi đó thì nút thêm ảnh bên trên sẽ được thay thế bằng nút thay thế ảnh, để mọi người có thể thay đổi ảnh muốn chỉnh sửa. Tuy nhiên nếu để ảnh đúng độ phân giải thì ảnh sẽ bị chèn quá mức, nên em sẽ để nó hiển thị theo scale factor dựa trên kích thước cửa sổ với công thức:

$$Kích\ thước\ ảnh = kích\ thước\ gốc * tỉ\ lệ = kích\ thước\ gốc * \frac{kích\ thước\ cửa\ sổ}{kích\ thước\ ảnh}$$

b. Xuất hình ảnh:

```
def export_image():  1 usage
    global current_image
    if current_image is None:
        messagebox.showwarning( title: "Warning",  message: "Không có hình ảnh nào để xuất!")
        return

    # Open file dialog to save the image
    file_path = filedialog.asksaveasfilename(
        defaultextension=".png",
        filetypes=[("PNG Files", "*.png"), ("JPEG Files", "*.jpg;*.jpeg"), ("Bitmap Files", "*.bmp")],
        title="Lưu hình ảnh"
    )
```

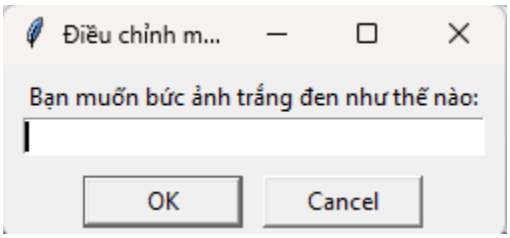
Nó cũng sẽ hiện ra cửa sổ cho chúng ta lưu hình ảnh vào vị trí bất kì trong File Explorer.



c. <u>Làm việc với ảnh:</u>

a. Đen trắng:

Chế độ này đưa ảnh về chế độ đen trắng, và chúng ta có thể điều chỉnh mức độ đen trắng khi cần thiết. Ứng dụng sẽ hỏi trực tiếp khi chúng ta điều chỉnh:



Con số được nhập trong hàm này sẽ được tính theo tỉ lệ phần trăm. Con số càng cao thì ảnh càng sáng.

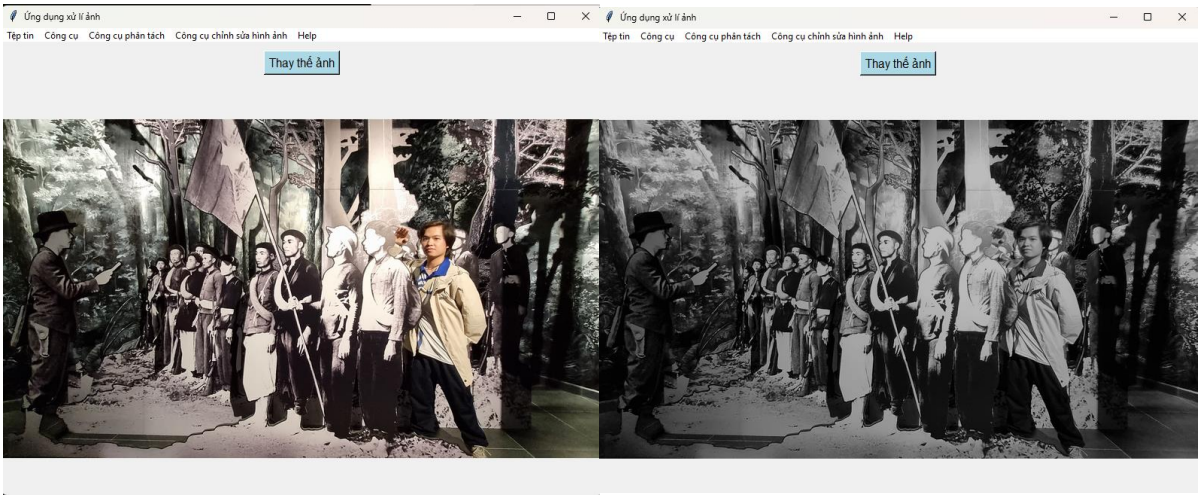


Figure 1: Ảnh trước ban đầu và ảnh chỉ còn 75% độ sáng với trắng đen.

