Report On

Image to Sketch Generator

Submitted in partial fulfillment of the requirements of the Course project in
Semester IV of Second Year Artificial Intelligence and Data Science

By
Rohan Ganpatrao Mangaonkar(27)
Vaibhav Sopan Narute(33)
Dnyanesh Baburao Panchal (34)
Krithik Devendra Pandey (35)

Supervisor
Prof. Kshitija Gharat

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science

(2023-24)

Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence and Data Science

CERTIFICATE

This is to certify that the project entitled “**Image to Sketch generator**” is a bonafide work of "Dnyanesh Baburao Panchal (34), Krithik Devendra Pandey (35), Vaibhav Sopan Narute(33),Rohan Ganpatrao Mangaonkar(27)," submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester IV of Second Year Artificial Intelligence and Data Science engineering.

Supervisor

Prof. Kshitija M. Gharat

Dr. Tatwadarshi P. N.
Head of Department

Title: Image to Sketch Generator using Python

PROBLEM STATEMENT

For the following reasons, there is a demand for something to get the users a sketch directly through an image:

1. Practical Applications: Image-to-sketch conversion finds applications in various fields such as art, design, digital image editing, and forensics. Understanding and implementing effective techniques for this transformation can enhance the capabilities of software tools used in these domains.

2. Artistic Expression: Many artists and designers use sketching as a medium for creative expression. Digital tools that can convert images into sketch-like representations offer new avenues for artistic exploration and experimentation.

3. Research and Development: Advancements in image processing and computer vision techniques continuously evolve. Conducting a comparative study of different methodologies for image-to-sketch conversion helps researchers and developers understand the strengths and limitations of existing approaches, leading to further innovation in the field.

4. Automation and Efficiency: In industries such as architecture and fashion design, where sketches play a crucial role in conceptualization, automating the process of generating sketches from images can significantly improve workflow efficiency and productivity.

5. Educational Purposes: Image-to-sketch conversion serves as a valuable educational tool for teaching image processing concepts, such as edge detection, filtering, and feature extraction. A study on this topic can provide educators with practical examples and insights to enhance their teaching materials.

6. User Experience Enhancement: In applications like photo editing software and mobile apps, offering image-to-sketch conversion features can enhance the user experience by providing users with creative tools to transform their photos into unique and visually appealing sketches.

7. Accessibility and Inclusivity: Simplifying the process of creating sketches from images using automated algorithms can make sketching more accessible to individuals with limited artistic skills or physical abilities, fostering inclusivity and democratizing creative expression.

In summary, conducting a study on image-to-sketch conversion using Python addresses a practical need for enhancing creative tools, advancing research in image processing, and improving user experiences across various domains.

Objective:

The objective of the proposed Python-based image to sketch conversion solution is to provide a user-friendly and efficient method for transforming regular images into sketch-like representations. By leveraging the capabilities of Python and relevant image processing libraries, the program aims to achieve the following objectives:

1. Simplify the Conversion Process: Enable users to effortlessly convert their images into sketches without the need for complex manual editing or specialized software.

2. Enhance Creativity and Expression: Empower users to explore artistic and creative possibilities by generating sketch-like renditions of their photographs and images.

3. Facilitate User Interaction: Incorporate a graphical user interface (GUI) to streamline the selection of input images, initiation of the conversion process, and visualization of the resulting sketches.

Proposed Solution

The proposed solution involves developing a Python-based program for converting images into sketches. This program will utilize various image processing libraries such as OpenCV and NumPy to manipulate the image data and perform the conversion process.

The first step of the solution will involve loading the input image using OpenCV and then converting it into a grayscale representation. This grayscale image will serve as the basis for generating the sketch.

Next, the program will apply image processing techniques such as edge detection and thresholding to accentuate the prominent features and outlines within the grayscale image. This will help in creating a sketch-like effect by emphasizing the edges and details of the original image.

After the sketch conversion process is completed, the program will then save the resulting sketch image, allowing users to visualize the transformed output.

To enhance the user experience, the program may also incorporate a graphical user interface (GUI) using libraries such as Tkinter, enabling users to easily select input images, initiate the conversion process, and view the generated sketch.

Additionally, the program may offer customization options, allowing users to adjust parameters such as line thickness, contrast, and brightness to tailor the sketch output according to their preferences.

Finally, the program will provide documentation and examples to guide users on how to utilize the image to sketch conversion functionality within their own Python projects.

In conclusion, the proposed Python-based solution aims to provide a convenient and effective method for converting images into sketches through the use of image processing techniques. By leveraging the capabilities of Python and relevant libraries, users will be able to seamlessly transform their images into sketch-like representations, opening up creative possibilities and applications in various domains such as art, design, and image manipulation.

