Day 1 Assignment: Pencil Sketch Effect

Computer Vision Bootcamp

made with ♡, by Aryan

Due: Next Session

1 Objective

Create a Python program that transforms any photograph into a realistic pencil sketch drawing using classical image processing techniques.

2 Background

The pencil sketch effect simulates traditional pencil drawing by:

- Extracting edges and contours
- Creating grayscale tones that mimic pencil shading
- Using inversion and blending techniques

This technique is based on the "dodge and burn" method from photography, adapted for digital image processing.

3 Algorithm

Your program should implement the following steps:

3.1 Step 1: Convert to Grayscale

Convert the input color image to grayscale using OpenCV:

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

3.2 Step 2: Invert the Grayscale Image

Create a negative by inverting pixel values:

$$inverted(x, y) = 255 - gray(x, y) \tag{1}$$

3.3 Step 3: Apply Gaussian Blur

Blur the inverted image with a large kernel (21x21):

```
blurred = cv2.GaussianBlur(inverted, (21, 21), 0)
```

The large kernel size creates soft, diffused tones.

3.4 Step 4: Invert the Blurred Image

Invert again to prepare for blending:

$$inverted\_blur(x, y) = 255 - blurred(x, y) \tag{2}$$

3.5 Step 5: Divide and Scale

Create the sketch effect using division and scaling:

$$sketch(x, y) = \min\left(255, \frac{gray(x, y)}{inverted\_blur(x, y)} \times 256\right) \tag{3}$$

This step brightens highlights and emphasizes edges, creating the pencil sketch appearance.

4 Requirements

4.1 Functional Requirements

Your program must:

1. Accept an image file path as input
2. Implement the complete pencil sketch algorithm
3. Display the original and sketch side-by-side
4. Save the sketch output to a file
5. Include error handling for:
    - File not found
    - Invalid image format
    - Processing errors

4.2 Code Quality Requirements

- Use functions to organize code (minimum 2 functions)
- Include docstrings for all functions
- Add inline comments for complex operations
- Follow Python naming conventions (PEP 8)
- Handle edge cases gracefully

4.3 Testing Requirements

Test your program with at least 3 different images:

1. A portrait/face
2. A landscape/scenery
3. An object with distinct edges

5 Starter Code

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

def pencil_sketch(image_path, blur_kernel=21):
    """
    Convert an image to pencil sketch effect.

    Args:
        image_path (str): Path to input image
        blur_kernel (int): Gaussian blur kernel size (must be odd)

    Returns:
        tuple: (original_rgb, sketch) or (None, None) if error
    """
    # TODO: Implement the algorithm
    # Step 1: Load image

    # Step 2: Convert to grayscale

    # Step 3: Invert grayscale

    # Step 4: Apply Gaussian blur

    # Step 5: Invert blurred image

    # Step 6: Divide and scale

    pass


def display_result(original, sketch, save_path=None):
    """
    Display original and sketch side-by-side.

    Args:
        original: Original image (RGB)
        sketch: Sketch image (grayscale)
        save_path: Optional path to save the sketch
    """
    # TODO: Create matplotlib figure with 1 row, 2 columns
    # Display original on left, sketch on right
    # Add titles and remove axes
    # If save_path provided, save the sketch

    pass


def main():
    """Main function to run the pencil sketch converter."""
    # TODO: Get image path from user or command line
    # Call pencil_sketch function
    # Call display_result function
    # Handle any errors

    pass
```

```
57
58
59 if __name__ == '__main__':
60     main()
```

6 Submission Guidelines

6.1 What to Submit

1. Python script (`pencil_sketch.py`)
2. 3 test images (original)
3. 3 output sketches
4. README file with:
    - How to run your program
    - Any dependencies
    - Observations about results
    - Any challenges faced

6.2 Submission Format

Create a ZIP file named `Day1_Assignment_YourName.zip` containing:

```
Day1_Assignment_YourName/
|-- pencil_sketch.py
|-- test_images/
|   |-- test1.jpg
|   |-- test2.jpg
|   +-- test3.jpg
|-- output_sketches/
|   |-- sketch1.jpg
|   |-- sketch2.jpg
|   +-- sketch3.jpg
+-- README.txt
```

7 Grading Rubric

Criteria	**Points**
Algorithm Implementation	30
Code Quality & Organization	20
Error Handling	10
Display & Visualization	15
Testing (3 images)	15
Documentation	10
Total	**100**

8 Bonus Challenges

Implement any of these for extra credit:

8.1 Bonus 1: Adjustable Blur Parameter (+10 points)

Add command-line argument or user input to adjust blur kernel size dynamically.

8.2 Bonus 2: Color Pencil Sketch (+15 points)

Create a colored sketch version:

- Convert to HSV color space
- Apply sketch algorithm to Value channel only
- Combine with Hue and Saturation
- Add slight desaturation for realistic effect

8.3 Bonus 3: Video Processing (+20 points)

Extend your program to process video files frame-by-frame:

- Read video using `cv2.VideoCapture`
- Apply sketch effect to each frame
- Write output video using `cv2.VideoWriter`
- Display progress bar

8.4 Bonus 4: Interactive GUI (+25 points)

Create a simple GUI using Tkinter or similar:

- File selection dialog
- Sliders to adjust blur kernel size
- Real-time preview
- Save button

9 Tips for Success

- **Start simple:** Get the basic algorithm working first before adding features
- **Test incrementally:** Display intermediate results after each step
- **Handle division by zero:** Add small epsilon when dividing
- **Experiment with kernel size:** Try different values (15, 21, 25, 31)
- **Check data types:** Ensure proper uint8 conversion for display
- **Use git:** Version control helps track your progress

10 Common Pitfalls

1. **Division by zero:** Add `+ 1e-6` to denominator
2. **Overflow errors:** Use `np.clip()` to constrain values
3. **Wrong data types:** Convert to `uint8` for display
4. **BGR vs RGB:** Remember OpenCV uses BGR
5. **Odd kernel sizes:** Gaussian blur requires odd kernel sizes

11 Resources

- OpenCV Documentation: `https://docs.opencv.org`
- NumPy Documentation: `https://numpy.org/doc`
- Matplotlib Gallery: `https://matplotlib.org/stable/gallery`

Good luck! We look forward to seeing your creative sketches!