Hàm thực hiện trong trường hợp này đó là grayscale.py, gồm hàm chuyển grayscale và hàm điều chỉnh mức độ ánh sáng trong grayscale đó khi nó được để sáng theo tỉ lệ phần trăm.

```
def adjust_grayscale(image, intensity_percentage):
    """
    Adjust the grayscale intensity based on a given percentage.
    0% = completely black, 100% = original grayscale.
    """
    try:
        # Convert image to grayscale first
        grayscale_image = convert_to_grayscale(image)

        # Use ImageEnhance to adjust brightness
        enhancer = ImageEnhance.Brightness(grayscale_image)

        # Convert the intensity percentage to a factor (0 to 1)
        intensity_factor = intensity_percentage / 100.0

        # Enhance the image based on intensity factor
        adjusted_image = enhancer.enhance(intensity_factor)

        return adjusted_image
    except Exception as e:
        raise e
```

b. Siêu Phân giải: Super Resolution.

Chế độ siêu phân giải là chế độ giúp tăng cường độ sắc nét của ảnh bằng cách làm nét hóa và tăng kích thước của ảnh bằng cách tăng độ phân giải thực tế. Trong trường hợp này thì em sẽ chỉ tăng kích thước do giới hạn của PIP và Pycharm, thông qua thư viện PIL:

```
from PIL import Image, ImageEnhance, ImageFilter
```

```python
def upscale_image(image, factor):
    global current_image
    current_image = image
    try:
        # Upscale the image
        width, height = image.size
        new_width = int(width * factor)
        new_height = int(height * factor)

        # Resize the image
        upscale_image = image.resize((new_width, new_height), Image.LANCZOS)

        # Apply sharpening filter
        upscale_image = upscale_image.filter(ImageFilter.SHARPEN)

        return upscale_image
    except Exception as e:
        raise e
```

Tuân theo công thức được ghi trong hàm là kích thước mới = kích thước cũ * hệ số. Tuy nhiên kích thước cũ trong chương trình là kích thước đang hiển thị sau khi được scale xuống theo cửa sổ.

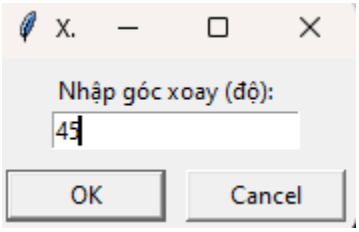
c. Xoay ảnh

Chức năng xoay ảnh sẽ sử dụng hàm xoay ảnh theo góc nhất định trong PIL, với mục đích xoay ngang, dọc chéo với góc tương ứng. Nếu như chiều dài chiều rộng thay đổi, thì sẽ có một khoảng đen tương ứng bù vào, để đảm bảo ảnh nó sẽ vẫn hiển thị đầy đủ.

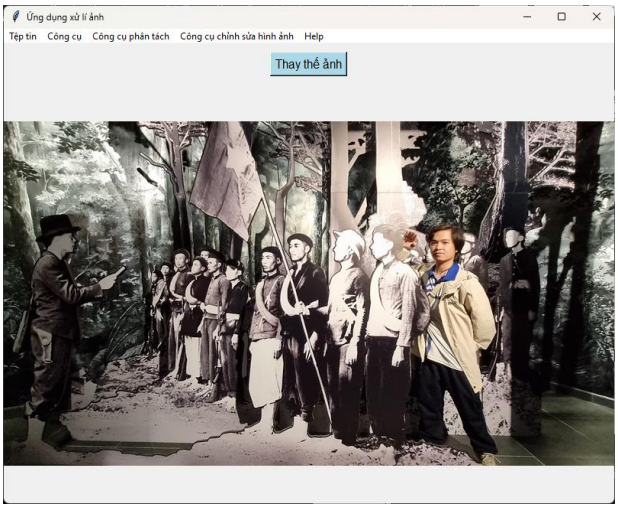
Hàm xoay ảnh trong PIL:

