EECS 332 Final Project

Smart Camera

Human-Background Segmentation with Artistic Style Transfer

Haishan Gao

Submitted to Professor Ying Wu

Table of Contents

Introduction

This report contains the following contents: a summary of the course and an overview of our final project. For the overview of our project, first, we described the tasks and specifications of our project. Then we outlined the design of features of our solution. Finally, we obtained and analyzed our results, and proposed some future developments.

Course Summary

Computer vision is an interdisciplinary field that deals with how computers can be made to gain a high-level understanding from digital images or videos. The field of Computer Vision is a field with a large number of possible applications that can be used in daily life. In general, it provides the technology and methods used to compute imaging-based automatic inspection and analysis. Such application range from tasks such as industrial machine vision systems which inspect bottles speeding by on a production line, to research into artificial intelligence and computers or robots that can comprehend the world around them.

This course has given us a preliminary understanding of Computer Vision fields, methods, tasks, and applications. During the first week, we were given an introduction to binary image processing: binary images were used to detect segment with connected-component labeling. The next topic was around morphological operators such as erosion, dilation, opening and closing. With these basic operators, we were able to remove noise from the input image and detect the boundary of a segment. While the details of an image might be too trivial for human eyes, we then learned how to apply histogram equalization to process the image. Given the concept of histograms for all grayscale images, by summing and normalizing to acquire the equalized histogram, more details of the image could be seen and human eyes can detect more trivials.

After binary image processing, we were introduced to colored image processing. The first part is a fundamental introduction into color spaces as RGB, YMC and HSI. Our first interesting and inspiring problem was to implement a color-based segmentation which can be applied to skin color detection. We then utilized this concept to learn different types of edge detector and

implement Canny detector by ourselves. Furthermore, Hough Transform was introduced to detect lines/directions.

During the last phase of the course, we were introduced to more advanced topics. Associated with computational photography, image formation and camera model were introduced to us. Moreover, we learned about face motion and image stitching which integrated much knowledge from the course to implement. Finally, we learned about feature extraction and image tagging, the current hot topics in Computer Vision. Except for effective and simple solution such nearest neighbor, we also knew that Deep Learning techniques could be applied, which will be a focus in the advanced course EECS 432.

Project Description

The problem we want to solve in this project is that sometimes users want to post a photo on social media but they are at their homes that haven't been cleaned for weeks, with mountains of clothes scattered on the bed and cabbage on the floor. To cope with such situations, our team wants to create an application that can simply capture the user's real-time image and replace his/her background image. In addition, we would like to replace the background with something fancy, like a painting-style scenery. We also want to integrate the knowledge we learned in class to implement a function that enables the user to change the background with simple gestures in front of the camera without physical interactions with the computer.

In conclusion, the goal statement of the project is to implement an application that could enable the user to change his/her current background to artistic style sceneries using simple gestures.

Design Overview

To achieve the goal statement that we defined earlier, our application should include the following functions:

1. Segmentation of the user and his/her background in a video
2. Hand poses recognition and gesture control
3. Artistic style transfer to background images

In the next sections, this report will introduce and explain how we implemented these three different functions including both algorithm and result.

1 Segmentation

1 Relevant Definition

A graph $G = \langle V, E \rangle$ is an organic whole consisting of a set of points V and a set of edges E. Among them, the point set V is composed of a series of vertices, and the edge set E is composed of a series of edges connecting the vertices. If there exists an edge (p, q), then the vertex *p, q* is called adjacent. If every edge (p, q) all has its corresponding edge (q, p) in the graph, the graph is called an undirected graph, otherwise, it is called a directed graph. If there are always a series of edges (p, p_1), (p_1, p_2), … (p_{n-1}, p_n), (p_n, q) which connect any two vertices in the graph, the graph G is called a connected graph, otherwise the graph G is separated. If the set C is a subset of the edge set E of the Uniform graph G and satisfy that $G(C) = \langle V, E - C \rangle$ is separate, it is called a cut of the graph G. According to the needs of the application, each edge e of the graph can be given a weight w_e, and the graph given the weight is the weight map.