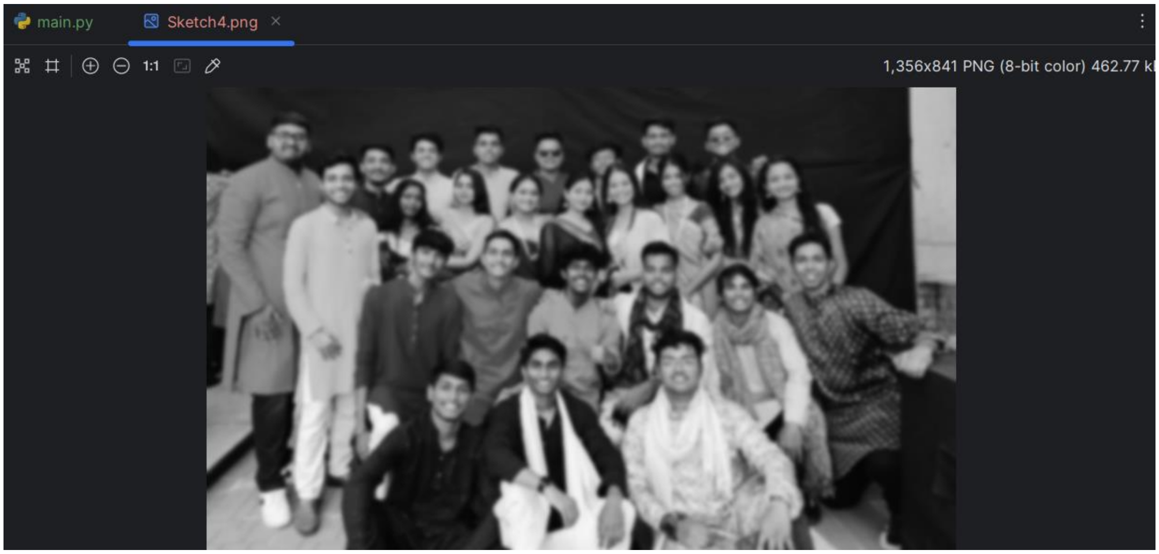
Hardware Requirements:

Computer: A desktop or laptop computer capable of running Python and image processing libraries.
Input Device: A device such as a keyboard or mouse to interact with the program and provide input, if a graphical user interface (GUI) is implemented.

Software Requirements:

Python: The Python programming language needs to be installed on the computer.
OpenCV Library: The program relies on the OpenCV library for image processing operations. Ensure that OpenCV is installed using pip or another package manager.
NumPy Library: NumPy is used for numerical computing and array operations. It should be installed alongside OpenCV.
Operating System: The program should be compatible with the operating system running on the computer, whether it's Windows, macOS, or Linux.
Overall, the hardware requirements are standard for running a computer program, while the software requirements include Python, OpenCV, NumPy, and a compatible operating system.

RESULTS



1. Reading the Image: The program reads the input image "bro.png" using the OpenCV library.

2. Converting the Image: The input image is converted into a grayscale image using the cvtColor function from the OpenCV library.