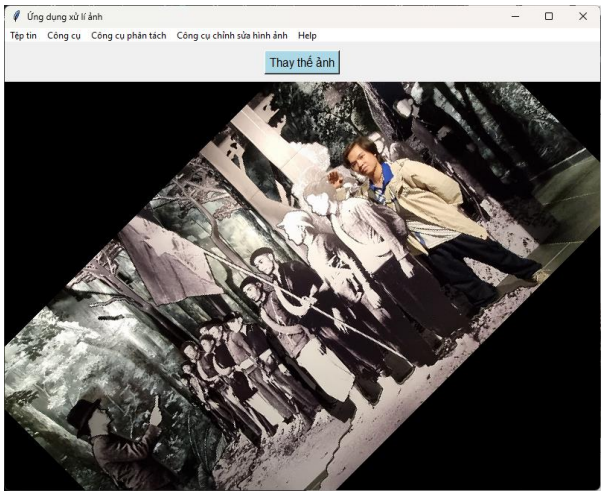
```python
from PIL import Image

def rotate_image(image, angle):
    """
    Rotate the image by the specified angle.
    """
    try:
        # Rotate the image by the specified angle
        rotated_image = image.rotate(angle, expand=True)  # expand=True
ensures the whole image fits in the frame
        return rotated_image
    except Exception as e:
        raise e
```

Trong đó angle là góc xoay.



Từ đó chúng ta có:





d. Tăng HDR:

Tăng HDR hay tăng dải nhạy sáng là kĩ thuật giúp ảnh từ tối đến sáng. Trong phần mềm thì dải nhạy sáng sẽ được tăng từ 0 đến 5. Nó sẽ để mức mặc định là 2, bằng cách chuyển bảng màu của ảnh về dạng RGB, sau đó sẽ đẩy mức độ sắc nét của ảnh để tăng chi tiết, và đẩy mức độ tương phản lên để tăng HDR.

```
def enhance_hdr(image, strength=2.0):


    try:
        # Convert to RGB (if not already in that mode)
        image = image.convert("RGB")

        # Apply enhancement to simulate HDR effect
        enhancer = ImageEnhance.Contrast(image)
        enhanced_image = enhancer.enhance(strength)  # Increase contrast for
HDR look

        # Optionally apply some sharpening to simulate HDR clarity
        enhanced_image = enhanced_image.filter(ImageFilter.SHARPEN)

        return enhanced_image
    except Exception as e:
        raise e
```