In the weighting graph, you can define the cost of the cut as $\| C \| = \sum_{e \in C} w_e$. The cut with the least cost is called the minimum cut. Solving the minimum cut of the weighted graph is a classic problem in the combined number theory. There are low-order polynomial time algorithms (such as the maximum flow minimum cut algorithm), which can be solved efficiently.

2 Interactive Image Segmentation

Mathematically, the user's input divides the entire picture into three parts: foreground (OBJ), background (BKG), and unknown (UNKNOWN). The task of the algorithm is to obtain a segmentation of the unknown portion of the pixel that satisfies the energy (cost) in a certain sense, that is, to determine whether each pixel belongs to the background or foreground. Generally, the energy of image segmentation includes two aspects, which respectively reflect the regional and boundary properties of the image, which can be called as regional energy and boundary energy respectively. Let the set of pixels of an image be P, and the set of all adjacent

pixels be recorded as N(ie, p and q are adjacent equals to $\{p,q\} \in N$). Set $A = (A_1, ..., A_p, ..., A_{|P|})$ to a split of the image. And $A_p = OBJ$ means that the pth pixel belongs to the foreground and $A_p = BKG$ means that the pth pixel belongs to the background. The total energy, regional energy, and boundary energy of the image segmentation are respectively defined as $E(A)$ 、 $R(A)$ and $B(A)$, then we can get:

$$E(A) = \lambda \cdot R(A) + B(A)$$

$$R(A) = \sum_{p \in P} R_p(A_p)$$

$$B(A) = \sum_{\{p,q\} \in N} B_{\{p,q\}} \cdot \delta(A_p, A_q)$$

$$\delta(A_p, A_q) = \{1 \quad A_p \neq A_q \; 0 \quad A_p = A_q$$

Parameter λ is used to reflect the relative importance of regional energy and boundary energy. The larger the λ is, the greater the proportion of the regional energy in the total energy, on the contrary, the greater the proportion of the boundary energy in the total energy. The area energy $R(A)$ reflects the cost of splitting each pixel p into A_p . The smaller the $R_p("BKG")$ is, the more similar it is to the user-specified background pixel; the smaller the $R_p("OBJ")$ is, the more similar the pixel p is to the user-specified foreground pixel. To measure the similarity of a pixel to a user-specified background pixel and foreground pixel, a statistical model is usually used. Here, the user's input acts as a statistical source: it tells the program what the background is and what the foreground looks like. For monochrome images, gray histograms are generally used as statistical models. For color pictures, the overhead of statistical histograms is too large. The commonly used statistical model is the Gauss Mixture Model (GMM). The boundary energy $B(A)$ reflects the cost of any two adjacent pixels p and q in the set of pixels, when the segmentation is discontinuous, that is, $A_p \neq A_q$. The more similar p and q, the larger the 则 $B_{\{p,q\}}$; the opposite is the smaller. Generally, taking $B_{\{p,q\}} = e^{-\beta \| c_p - c_q \|}$, where c_p, c_q is the color of the pixel p and q respectively.

Ultimately, interactive image segmentation is translated into the following optimization issues.

$$E(A) = \lambda \cdot R(A) + B(A), subto: \quad A_p = \{BKG \quad if\ p \in B \quad OBJ \quad if\ p \in O .$$

Where O、B are the foreground and background sets entered by the user respectively.

3 Algorithm implementation

3.1 Graph Cut

The graph cut algorithm can be used to efficiently solve the optimization problem proposed in the previous section. A map $G = \langle V, E \rangle$ can be defined for this purpose. And $V = P \cup \{S, T\}$ is its vertices set. P corresponds to the pixel set of the image, and S, T are two terminals. Side set. Where is a collection of all adjacent pixels in the image. The weights of the edges are specified in the following table:

Edge	weight	condition
$\{p, q\}$	$B_{\{p,q\}}$	$\{p, q\} \in N$
$\{p, S\}$	$\lambda \cdot R_p("BKG")$	$p \in P - (O \cup B)$
	K	$p \in O$
	0	$p \in B$
$\{p, T\}$	$\lambda \cdot R_p("OBJ")$	$p \in P - (O \cup B)$
	0	$p \in O$
	K	$p \in B$

Among them,

$$K = 1 + \max_{p \in P} \sum_{q:\{p,q\} \in N} B_{\{p,q\}}$$

It can be shown that the minimum cut $\hat{C}$ of the graph G corresponds to the optimal segmentation of the image. If $\{p,S\}\in\hat{C}$, the pixel p is divided into backgrounds; otherwise, there must exist $\{p,T\}\in\hat{C}$, and the pixel p is segmented into the foreground.

