Sketchy Business: Implementing CLIPascene

Apoorva Talwalkar, Sophie Shao and Fiona Fan

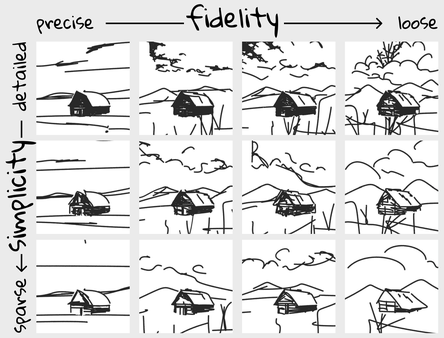
Introduction

For our project we created a model that

converts an input image into a sketch.

To do this, we planned on following the **CLIPascene: Scene Sketching with Different Types and Levels of Abstraction** paper by researchers at Tel Aviv University and Reichman University. The paper presents a method for converting a given scene image into a sketch using different types and multiple levels of abstractions. The model outputs a matrix of sketches with axes representing fidelity and simplicity, from which the user can select one.

input



We chose this paper because it provides a **straightforward approach to transforming images into a new artistic style** while giving users some adjustability. Additionally, the three of us are also artists so we found this paper especially interesting and applicable to us.

Paper Methodology

The paper's architecture

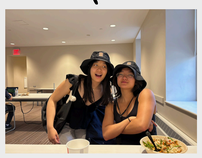
trains 2 multi-layer perceptrons

to complete the task. First, the image is converted into a set of Bezier curves. The first MLP takes the curves' control points as input and outputs offsets to adjust them. Then, for a given n levels of fidelity and m levels of simplicity, a sketch abstraction matrix of size m * n is constructed. The sketches along the first row are generated (each decreasing in fidelity).

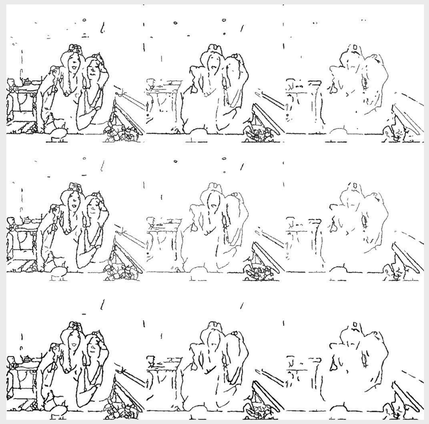
Then for each of these sketches, an iterative simplification is performed to fill in each column of the matrix, for which they trained another MLP. This second MLP receives a random-valued vector and learns an n-dimensional vector representing the probability of the i-th stroke appearing in the final sketch and outputs a simplified sketch.

Results

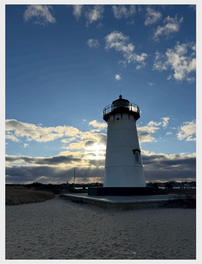
input



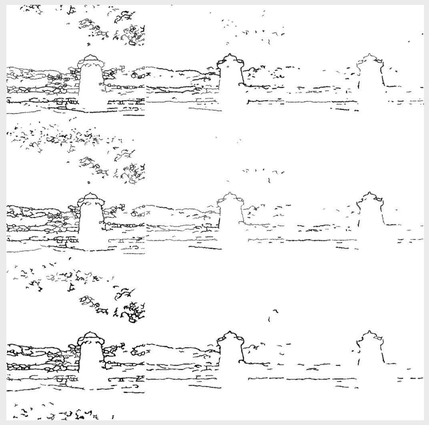
accuracy: 50%



input

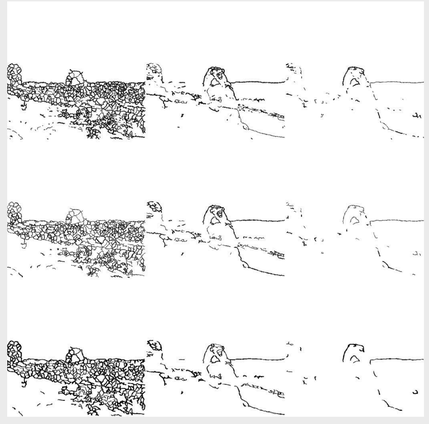


accuracy: 95%



input

accuracy: 80%



Our Implementation

While we first tried implementing the exact methodology from the paper, in response to challenges, we decided to opt to use

a saliency map and learned filters

in our implementation. For our model, we first generate vectorized sketches from an image by resizing and normalizing the input. Then we create a saliency map using Sobel edge detection to highlight important structures, and apply a pencil sketch filter to the image and mask it using the saliency map to retain only significant edges. The resulting sketch is skeletonized into thin lines, and contours are extracted, downsampled, and jittered to simulate hand-drawn strokes. These strokes are drawn on a blank canvas with varying thicknesses to create a natural sketch effect. The process is then repeated across different settings of fidelity and simplicity, to produce a matrix of sketches like the original paper

Discussion

challenges

We found that using a rasterizer with the MLPs was the most hardest part of the original paper. It was a challenge to generate random control points and the render them with a rasterizer. We instead decided to implement a saliency map with learned filters to complete the same task. We found this to be much simpler and still provided sketch-like outputs.

accuracy

Another challenge we faced, was defining accuracy, since the authors of the original paper didn't explicitly define how they arrived at an accuracy value.

To measure accuracy for ourselves, we chose 20 images to test. For each of these images we predefined a set of "ground-truth" tags (eg. dog, cat, people). We fed the output sketches for each of these into an LLM (Chat-GPT) with the prompt:

"list the objects you see in this sketch"

We specified that we wanted the top 5 objects. From here, we computed the percent of images where at least 2 ground-truth labels were present in the LLM's predictions. We used this as our accuracy metric.

future work

The methods in this paper can be expanded to adapt for artistic style or different brush types. For example, a model can be trained to implement sketches in the style of Van Gogh or to sketch with pencil, pen, or charcoal.