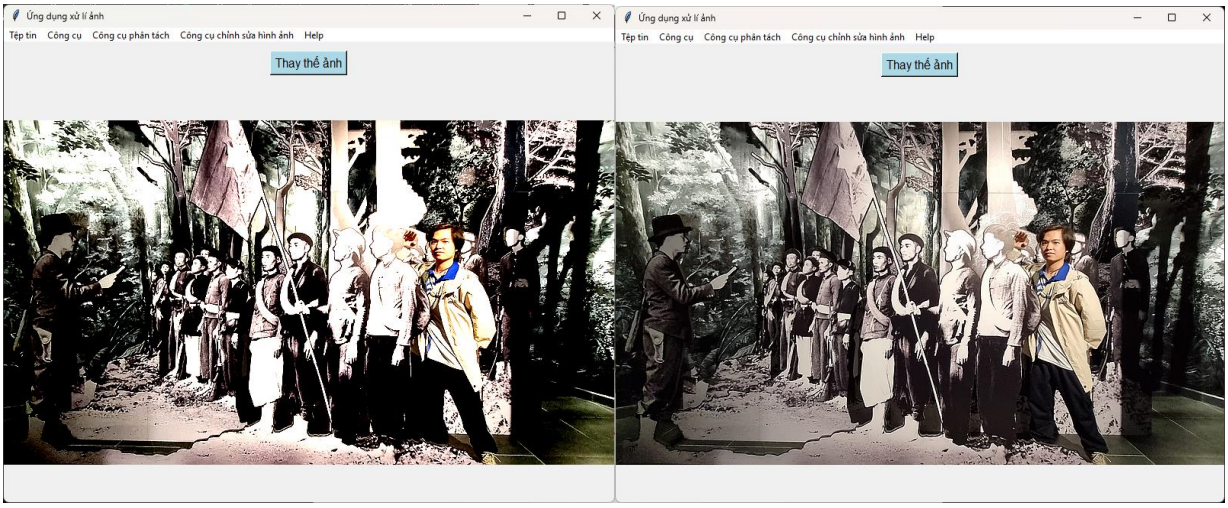


Figure: Hình ảnh khi tăng HDR lên 3 và ban đầu

e. Làm nét ảnh:

Làm nét ảnh (Sharpening) sẽ sử dụng bộ lọc của PIL và tăng cường độ nét thông qua hàm sharpness trong hàm image.filter (là bộ lọc làm nét) và sau đó sẽ điều chỉnh cường độ nét bằng image enhance

```
"""
Làm nét ảnh bằng cách sử dụng bộ lọc của PIL và tăng cường độ nét.

Args:
    image (PIL.Image.Image): Ảnh đầu vào (PIL Image).
    sharpness_level (float): Mức độ làm nét (mặc định = 2.0).

Returns:
    PIL.Image.Image: Ảnh đã được làm nét.
"""
```

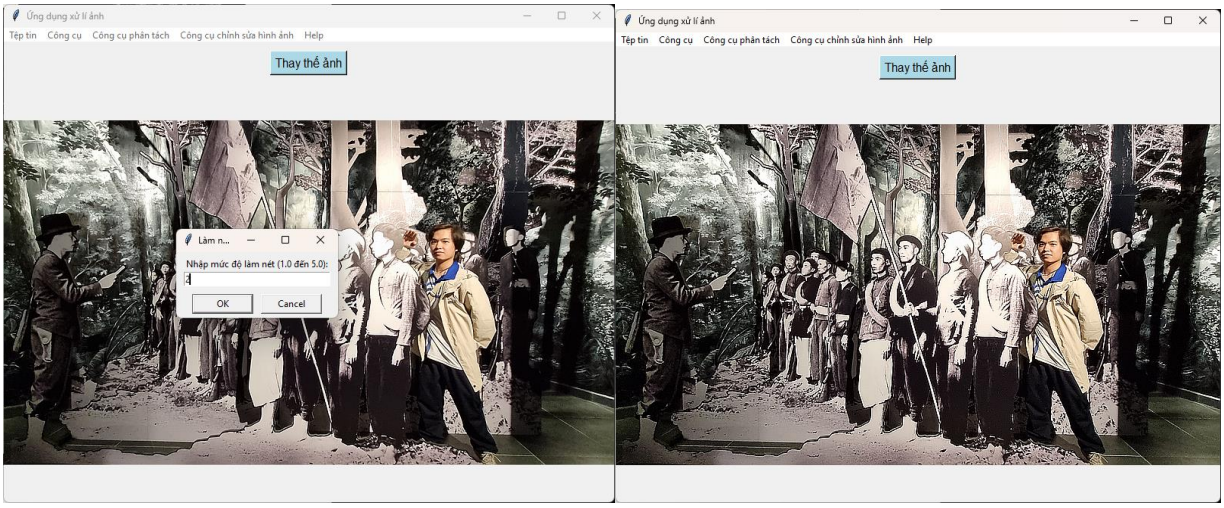


Figure: Tăng độ nét lên 2 và kết quả

f. Khử nhiễu và làm mờ ảnh bằng Gaussian Blur

Khử nhiễu ảnh bằng cách sử dụng bộ lọc Gaussian Blur để làm mờ nhiễu.

