Drawings from Photos

Hubert Hua Kian Teo
Computer Science dept.
Stanford University
hteo@stanford.edu

Gael Colas
Aeronautics and Astronautics dept.
Stanford University
colasg@stanford.edu

Andrew Deng
Electrical Engineering dept.
Stanford University
andrewde@stanford.edu

***Abstract*—Many digital artists specialize in producing stylized, non-photo-realistic (NPR) digital paint drawings. These lovingly-created drawings can be very expressive and incorporate unique aesthetic decisions that become part of the artist's style. Our goal is a form of image translation: to produce NPR drawings from real-world photographs.**

I. Introduction

Graphic manipulation software such as Adobe Photoshop provide built-in "filters" to apply effects on images. Some of these are dedicated to achieving the effect of NPR digital paint drawings. However, they are not easily customizable and require a decent amount of user tuning and manipulation to produce a desirable result. Our project aims at building a "Photo2Drawing" pipeline to automatically achieve image translation from photos to drawings. Pipeline components are build using Digital Image Processing techniques. The pipeline can be decomposed in two main parts: getting the contour edges of the drawing and applying the colors. We extended our deterministic image translation method to a more customizable method that allows extensive artistic control over the result. For each pipeline components, we identified hyperparameters that we gathered into meaningful user-oriented customization axis. Finally, we provide a Graphical User Interface (GUI) to dynamically tweak the result along these axes.

II. Dataset

Due to time constraints, we restricted ourselves to photos of faces. These are also a good choice for translation to digital paintings because portraits are a common subject for digital paintings. The advantage of using only face pictures is that the images have similar contents and are more easily comparable. In addition, they still offer a lot of diversity in many aspects: image quality, variety of colors, lighting and background. Moreover, image translation on face images is an important sub-task considering the current "selfie" trend. For example, the mobile application Snapchat provides users with a wide variety of filters to apply on their selfies. Our method could be one of them.

We used the 'Labeled Faces in the Wild' Face dataset [1]. This dataset was generated by scraping the Internet for celebrities face photos. It is composed of 13,000 celebrities face photographs. All the images have been resized to the same size: 250×250 RGB pixels.

III. Previous Work on NPR components

There has been previous work on each of the intermediate goals of our "Photo2Drawing" pipeline: line-sketch generation, region of interest selection, color generation and final blending. This section aims both at presenting these state-of-the-art methods and explaining how they fit in our pipeline. We believe that a pleasing result can be obtained by producing several NPR components from a source image and blending them together.

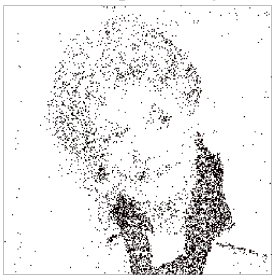
A. Line-sketch generation

The first component of our pipeline aims to automatically generate a line drawing of the image, representing what an artist would do to outline the edges and fine details of the photo.

1) **Canny edge detection**
   Gradient-based edge detection offers a simple method to extract a line drawing from a photograph. We used the Canny edge detector. This algorithm produces binary representations of the lines in the photographs. By changing the detection threshold, the amount of detected edges can be tuned. Effectively this changes the granularity of the line-sketch. Finally our component dilates the edges with a structuring element. The size of the structuring element can be adjusted to match the preference of the user: a bigger structuring element means thicker edges.
2) **Line-integral convolution (LIC)**
   We also considered line-integral convolution methods [2] [3] because they promise to provide more control over the result.
   These methods produce a pencil sketch image by separately choosing the directions of the pencil strokes, and where the strokes are placed. The local direction of the strokes is determined by a vector field, and the stroke placement is determined by a black and white noise image, where each black pixel on the white background represents a stroke. LIC then moves a particle backwards and forwards in the vector field starting from each pixel position, samples the noise image at equal intervals over that streamline, and convolves those samples with a kernel to produce the final pixel color. The effect of this is to smear each black pixel along the vector field's direction, producing a result not unlike pencil strokes.

(a) Region labels used for vector field processing.

(b) Streamlines in the resulting vector field.



(c) Noise image used in convolution.

(d) Result obtained by convolving the noise result.

