Oil Painting Style Transfer for Images Using Both Traditional and Deep Learning Methods

Chuan Sun  
University of KansasLawrence, KS  
chuansun@ku.edu

*Abstract*—In this paper, .

Keywords—Style transfer, F

# Introduction

With the rapid development of image processing and computer vision, the analysis on image styles has been making decent progress. Image styles are characterized by the artists’ painting custom and effect, which in practice correspond to a bunch of painting styles such as watercolor, sketch, oil painting, sand painting and so on. Transferring the style of an image to another is one of the main focus and practical application amidst the domain, which aims to take an arbitrary image as input, and generate an output with the same content but in the desired painting style.

A number of recent works have been conducted on analyzing the style of an image and trying to convert it to another. Traditional ways tend to utilize image processing techniques including image sharpening, edge detection, segmentation, reconstruction, etc. [1], to compute style features mathematically and apply the extracted features onto new images.

The work by Haeberli [2] proposed a method to simulate image tile mosaics using Voronoi diagram and distance measurement, which generates a mosaic-like filter style on images while preserving its contents. Fan Hui, Chen Zheng and Li Jinjiang designed a sand style transfer algorithm [1] using fuzzy enhancement and texture synthesis. These methodologies make full use of mathematical tools such as gradient, differential, probabilistic modeling, Manhattan distance, etc. to simulate the desired style patterns to be applied.

On the other hand, the flourishing development on neural network and deep learning provides a different approach for style comprehension and transformation. A lot of research work has been done on representing image content and style using deep learning techniques such as CNNs and VGG networks. For instance, the work by Leon A. Gatys separates the representation of image content from its style through convolutional neural networks [3]. Justin Johnson replaced the traditional pixel-level loss with perceptual loss defined on high level features [4] for the neural network doing image transformation.

Deep learning methodologies enables better generalization on feature extraction and representation through a stack of neural layers such as normalization, convolution, pooling, or even long short term memory for preserving long sequence information. While at the same time, they require real target tags to do supervised training in most cases.

In this paper, we explore the methodologies of transferring the style of an image to that of an oil painting. To achieve a good blend between oil painting style and image content, the oil brush patterns are analyzed. Then both traditional method and deep learning networks are applied to simulate the oil painting style and compared to see their pros and cons.

# Method

The oil painting style contains a characteristic visual appearance pattern originated from the oil brush, which differentiates itself from other painting styles. We start from constructing a transformation to simulate this characteristic pattern using traditional image processing techniques.

## Traditional Approach

Since the idea comes from producing oil painting styles by simulating every physical stroke of oil paintings, we first need to determine the locations, magnitudes and directions of all potential brush strokes to be put on the canvas. Based on the knowledge learned during the course, edge detection algorithms provide a superb solution for all these three requirements. Gradients of the image are calculated based on formula 1.



The gradients on both horizontal and vertical axis determine the magnitude and the size of the stroke, denoted as

2

In our method, the pattern of every stroke is approximated by an eclipse, and the magnitude of every gradient determines the length and size of the eclipse. The direction of the eclipse is determined by

3

We evenly distribute every stroke across the entire width and height of the canvas on a predefined brush width, and add some random noise to the stroke order so that every stroke looks different and randomized in both size and location.

To approach more resemblance of manually crafted paintings, the original image is first slightly blurred using a medium filter to smooth out sharpness from real camera-taken images. Then the stroke orders parameterized by brush size are generated with random noise, determining the oil brush stroke locations. Finally different eclipses are constructed on these locations based on edge magnitude and orientation.

## Deep Learning Approach

The traditional method turns out to work quite well on transferring image into an oil painting style, while at the same time having limitations as well, especially in terms of style generalization. In this project, the deep learning methodology is further added to explore the effect of neural network layers on style representation and loss optimization during the style transfer process.

This proceeds our work from unsupervised computation into supervised learning domain, where a desired style example will be required as input, and the model is trained to generates an output image in the style of this input.