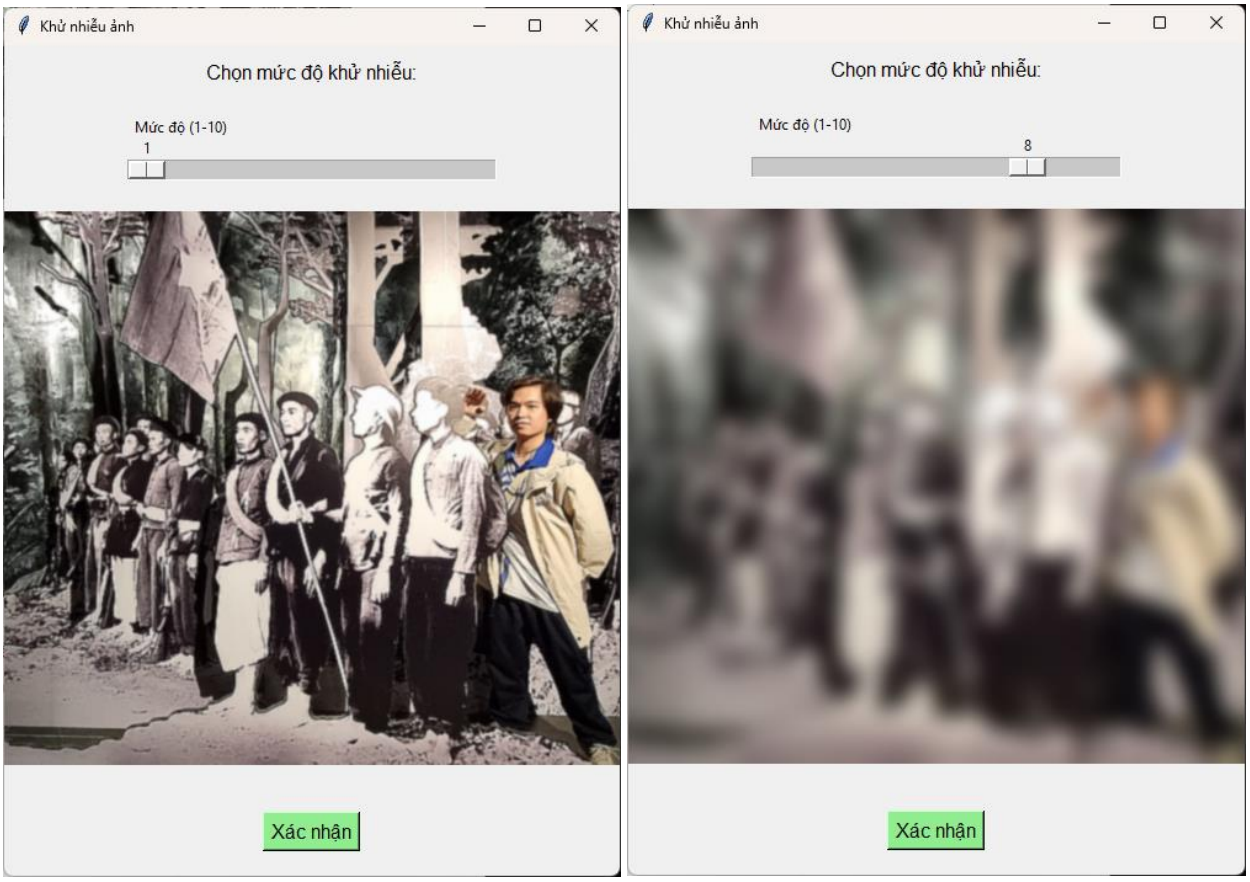
Args:

image (PIL.Image.Image): Ảnh đầu vào (PIL Image).

filter_strength (int): Độ mạnh của bộ lọc (mặc định = 2).

Returns:

PIL.Image.Image: Ảnh đã được khử nhiễu.