Fig. 1: LIC-based pencil sketch generation pipeline

In our implementation, we used a variant of the method from [2] to extract vector fields and produce noise images. First, image gradients are computed and rotated by 90 degrees, and then flipped so that all vectors point to the right. The rotation and flipping ensure that the vectors will be parallel to any preexisting striped region. Next, we segment the image into connected regions based on the watershed method, merging 8-connected pixels together in the order of lowest euclidean distance in Lab space. For each region, we then snap all vectors to the mean vector direction if the variance in their direction is above a certain threshold. This produces many regions with vectors pointing in roughly the same direction. For speed, we used FastLIC [4] for the convolution. See Fig. 1 for an illustration of the overall process.

B. Regions of interest identification

The next component considered was to identify regions of interest. These regions could be used to allow for more control in the other pipeline steps. Examples of strategies taking advantage of the region of interests would be to only extract line sketches from regions of interest or only use colors from regions of interest to generate the final colors. Finally if the faces had been well identified by this method we could have used this component to get rid of the background.

We experimented with saliency-based masks for automatic region identification. Based on existing work using saliency maps for spot coloring [5] [6], the idea was to extract the saliency values for each pixel in the image as a measure of how visually important each pixel was and use binarization with some post-processing to form contiguous regions.

Saliency values were computed using the color L2-distance between pixel colors. To avoid excessive computation, color values were quantized into 12 values for each channel. Quantization in the RGB-space yielded better results. The corresponding 3D-color histogram of dimensions $12 \times 12 \times 12$ was formed of to count the occurrence of each distinct quantized color. Saliency was computed for every discretized color with nonzero count as:

$$S(C) = \sum_{C_k \in \mathcal{C}} n_k \|C - C_k\|_2 \tag{1}$$

where $\mathcal{C}$ is the set of all of the quantized colors and n_k is the count in the color histogram bin for color C_k. Finally the saliency values were normalized to be between 0 and 1.

Replacing each color pixel with its associated saliency produced a grayscale image called a saliency map. Otsu's method was used to binarize the map into foreground and background. The idea was that visually distinct regions would have high salency scores. Finally the binarization process produced both small noise pixels as well as holes, so morphological image processing was applied to eliminate these flaws in the mask.

The parameters tuned for this experiment include the quantization factor and the binarization threshold. It was found that increasing the quantization factor resulted in significantly longer computation time (cubic increase in number of bins), and decreasing it resulted in larger regions that failed to capture boundary details between some of the more similar regions. An axis quantization in 12 bins was found to be a good trade-off. Finally the binarization threshold was computed using Otsu's method by default. However, it can also be fixed by the user for customization purpose. For the majority of photos in the dataset it was found that choosing a base threshold of 0.4 achieved good results.

C. Color generation

1) **LAB-space k-means clustering** A simple method to select the colors of the output drawing is to perform clustering on the pixels colors. K-means clustering was used to compute the clusters. The clusters were then used as a fixed color palette to re-render the photograph: each color pixel in the input image was replaced by the closest cluster. This allows us to effectively reduce the number of colors used while preserving the overall color information of the input. Finally, customizable transformations allow us to exaggerate or minimize color differences. In particular, we can apply a γ transformation to the the maximum LAB-channel of the color cluster with the power hyperparameter $\gamma \leq 1$ to amplify/subdue color differences.
2) **Color from region segmentation** The goal is to ensure that different regions of the images are assigned different colors, to preserve the structure content of the input. The idea is to use a region segmentation algorithm to identify what are the different regions of the image. Then

for each region, assign the mean color value to every belonging pixels. The averaging was done in the LAB-space that better accounts for human color perception. As in the previous method, the power hyperparameter $\gamma \leq 1$ can be changed to amplify color differences. Let's describe the region segmentation algorithm in more details. First, opening-closing by reconstruction is performed on the grayscale version of the input image to sharpen the transitions between regions. Then a Sobel gradient-based edge detector is applied to detect the region edges. Finally, region labeling is run on the complement binary image to identify the distinct regions. Fig. 2 shows the region segmentation steps on an example. The threshold of the edge detector can be modified by the user. A smaller threshold implies that more edges thus more regions will be detected. The output image will have more colors and be more detailed.

D. Blending

To get the final drawing output, the line sketch is applied on the generated colors. The pixels lying on the line sketch are replaced with gray pixels. The color of the line sketch can be chosen by the user. Finally to give some depth to the drawing, we apply an element-wise product between the drawing and the grayscale version of the input image.

IV. Analysis

This section explains how we chose the NPR components of our final model.

A. Line-sketch generation

We found that the only meaningful control over the line sketch result was the pencil stroke length. Tuning the threshold at which a region's vectors are snapped to the same mean vector does not change the result much. We eventually chose a threshold of 0.5. Larger and smaller values simply result in less interesting and overly messy vector fields, respectively.

