Adobe GenSolve: CURVETOPIA

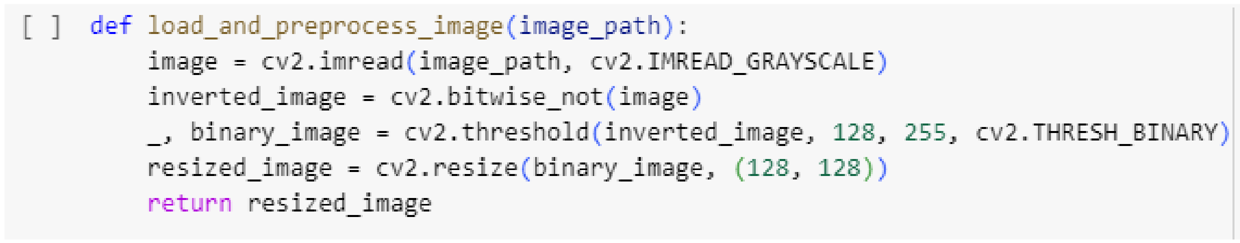
Algorithm for Curve Regularization, Symmetry, and Completion

Objective:
The goal is to create an algorithm that takes an image containing 2D curves, makes the curves look smoother and more symmetrical, and fills in any gaps to complete the shapes. The final output is a cleaned-up version of the curves that are ready for further use.

Step 1: Load and Preprocess the Image

- **Input:** A grayscale image containing 2D curves.
- **Process:**
    - Load the image in grayscale mode.
    - Invert the colors to make the curves black and the background white.
    - Convert the image to a binary format.
    - Resize the image to a standard size (e.g., 128x128 pixels) for easier processing.

Code:

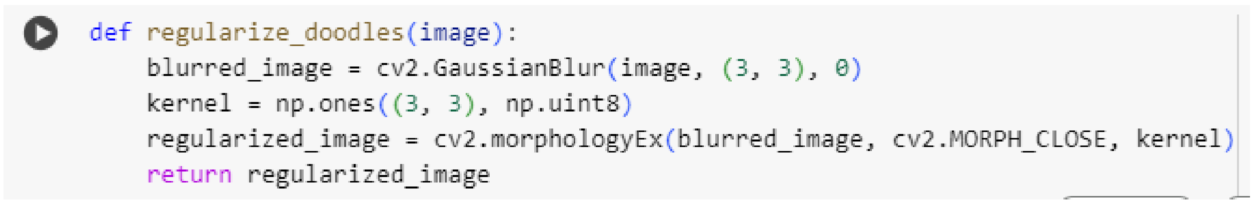


```python
def load_and_preprocess_image(image_path):
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    inverted_image = cv2.bitwise_not(image)
    _, binary_image = cv2.threshold(inverted_image, 128, 255, cv2.THRESH_BINARY)
    resized_image = cv2.resize(binary_image, (128, 128))
    return resized_image
```

Step 2: Regularize the Curves

- **Input:** The preprocessed binary image.
- **Process:**
    - Smooth the image using a small blur to reduce noise.
    - Close small gaps in the curves to make the shapes more regular and uniform.

Code:

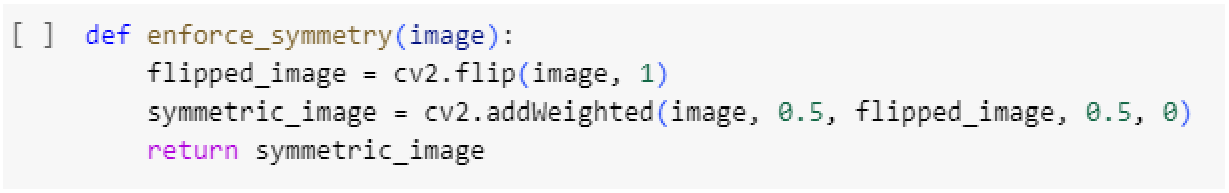


```python
def regularize_doodles(image):
    blurred_image = cv2.GaussianBlur(image, (3, 3), 0)
    kernel = np.ones((3, 3), np.uint8)
    regularized_image = cv2.morphologyEx(blurred_image, cv2.MORPH_CLOSE, kernel)
    return regularized_image
```

Step 3: Enforce Symmetry

- **Input:** The regularized image.
- **Process:**
    - Flip the image horizontally to create a mirror image.
    - Combine the original and the flipped images to create a symmetric doodle.

Code:

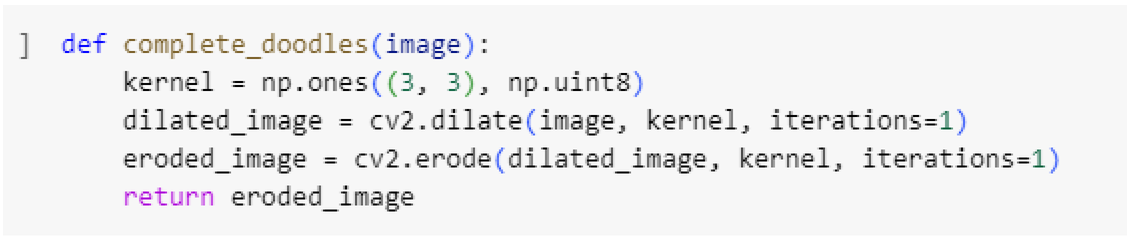


```python
def enforce_symmetry(image):
    flipped_image = cv2.flip(image, 1)
    symmetric_image = cv2.addWeighted(image, 0.5, flipped_image, 0.5, 0)
    return symmetric_image
```

Step 4: Complete the Curves

- **Input:** The symmetric image.
- **Process:**
    - Expand the curves slightly to fill in any remaining gaps.
    - Refine the edges to remove any unwanted artifacts.

