2025 Spring | Computer Graphics Term Project Report

StyleSplit: Mix Artistic Styles on Images

R13922A04 操之晴、R13944008 吳宜宸、R13944035 張翔

Project Code：https://github.com/7ching/StyleSplit

1. Introduction

In this project, we present StyleSplit, an interactive image stylization tool that applies different artistic effects—oil painting, watercolor, sketch, and comic—to segmented regions of an image. Rather than relying on deep learning-based style transfer methods, all visual effects in StyleSplit are implemented using traditional image processing techniques.

To enhance user interactivity and control, we integrate the Segment Anything Model (SAM) [1], which automatically segments images into semantically meaningful regions. Through a web interface built with Streamlit, users can intuitively assign different artistic styles to each segmented region.

2. System Overview

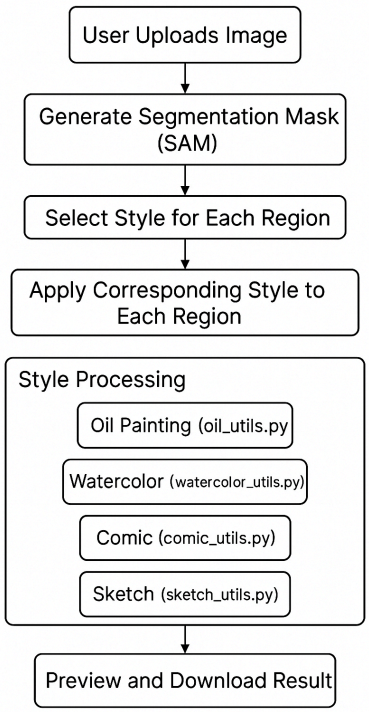
Our system is designed to provide flexible, region-based artistic stylization by combining traditional image processing techniques with a modern segmentation model. It is composed of three main components: the Segmentation Module, Style Modules, and a User Interface.

- **Segmentation Module**
  We apply the Segment Anything Model (SAM) to perform automatic image segmentation. It generates multiple region masks that allow users to apply different styles to different areas of the image.

- **Style Modules**
  We implemented four distinct artistic styles using classical image processing techniques:

  - **Oil Painting:** Mimics brush strokes and color clustering effects.
  - **Watercolor:** Softens edges and simulates pigment diffusion.
  - **Sketch:** Converts the image to grayscale and highlights contours to simulate pencil strokes.
  - **Comic:** Applies halftone patterns and strong outlines to create comic-style visuals.

- **User Interface**
  A simple and intuitive web interface built with Streamlit supports image upload, region-based style assignment, live preview of the result, and downloading the final output image.



3. Implementation Details

Our project is implemented in Python. Each style effect is designed as a standalone module for clarity and modularity. We integrated the SAM model to handle segmentation, enabling region-specific style application. The user interface is built using Streamlit, providing a seamless and interactive workflow from image upload to stylized output preview and download.

3.1. Segmentation

We use the Segment Anything Model (SAM) to automatically segment the input image into multiple regions. To avoid clutter and improve visual clarity, we discard masks whose areas are smaller than 5% of the total image size.

3.2. Oil Painting

The implementation simulates a painterly rendering process by iteratively applying elliptical brush strokes across an input image. The algorithm begins by determining a base brush radius based on a given parameter, with a lower bound to ensure stability. Prior to stroke generation, the image may be preprocessed into grayscale format depending on whether edge alignment or adaptive brush sizing is enabled. If edge alignment is requested, image gradients are computed using directional Sobel filters, and orientations are derived via arctangent operations. For adaptive brush scaling, a detail map is generated using a Laplacian filter, normalized to a [0, 1] range.

A collection of randomized brush templates is generated in advance. Each template encodes a random elliptical scale factor and a base rotation angle, influenced by an expression control parameter. These templates serve to introduce controlled variation in stroke appearance across the canvas.

The canvas is then traversed in a grid-like fashion, with the spacing between strokes determined by the brush radius and a predefined spacing factor. At each location:

1. The brush radius may be adjusted dynamically based on local image detail, if the adaptive sizing feature is active.
2. A pre-generated template is selected to determine the stroke's secondary radius.

3. The final rotation angle is computed, optionally blending the template's inherent angle with local gradient orientation to achieve edge alignment.
4. An elliptical mask is rendered using the computed radii and rotation, centered around the current location.
5. The base color for the stroke is sampled from the image; if specified, a color perturbation is applied using random noise within a constrained range.
6. Finally, the colored elliptical stroke is applied onto the result canvas at the corresponding coordinates, respecting image boundaries to avoid indexing errors.

The algorithm iterates this procedure until the entire canvas is populated with strokes, yielding a stylized output image that reflects the original content while incorporating expression, randomness, and edge-aware structure.

3.3. Watercolor

Our watercolor effect is implemented following the methodology described in the paper "*Towards Photo Watercolorization with Artistic Verisimilitude*" by M. Wang et al. [2], and is realized entirely through traditional image processing techniques. For implementation details not fully covered in the original paper, we referred to an open-source GitHub project [3]. The implementation is structured as a multi-stage pipeline, described as follows:

Color Adjustment. The input image is converted to the Lab color space and its mean and standard deviation are matched to a predefined color palette using the Reinhard color transfer technique. This step harmonizes the color tone with the target watercolor style.