Proof: First define the concept of feasible segmentation. If F is a cut of the figure G, it satisfies:

- For any vertex p, F include one of $\{p,S\}$ and $\{p,T\}$;
- If and only if p、q connected to different terminal points, $\{p,q\}\in F$;
- If $p \in O$, $\{p,T\} \in F$;
- If $p \in O$, $\{p,S\} \in F$.

The feasible segmentation of the graph G is one-to-one correspondence with the segmentation of the image P. In fact, for feasible segmentation F, the corresponding image segmentation $A(F)$ can be defined as

$$A_p(F) = \{BKG \qquad \{p,T\} \in F \quad OBJ \qquad \{p,S\} \in F$$

It is easy to prove that the minimum cut $\hat{C}$ of the graph G is a feasible split. The cost of any feasible segmentation F of the graph G is:

$$\begin{aligned}\|F\| &= \sum_{p\notin O\cup B} \lambda\cdot R_p\left(A_p(F)\right) + \sum_{\{p,q\}\in N} B_{\{p,q\}}\cdot\delta\left(A_p(F),A_q(F)\right)\\ &= E(A(F)) - \sum_{p\in O}\lambda\cdot R_p(\text{"OBJ"}) - \sum_{p\in B}\lambda\cdot R_p(\text{"BKG"})\end{aligned}$$

That is, $\| F \| = E(A(F)) - C$, So H is, among them, a collection of all image partitions that satisfy the constraints of user's input.

3.2 Grabcut algorithm

Due to the rectangle in the foreground, we are unable to directly obtain the GMM statistical samples of the foreground. Therefore, iterative and incomplete labeling methods need to be used to solve the problem step by step.

The method divides pixels into three categories, foreground region (O), background region (B), and unknown region (U). The pixels in U are classified into O and B one by one. The GMMs are calculated for each of the three types of areas, and the color information is counted. For each GMM, we take the number of front background components as K=5.

Therefore, the formula shown as above can be redesigned as follows:

$$E(\underline{\alpha}, k, \underline{\theta}, z) = R(\underline{\alpha}, k, \underline{\theta}, z) + B(\underline{\alpha}, z)$$

The regional energy term is:

$$R(\underline{\alpha}, k, \underline{\theta}, z) = \sum_{n} R_n(\alpha_n, k_n, \underline{\theta}, z_n)$$

$$R_n(\alpha_n, k_n, \underline{\theta}, z_n) = -\log log\, \pi\ (\alpha_n, k_n, \underline{\theta}, z_n) + \tfrac{1}{2} \sum n\ (\alpha_n, k_n) + \tfrac{1}{2}[z_n - \mu(\alpha_n, k_n)]^T \sum(\alpha_n, k_n)^{-1} [z_n - \mu(\alpha_n, k_n)]$$

Where α_n is the attribute of the pixel n, $\alpha_n = 0$ represents the foreground, and $\alpha_n = 1$ represents the background. k_n represents the Gaussian mixture model label to which the pixel point n belongs. The parametric model of the Gaussian mixture model is represented by $\underline{\theta} = \{\pi(\alpha, k), \mu(\alpha, k), \sum(\alpha, k), \alpha = 0, 1, k = 1...K\}$.

The boundary energy term is:

$$B(\underline{\alpha}, z) = \sum_{(m,n) \in C} \delta(m, n) \exp exp\ -\ \beta \parallel z_m - z_n \parallel^2$$

According to the above definition, we can solve it by an iterative method.

1. Define the background set T_F as the outer part of the rectangle, the foreground set $T_O = \varphi$, and the unknown area $T_U = \bar{T}_B$. The attribute $\alpha_n = 1,$ and the remaining pixels $\alpha_n = 0$. The GMM is separately calculated for two types of pixels, $\alpha_n = 1$ and $\alpha_n = 0$.