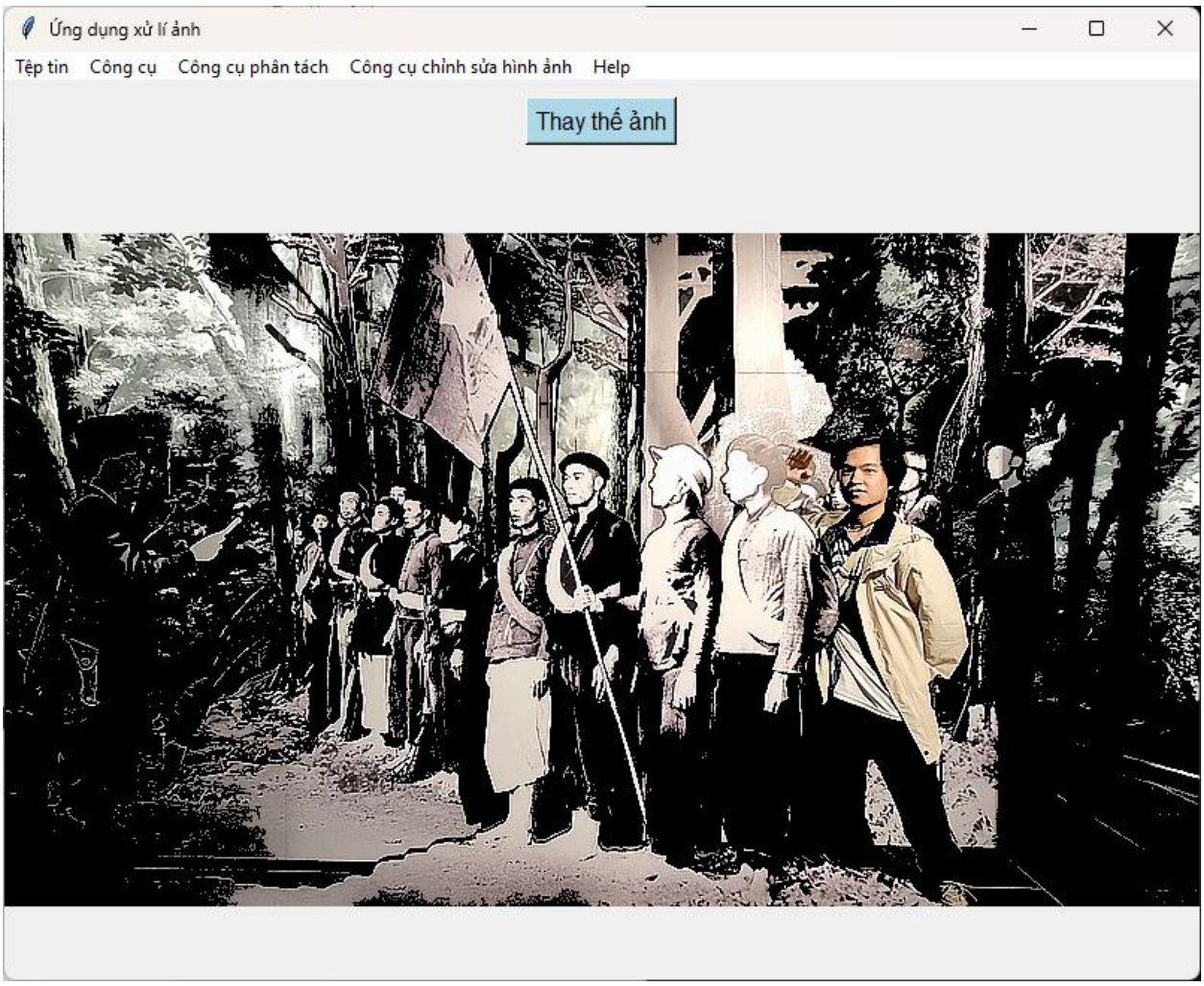
g. Phân tách ảnh thông qua thư viện CV2 và Threadshold

d. Tham số:
image: Đối tượng hình ảnh PIL
block_size: Kích thước của vùng lân cận điểm ảnh được sử dụng để tính toán giá trị ngưỡng.
C: Hằng số bị trừ khỏi giá trị trung bình hoặc trung bình có trọng số.

e. Trả về:
Đối tượng hình ảnh PIL đã phân đoạn.