Saliency Distance Field Computation. A fine-grained saliency map is computed using OpenCV's static saliency detector. This binary saliency map is then inverted to produce a distance field, which encodes the

proximity of each pixel to non-salient regions. The distance field is repeatedly smoothed using Gaussian blur for stability and spatial coherence.

Abstraction. The image is segmented using the HFS (Hierarchical Feature Segmentation) algorithm. For each pixel, a patch around it is analyzed; salient regions use a fixed kernel size, while non-salient regions use an adaptive kernel based on the saliency distance. Within the patch, colors from similar segments are averaged to simplify unimportant regions while preserving detail in salient areas.

Boundary Classification. Gradients are computed using the Scharr operator, and pixel boundaries are categorized into three types based on saliency values and hue differences in the HSV color space. Boundaries in low-saliency areas with minimal hue contrast are considered part of wet-in-wet regions and are left undistorted. Boundaries between regions with similar hues or low chromatic intensity are marked for soft distortion. All other boundaries, typically involving high color contrast, are marked for stronger distortion with overlaps and gaps.

Wet-in-Wet Simulation. This effect is applied to boundaries classified as undistorted. A combination of color scattering and oriented filtering is performed along the gradient direction to simulate pigment diffusion on wet paper. Directionally adaptive kernels are used to create soft, feather-like blending across region edges.

Hand Tremor Simulation. For distorted boundaries, we apply Perlin noise to displace pixels and introduce irregularities that mimic human hand tremors. Boundaries classified for soft distortion are slightly jittered, while high-contrast edges undergo stronger displacement with controlled overlaps and gaps to emulate the unstable linework of traditional brush strokes.

For completeness, we also implemented some other watercolor effects that have been proposed in previous methods:

Edge Darkening. Gradient-based edge detection followed by local blurring and dilating simulates pigment accumulation along edges. The effect is added multiplicatively to the original image to enhance depth and definition.

Granulation. Multi-scale Perlin noise is applied to simulate pigment granules accumulating unevenly on textured paper. This noise is blended multiplicatively with the image to add high-frequency texture.

Turbulence Flow. A large-scale low-frequency Perlin noise texture is used to simulate uneven pigment distribution due to variations in water density. This adds broad, sweeping texture variation across the image.

Finally, we apply FXAA (Fast Approximate Anti-Aliasing) to smooth pixelated edges. This complete pipeline captures the essential characteristics of watercolor rendering, including soft transitions, texture variation, and stroke irregularities. It produces visually coherent results suitable for artistic image stylization.

3.4. Sketch

Our sketch effect is implemented entirely through traditional image processing techniques using OpenCV, inspired by commonly used pencil sketch stylization pipelines. While the implementation is not based on a specific academic paper, it follows the general principles of luminance preservation and stroke-based blending. For fine-tuning, we referred to open-source projects and tutorials on photographic-to-sketch conversion using division-based blending. The implementation is structured as a simple yet effective multi-stage pipeline, described as follows:

Grayscale Conversion. The input color image is first converted into a grayscale image using OpenCV's cv2.cvtColor function. This reduces computational complexity and emphasizes luminance contrast, which is crucial for simulating pencil lines.

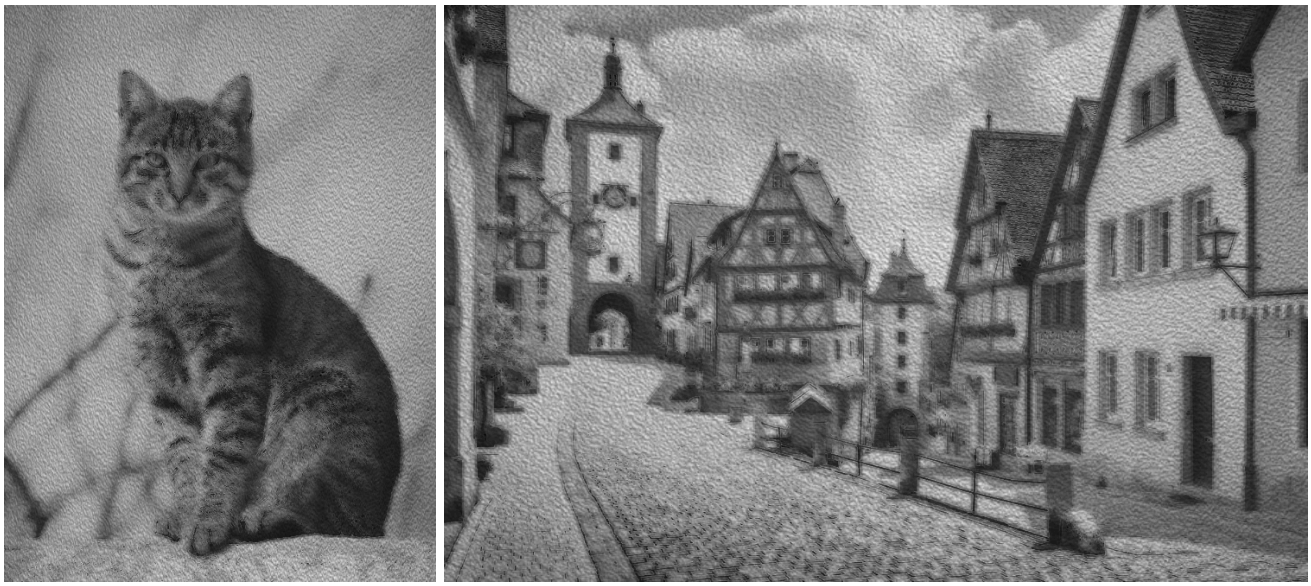
Inversion. The grayscale image is then inverted (i.e., 255 - gray) to prepare for the dodge blending technique. The inversion process highlights edges and contours in the image that will later be emphasized in the sketch.