2. Estimate the GMM component in which the pixel n in T_U is located:

$$k_n = \arg arg\ \min_{k_n}\ D_n(\alpha_n, k_n, \theta, z_n)$$

3. Obtain the GMM parameter $\underline{\theta}$:

$$\underline{\theta} = \arg arg \min_{\underline{\theta}} (\underline{\alpha}, k, \underline{\theta}, z)$$

4. Estimated segmentation by minimum cut method:

$$\min_{\{\alpha_n : n \in T_U\}} \min_{k} E(\underline{\alpha}, k, \underline{\theta}, z)$$

5. Repeat step 2 until it converges.

4 experiments and results

4.1 The experiments on a single photo

We tested the actual effect of this algorithm in processing a single photo with different foreground and background. The results are shown below. We observe that the results are decent and could exhibit good performance.

Figure 1 The original photo with white and bright background

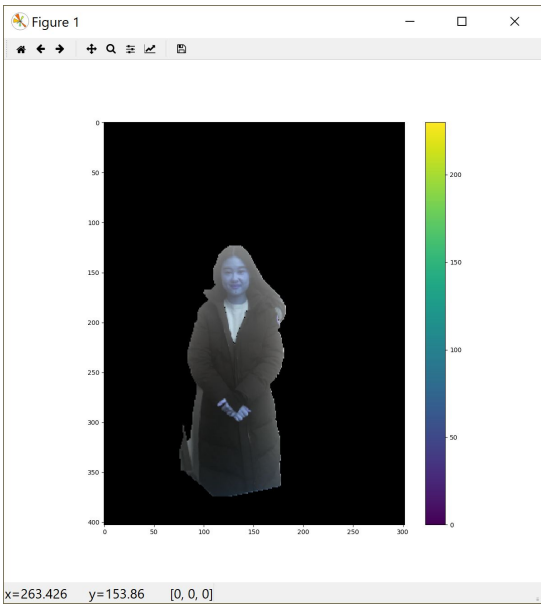

Figure 2 Split photo with white and bright background

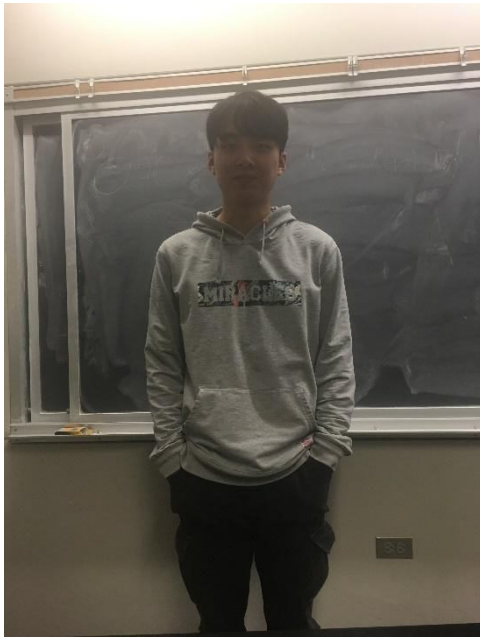

Figure 3 The original photo with dark background

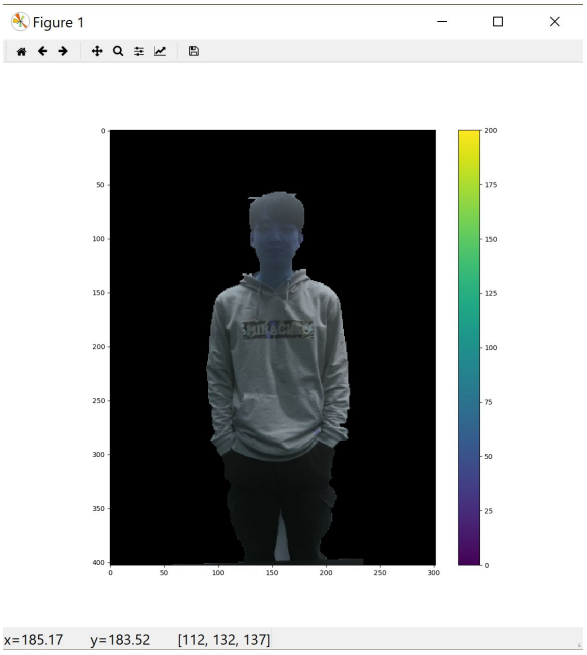

Figure 4 Split photo with dark background

4.1 The experiments on video

We also tested the effect of this algorithm in video processing. The results are shown as below. The upper image is the real-time video and the lower is the processed video with segmentation that filled the background with plain white. We observe that the results are clear enough for further processes.

Figure 5 Segmentation in video

2 Finger Recognition

1 Background subtraction

Background checkout is a very important step in basic finger recognition applications. We can only deal with the gesture after extracting the hand or the finger separately.

Technically, we need to extract the foreground of the movement from the still background. If you have a background (only the background does not contain the foreground) images, such as no customer's room, no means of transportation, etc., then it is easy. We only need to subtract the background from the new image to get the foreground object.

But in this cases, we have to deal with the video, which means that we can hardly get such (background) images, so we need to extract the background from the frames we have. If the vehicle in the image still has a shadow, then the work is even harder, because the shadow is also moving, and just using subtraction will treat the shadow as a foreground. It's a very complicated thing. In order to achieve this goal, we try to use the BackgroundSubtractorMOG2 algorithm which are already included in OpenCV.