3. Inverting the Image: The grayscale image is inverted to create a negative using the bitwise_not function.

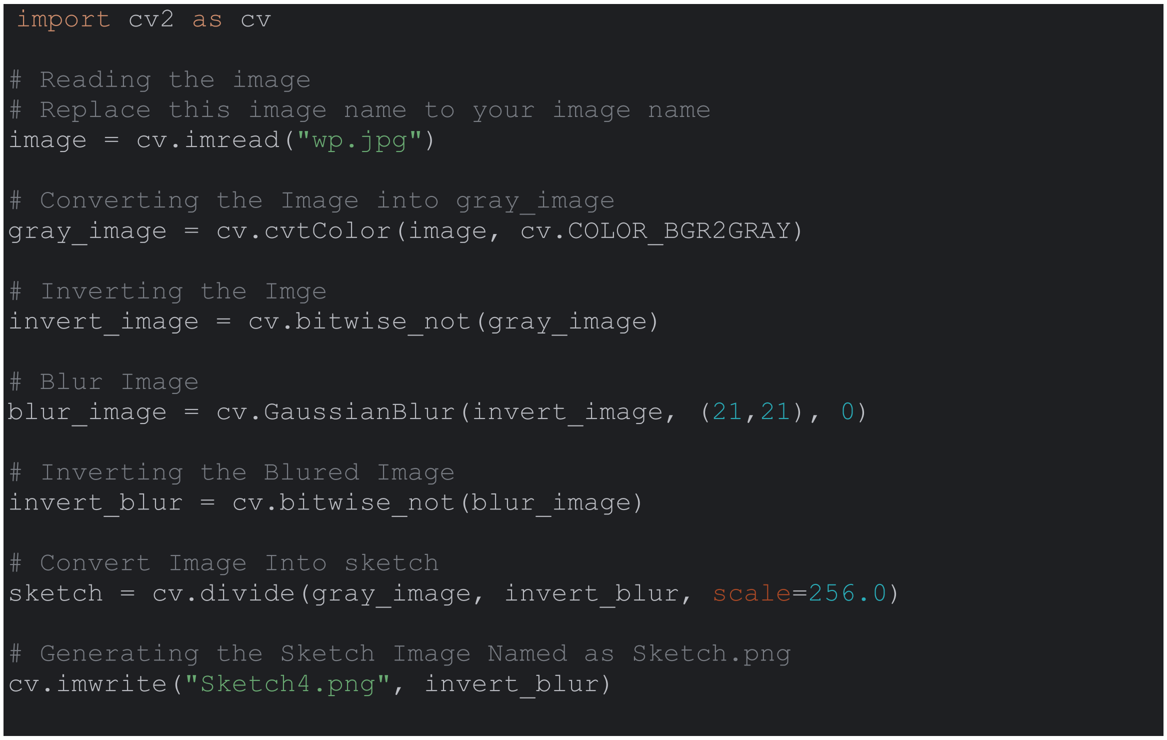
4. Blurring the Image: The inverted grayscale image is blurred using GaussianBlur to reduce noise and emphasize edges.

5. Inverting the Blurred Image: The blurred image is inverted again to bring back the sketch-like effect.

6. Generating the Sketch: The final sketch image is obtained by dividing the original grayscale image by the inverted blurred image using the divide function.

7. Saving the Sketch Image: The resulting sketch image is saved as "Sketch4.png" using the imwrite function from the OpenCV library.

PROGRAM:-



```
import cv2 as cv

# Reading the image
# Replace this image name to your image name
image = cv.imread("wp.jpg")

# Converting the Image into gray_image
gray_image = cv.cvtColor(image, cv.COLOR_BGR2GRAY)

# Inverting the Imge
invert_image = cv.bitwise_not(gray_image)

# Blur Image
blur_image = cv.GaussianBlur(invert_image, (21,21), 0)

# Inverting the Blured Image
invert_blur = cv.bitwise_not(blur_image)

# Convert Image Into sketch
sketch = cv.divide(gray_image, invert_blur, scale=256.0)

# Generating the Sketch Image Named as Sketch.png
cv.imwrite("Sketch4.png", invert_blur)
```

Explanation:-

This Python code snippet utilizes the OpenCV library to perform image-to-sketch conversion. Here's a brief explanation of each step:

1. Reading the Image: The code reads an input image named "wp.jpg" using the `cv.imread()` function from OpenCV.

2. Converting to Grayscale: The input image is converted into a grayscale image using `cv.cvtColor()` function with the parameter `cv.COLOR_BGR2GRAY`.

3. Inverting the Image: The grayscale image is inverted using `cv.bitwise_not()` function to obtain a negative image.

4. Blurring the Image: The inverted image is blurred using Gaussian blur with kernel size (21,21) to smooth out the details and noise, resulting in a blurred image.

5. Inverting the Blurred Image: The blurred image is inverted again to bring back the original intensity values.

6. Creating the Sketch: The original grayscale image is divided by the inverted blurred image using `cv.divide()` function with a scale of 256. This operation enhances the edges and contrasts, resulting in a sketch-like effect.

7. Saving the Sketch: Finally, the generated sketch image is saved as "Sketch4.png" using `cv.imwrite()` function.

Overall, this code converts the input image into a sketch-like representation by inverting, blurring, and enhancing the edges of the image. The resulting sketch is saved as a PNG file named "Sketch4.png".

CONCLUSION

In conclusion, the Python program for converting an image to a sketch using the OpenCV library and associated functions provides a straightforward and effective method for transforming images into sketch-like representations. By sequentially applying operations such as grayscale conversion, inversion, blurring, and division, the program facilitates the generation of visually appealing sketch images from input photographs or images.

The program's functionality allows for customization through parameter adjustments, such as the blur radius and inversion, providing users with the flexibility to fine-tune the sketch output according to their preferences. Additionally, the capability to save the resulting sketch image as a separate file enables users to preserve and share the transformed artwork.

Overall, the program offers a user-friendly approach to image manipulation, catering to individuals interested in artistic expression, digital imaging, and creative visualization. Its accessibility and versatility make it a valuable tool for both amateur and professional users seeking to explore the artistic potential of image processing techniques within the Python programming environment.