Code:

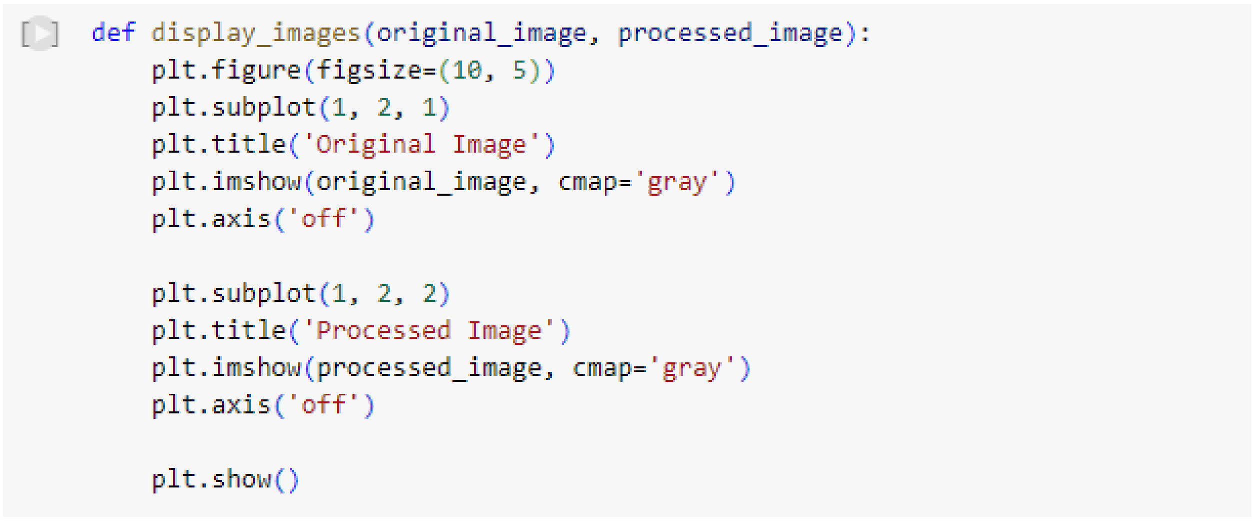


```python
def complete_doodles(image):
    kernel = np.ones((3, 3), np.uint8)
    dilated_image = cv2.dilate(image, kernel, iterations=1)
    eroded_image = cv2.erode(dilated_image, kernel, iterations=1)
    return eroded_image
```

Step 5: Display the Final Processed Image

- **Input:** The final processed image.
- **Process:**
    - Display the processed image using a visualization tool to view the results.

Code:

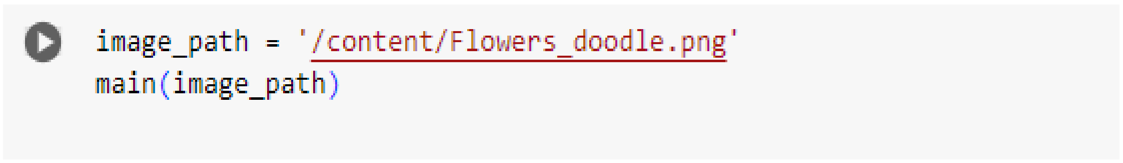


```python
def display_images(original_image, processed_image):
    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.title('Original Image')
    plt.imshow(original_image, cmap='gray')
    plt.axis('off')

    plt.subplot(1, 2, 2)
    plt.title('Processed Image')
    plt.imshow(processed_image, cmap='gray')
    plt.axis('off')

    plt.show()
```

Step 7: Run the Complete Process

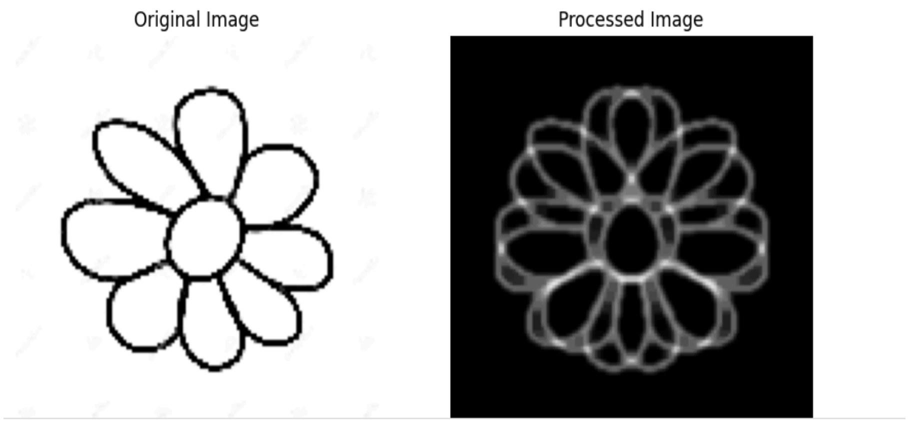
- **Input:** Path to the image file.
- **Process:**
  - Load and preprocess the image.
  - Apply the regularization, symmetry enforcement, and completion steps.
  - Display the final result.

Code:



```python
image_path = '/content/Flowers_doodle.png'
main(image_path)
```

Output:


Original Image
Processed Image