Blurring. A Gaussian blur is applied to the inverted grayscale image using a kernel size such as (21, 21). This blurred image serves as the "light map" for the dodge blending, smoothing out harsh transitions and simulating the soft shading of a pencil.

Dodge Blending. The core sketch appearance is generated by dividing the grayscale image by the inverted, blurred image using OpenCV's cv2.divide function. This dodge blend mimics the way highlights are formed when artists use erasers or light strokes over shaded regions, effectively producing the light-sketch illusion.

Post-Processing (optional). In some variants, we also apply contrast enhancement or clipping to adjust the visual strength of the stroke lines. These can be tuned for aesthetic preference or better foreground-background separation.

This lightweight yet effective pipeline produces a clear, high-contrast pencil sketch that visually resembles hand-drawn artwork. The method is robust, real-time, and easy to integrate into stylization frameworks, particularly in combination with segmentation masks (e.g., from SAM) to allow localized sketch application (e.g., only on background or foreground).



3.5. Comic

This method produces a comic-style effect on input images by integrating edge enhancement with color quantization. The process consists of three key stages: edge detection, color simplification, and image compositing.

The first step converts the input image into grayscale to facilitate edge analysis. A Gaussian blur is applied to suppress noise and reduce high-frequency artifacts, which helps prevent spurious edges in the next phase. Edge features are then extracted using a gradient-based method that emphasizes boundaries where intensity changes sharply. This results in a binary edge map highlighting prominent contours in the image.

In the second stage, the original color information is compressed using vector quantization via unsupervised clustering. The image is flattened into a two-dimensional array of color vectors and subjected to a clustering algorithm that partitions the color space into a limited number of representative centroids. Each pixel is then reassigned to the nearest color cluster, producing a simplified version of the image with a reduced palette. This reduction enhances the flat, posterized aesthetic typical of comic artwork.

Finally, the binary edge map from the first stage is overlaid onto the quantized image. Wherever strong edges were detected, the corresponding pixels are replaced with black, simulating inked contour lines. This compositing step fuses structural and color-based abstraction, yielding an output that combines bold edges with stylistically flattened color regions—emulating the visual language of hand-drawn comic illustrations.

4. Results

Our stylization framework allows for selective application of different visual styles to foreground and background regions. In the example shown above, the original image (left) features a hot air balloon floating above a house in a golden field. The processed result (right) demonstrates the use of comic stylization on the balloon, sketch rendering on the house, oil painting on the field, and a watercolor style on the sky background.

This result illustrates the system's ability to segment and stylize different semantic regions with distinct effects. The balloon maintains sharp edges and high-contrast lines characteristic of comic illustrations, while the house appears with pencil shading and contour lines typical of hand-drawn sketches. The field adopts a rich, textured look consistent with oil painting, while the sky exhibits pigment granulation and smooth tonal transitions characteristic of watercolor washes.

By combining semantic segmentation with style-specific processing modules, we achieve coherent multi-style rendering that respects object boundaries and preserves perceptual saliency.

Work Distribution

- R13922A04 操之晴：sketch style implementation, report
- R13944008 吳宜宸：watercolor style implementation, segmentation implementation, report
- R13944035 張翔：oil painting and comic style implementation, report

Acknowledgment of LLM Assistance

This report was revised using ChatGPT (OpenAI) for language improvement. During code development, GitHub Copilot was used to assist with code suggestions. Some demo images were generated by Gemini. All ideas, implementations, and final results were independently developed and verified by our team.

References

[1] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, A. Bordes, W.-Y. Lo, and R. Girshick, "The Segment Anything Model," arXiv preprint arXiv:2304.02643, 2023. [Online]. Available: https://github.com/facebookresearch/segment-anything.

[2] M. Wang, L. Wang, Y. Qiao, M. Gong, and Y. He, "Towards Photo Watercolorization with Artistic Verisimilitude," IEEE Trans. Vis. Comput. Graphics, vol. 20, no. 10, pp. 1451–1460, Oct. 2014, doi: 10.1109/TVCG.2014.2303984.

[3] devin6011, "ICGproject-watercolorization," GitHub repository, 2019. [Online]. Available: https://github.com/devin6011/ICGproject-watercolorization.