To select the appropriate line length, we generated LIC results for all line lengths from 2 to 20 pixels. Then, we selected a line length of 12 for the final result, as it offered a good balance between long and perceivable strokes having fine details that are not overly smeared.

B. Region of interest selection

The saliency computation is purely based on color distances. This made the results often unsuitable for the purposes of generating appropriately styled drawings. For example, if a person's face had a similar color as most of the background pixels, it would be identified as visually unremarkable. For the system's use case it was desired that faces should be always selected as regions of interest, so this region identification scheme was not used in the final pipeline.

Another characteristic of the saliency method for region identification that made it unsuitable for the final product was the end result of the morphological image processing techniques. Even after experimenting with multiple different structuring element types, the regions produced were often misshapen in some way that would impact the appearance negatively if it were used in the pipeline.

C. Color generation

Performing the k-means clustering in the RGB-space led to visually unpleasing results. The first defect was that a lot of the output clustered images had a 'brownish aspect'. Indeed, due to the space taken by the face in the image, a lot of the pixels had brown colors. This was also seen when the background had one predominant color. The second defect was that RGB-clustering led to 'bleeding colors' issues. Some regions that appear as distinct on the input were assigned to the same RGB-cluster.

Performing the k-means clustering in the LAB-space solved some of these issues. The overall result is more pleasing for a human viewer. We observed more vivid colors assigned to each regions: the color centroids are more perceptually separated. Indeed, the color distances in the LAB-space corresponds more closely to the difference in colors perceived by humans. This can be seen easily on Fig. 5: the red tie of the input is assigned to the same red cluster as the red background in the LAB-clustering. In the RGB-clustering it was instead assigned to the same gray cluster as the suit.

V. Models

A. Baseline

NPR components:

- Line-sketch generation: Canny edge detector;
- Color generation: LAB-space k-means clustering.

Fig. 6 illustrates the baseline pipeline steps on an example.

B. Final model

NPR components:

- Line-sketch generation: Line-integral convolution;
- Color generation: Color from region segmentation.

VI. Customization Axes

We provided the user with tuned hyperparameter presets. But he is also able to control the result by selecting parameters: tweaking knobs that change the components parameters. To make this easy to use, we gathered the hyperparameters into 2 meaningful axis: realism and amount of details. This axis are tweaked in a Graphical User Interface (GUI) we implemented, see Fig. 7.

A. Controlling realism

The user can control the realism of the result (how close to the input it is) by choosing the parameter λ_r in the 'screen blending' between the pipeline output im_{draw} and the initial image im_{rgb}. The screen blending output image is given by the following formula:

$$im_{out} = 1 - (1 - \lambda_r im_{rgb})(1 - (1 - \lambda_r) im_{draw})$$

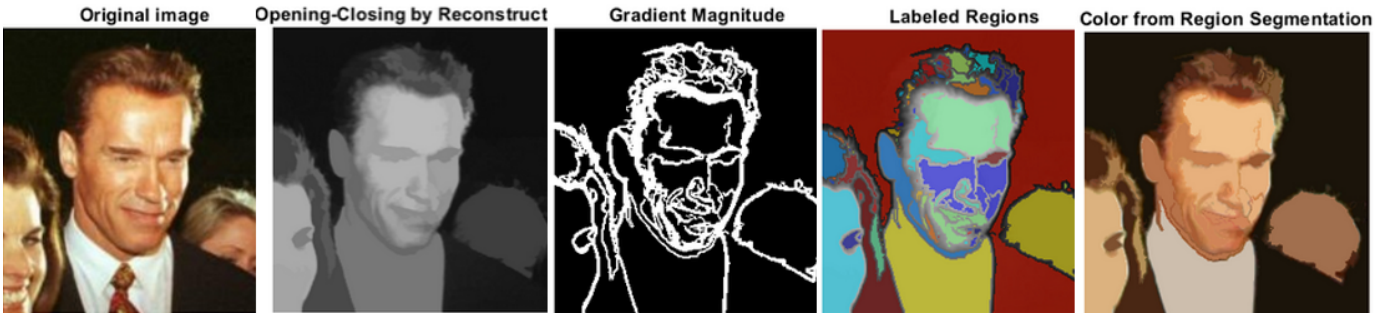


Fig. 2: Color region segmentation steps on an example

Fig. 3: An example of the region identification failing to identify a person's face