Phân tách vật thể thông qua thư viện Cv2, Threadshold và đi lại viền:

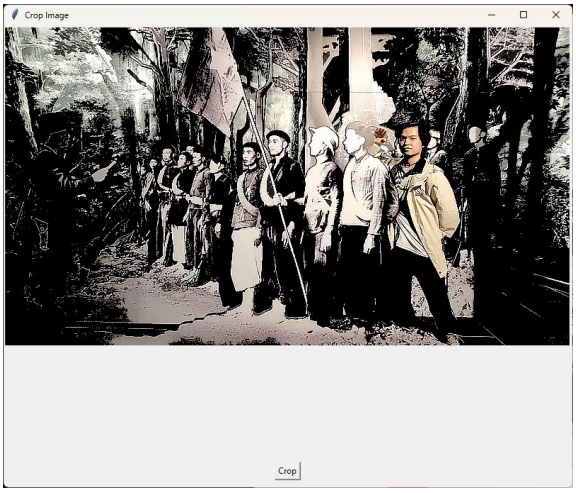
```
import cv2
from PIL import Image
import numpy as np

def segment_objects(image):
    # Chuyển đổi từ PIL Image sang định dạng OpenCV
    open_cv_image = cv2.cvtColor(np.array(image), cv2.COLOR_RGB2BGR)

    # Chuyển ảnh sang grayscale
    gray = cv2.cvtColor(open_cv_image, cv2.COLOR_BGR2GRAY)

    # Áp dụng threshold để tách vật thể
    _, binary = cv2.threshold(gray, 128, 255, cv2.THRESH_BINARY)

    # Tìm đường viền
    contours, _ = cv2.findContours(binary, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    # Tạo mask
    mask = np.zeros_like(gray)
    cv2.drawContours(mask, contours, -1, 255, thickness=cv2.FILLED)

    # Áp dụng mask lên ảnh gốc
    segmented = cv2.bitwise_and(open_cv_image, open_cv_image, mask=mask)

    # Chuyển đổi ngược lại sang PIL Image
    return Image.fromarray(cv2.cvtColor(segmented, cv2.COLOR_BGR2RGB))
```

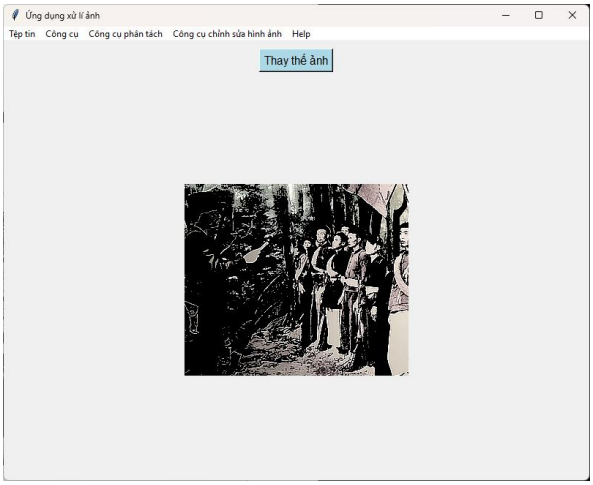


a. Cắt ảnh:

Phát triển tính năng cắt với GUI để có thể cắt 1 cách trực quan hơn.

```
"""
Crop an image to the specified rectangle.

Parameters:
    image: PIL Image object.
    left: The x-coordinate of the left boundary.
    top: The y-coordinate of the top boundary.
    right: The x-coordinate of the right boundary.
    bottom: The y-coordinate of the bottom boundary.

Returns:
    Cropped PIL Image object.
"""
```





b. *Lật ảnh:*

```
"""
Lật ảnh theo chiều ngang hoặc dọc.
direction: "horizontal" (ngang) hoặc "vertical" (dọc)
current_image: PIL.Image object
"""
```