The structure and weights of VGG-19 model with the ImageNet dataset is utilized here, comprised of 5 model blocks, with each block containing different number of convolutional layers and a max pooling layer, depicted in Fig 1.

A close up of text on a white background

Description automatically generated

Fig 1: Architecture of VGG-19.

Each convolution layer applies a number of feature filters to extract different features from the input image, and the model summarizes the extracted feature using max pooling layers in a hierarchical manner. The pretrained ImageNet weights enable strong capability in object recognition and feature extraction.

First some random Gaussian noises are added to the pattern image, shown in Fig 2.

A picture containing clock, plate

Description automatically generated

## Fig 2: The original pattern image and after adding random Gaussian noise.

A Fourier transform is applied on the noised image, and the Fourier spectrum is displayed in Fig 3.

A picture containing white, black, sitting, shower

Description automatically generated

Fig 3: The Fourier spectrum of noised pattern image.

A Gaussian lowpass filter and a Butterworth highpass filter are applied on the image in frequency domain. The cutoff frequency is tuned for each filter. Fig 4 shows the results obtained by Gaussian lowpass filter at different cutoff frequencies.

A picture containing clock, old, phone, field

Description automatically generated

Fig 4: Spatial domain results after applying Gaussian lowpass filter with different cutoff frequencies. Top left: 0.01; top right: 0.05; bottom left: 0.1; bottom right: 0.3.

It can be observed that smaller cutoff values generate more blurred outputs, since more high frequency areas are prohibited to pass in frequency domain. These outputs do not work well on removing the noises.

Fig 5 demonstrates the effect of Butterworth highpass filter with different cutoff values.

A picture containing meter, clock

Description automatically generated

Fig 5: Spatial domain results after applying Butterworth highpass filter with different cutoff frequencies. Top left: 0.05; top right: 0.2; bottom left: 0.5; bottom right: 0.9.

Butterworth highpass filter increases the sharpness of the image. With the cutoff frequency rising up, it tends to preserve the prominent features and suppress other backgrounds.

## space.tif

The Fourier spectrum is computed for the space image, displayed in Fig 6.

A picture containing white, playing, holding, player

Description automatically generated

Fig 6: The original space image and its Fourier spectrum.

There’re quite a few noise peaks scattered around the center point in the frequency spectrum, residing in a fixed interval. So we designed a notch array that programmatically covers a dot matrix that corresponds to the majority of prominent noise peaks in the Fourier spectrum. The result images in both spatial domain and frequency domain after applying the notch filters are displayed in Fig 7.

A picture containing white, photo, sitting, shower

Description automatically generated

Fig 7: The filtered space image and its Fourier spectrum.

After applying notch filters on prominent noise peaks, most of the periodic noises are removed. While there are still some subtle noises caused by the tiny peaks existing in Fourier spectrum, which are still not covered by our notch matrix.

Lowpass filters are also attempted for this scenario. A Butterworth low pass filter with different cutoff frequencies are applied on the image, and Fig 8 demonstrates their difference.

A picture containing photo, pizza, cake, rock

Description automatically generated

Fig 8: Result images after applying Butterworth lowpass filter at different cutoff frequencies. Top left: 0.01; top right: 0.05; bottom left: 0.1 bottom right: 0.5.

When lowpass filters are applied on the image, the noises are smoothed out at the cost of blurring the image as well, which is a tradeoff from non-selective filters.

# Conclusion

In this paper, we first present a brief overview of the process of frequency domain filtering, as well as some commonly used filters including lowpass filter, high pass filter and bandpass filter.

Then different type of filters with varying parameters were applied to the pattern image to demonstrate their corresponding filtering effect. And finally we tuned a notch filter and a lowpass filter to alleviate the noises in the space image.

##### References

1. M. Souden, J. Benesty and S. Affes, "On Optimal Frequency-Domain Multichannel Linear Filtering for Noise Reduction," in IEEE Transactions on Audio, Speech, and Language Processing, vol. 18, no. 2, pp. 260-276, Feb. 2010.
2. Soo-Chang Pei and Chien-Cheng Tseng, "Two dimensional IIR digital notch filter design," in IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 41, no. 3, pp. 227-231, March 1994.