Fig. 4: An example of the output of the saliency region identification system showing the ugly jagged boundaries of some of the regions

Fig. 5: From left to right: original image, RGB-clustering, LAB-clustering

Fig. 6: From left to right: original image, edge detection, LAB-clustering, blending, blending after depth information

Fig. 7: GUI for interactive user modification

B. Controlling detail

Similarly, the user can control the amount of details in the output result by tweaking the following parameters:

- The edge detection threshold: smaller threshold means more edges and also more regions detected
- The density and length of lines in the LIC component

Fig. 8: From left to right: original image, baseline output, final model output, Photoshop filter output

VII. Evaluation

The quality of the drawings produced by our system could only be judged subjectively. To evaluate our performance, we set aside a test set of 20 images and generated output for each image using our pipeline. We then selected a preset filter in Adobe Photoshop that looked comparable to our result, and applied this filter to all the test images. We decided on the "smudge stick filter" as the best candidate. From these images, we randomly selected 10 output images from our final pipeline and 10 processed with the Photoshop style filter. Fig. 8 shows the output from our two models compared to the Photoshop filter output. Finally we built a user survey were we asked 12 people to rank the resulting images. Each image was evaluated on 2 scales from 1 to 5 (5 being best), Beauty and Realism.

TABLE I: Survey Results

Pipeline	Beauty	Realism
Photo2Drawing	2.225	1.8818
Photoshop	3.05	3.5417

From the results, it is apparent that the result of our pipeline was not judged favorably compared to the Photoshop filters. However, neither filter result was very good according to our respondents, since none of the ratings are close to 5.

VIII. Conclusion & Future Work

Since we were regrettably unable to include the automatic region segmentation method as part of our pipeline, future work would likely include exploring other approaches to improve the system. One approach that could be tried to improve the contrast between high saliency regions and low saliency regions is to combine saliencies computed using color and intensity with a singular value decomposition and dynamic mode decomposition based scheme [7]. Other methods that could bypass the crudeness of the binarization & morphological image processing-based approach to identify regions include deep-learning methods [8]. These could help by both improving region identification robustness as well as reducing the artifacts that can result from traditional region identification techniques.

We could also stand to improve the pencil sketch drawing result by allowing the user to manually edit the vector field. This could help in reducing the number of swirling artifacts due to the turbulent vector fields generated from noisy images. We could also experiment with using noise images that are not just black pixels but instead splats from pre-measured pencil dots, which could add a layer of realism to the drawing.

Github link: https://github.com/ColasGael/photo2drawing

Work Breakdown

Gael Colas implemented the following NPR components: line-sketch from edge detection, color generation from LAB-clustering and color generation from region segmentation. He also build the framework of the drawing pipeline and implemented the GUI. Andrew Deng performed experiments on using saliency in the LAB color space for automatic identification of visually distinct regions. He also built the evaluation survey. Hubert Teo implemented the FastLIC line integral convolution method and experimented with tuning the vector field extraction process, as well as the optimum convolution kernel length.

References

[1] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[2] Xiaoyang Mao, Yoshinori Nagasaka, and Atsumi Imamiya. Automatic generation of pencil drawing using lic. In *ACM SIGGRAPH 2002 Conference Abstracts and Applications*, SIGGRAPH '02, pages 149–149, New York, NY, USA, 2002. ACM.

[3] Xingyu Gao, Jingye Zhou, Zhenyu Chen, and Yiqiang Chen. Automatic generation of pencil sketch for 2d images. pages 1018–1021, 01 2010.

[4] Detlev Stalling and Hans-Christian Hege. Fast and resolution independent line integral convolution. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 249–256, New York, NY, USA, 1995. ACM.

[5] Paul L. Rosin and Yu-Kun Lai. Non-photorealistic rendering with spot colour. In *Proceedings of the Symposium on Computational Aesthetics*, CAE '13, pages 67–75, New York, NY, USA, 2013. ACM.

[6] Ming-Ming Cheng, Guo-Xin Zhang, Niloy Jyoti Mitra, Xiaolei Huang, and Shimin Hu. Global contrast based salient region detection. *CVPR 2011*, pages 409–416, 2011.

[7] Akshay Gadi Patil and Shanmuganathan Raman. Automatic content-aware non-photorealistic rendering of images. *CoRR*, abs/1604.01962, 2016.

[8] R. Zhao, W. Ouyang, H. Li, and X. Wang. Saliency detection by multi-context deep learning. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1265–1274, June 2015.