##### Appendix

Code to calculate the Fourier spectrum of the pattern.tif:

pattern=imread('pattern.tif');

imshow(pattern)

PQ = paddedsize(size(pattern));

F=fft2(double(pattern),PQ(1),PQ(2));

Fc=fftshift(F);

S1=log(1+abs(Fc));

figure, imshow(S1,[])

Code to apply Gaussian lowpass filter to pattern.tif:

origin = imread('pattern.tif');

pattern = imnoise(origin, 'gaussian', 0, 0.1);

%Determine good padding for Fourier transform

PQ = paddedsize(size(pattern));

%Create a Gaussian Lowpass filter 5% the width of the Fourier transform

D0 = 0.05\*PQ(1);

d\_options = [0.01 0.03 0.05 2];

results = {};

for idx = 1:numel(d\_options)

ratio = d\_options(idx);

D0 = ratio \* PQ(1);

H = lpfilter('gaussian', PQ(1), PQ(2), D0);

% Calculate the discrete Fourier transform of the image

F = fft2(double(pattern), size(H,1),size(H,2));

% Apply the highpass filter to the Fourier spectrum of the image

lp = H.\*F;

% convert the result to the spacial domain.

spacial=real(ifft2(lp));

% Crop the image to undo padding

spacial=spacial(1:size(pattern,1), 1:size(pattern,2));

spacial=uint8(255 \* mat2gray(spacial));

results = [results, spacial];

end

figure

montage(results);

Code to apply Butterworth highpass filter to pattern.tif:

pattern = imread('pattern.tif');

pattern = imnoise(pattern, 'gaussian', 0, 0.1);

PQ = paddedsize(size(pattern));

D0 = 0.05\*PQ(1);

d\_options = [0.05 0.2 0.5 0.9];

results = {};

for idx = 1:numel(d\_options)

D0 = d\_options(idx);

H = hpfilter('btw', PQ(1), PQ(2), D0);

F = fft2(double(pattern), size(H,1),size(H,2));

lp = H.\*F;

spacial=real(ifft2(lp));

spacial=spacial(1:size(pattern,1), 1:size(pattern,2));

spacial=uint8(255 \* mat2gray(spacial));

results = [results, spacial];

end

figure

montage(results);

Code to denoise space.tif using notch filter:

space=imread('space.tif');

imshow(space);

PQ = paddedsize(size(space));

F=fft2(double(space),PQ(1),PQ(2));

%Create Notch matrix, ranging from -500 to 500, excluding the center point.

xs = linspace(-500,500,21);

ys = linspace(-500,500,21);

result = F;

for idx = 1:numel(xs)

for idy = 1:numel(ys)

x = xs(idx);

y = ys(idy);

if x == 0 && y == 0

continue

end

H = notch('btw', PQ(1), PQ(2), 15, x, y);

result = result.\*H;

end

end

% convert the result to the spacial domain.

spatial=real(ifft2(result));

spatial=spatial(1:size(space,1), 1:size(space,2));

figure, imshow(spatial,[])

% Display the Fourier Spectrum

Fc = fftshift(F);

Fcf = fftshift(result);

S1=log(1+abs(Fc));

S2=log(1+abs(Fcf));

figure, imshow(S1,[])

figure, imshow(S2,[])

Code to denoise space.tif using lowpass filter:

space=imread('space.tif');

PQ = paddedsize(size(space));

D0 = 0.05\*PQ(1);

d\_options = [0.01 0.05 0.1 0.5];

results = {};

for idx = 1:numel(d\_options)

ratio = d\_options(idx);

D0 = ratio \* PQ(1);

H = lpfilter('gaussian', PQ(1), PQ(2), D0);

F = fft2(double(space), size(H,1),size(H,2));

lp = H.\*F;

special = real(ifft2(lp));

special = spacial(1:size(space,1), 1:size(space,2));

special = uint8(255 \* mat2gray(spacial));

results = [results, spacial];

end

figure

montage(results);