2 Skin Detection

After extracting the foreground separately, we need to detect the gesture of a hand.

To detect it, we extract the color of body skin from samples. Then we are able to recognize the gesture of fingers. The same processes of Machine Problem 4 were implemented here.

3 Gesture detection and recognition

Using the convexDefects pit detection function provided by opencv to detect the concave points of the image, and then use the cosine theorem to calculate the angle between the two fingers according to the coordinates of the (starting point, ending point, and far point) in the concave point. It must be an acute angle, and the gesture is discriminated according to the number of acute angles.

Where the number of acute angles is 0, indicating that the gesture is a fist or one.

The number of acute angles is 0, indicating that the gesture is a fist or one.

The number of acute angles is 1, indicating that the gesture is scissors

The number of acute angles is 2, indicating that the gesture is three.

The number of acute angles is 3, indicating that the gesture is four.

The number of acute angles is 4, indicating that the gesture is five.

4 Gesture Control

As long as the gesture can be detected, the corresponding control can be defined easily. We designed the following four types of actions:

When the number of fingers is two, the replacement of the background will start and the background will turn to next one;

When the number of fingers is three, the background will turn to former one;

When the number of fingers is four, the style of the background will change to next one;

When the number of fingers is five, the style of the background will change to former one.

5 Experiments on Video

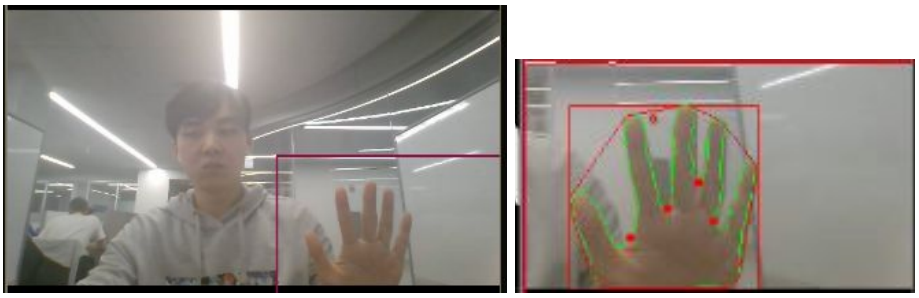

Figure 6 Finger recognition

As the above screenshot shows, the program can detect the number of fingers explicitly. The left image is the real-time video. Users have to show their hands inside the red box in the right-bottom corner so that the program can detect the hand. The right image is the result

returned by the program: it is very clear that four acute angles is detected as the red dots which represent “five fingers”.

We observed that the results are clear enough for gesture detection and we further associate different gesture controls with them as described in section 4.

3 Artistic Style Transfer

1 Introduction

To obtain painting-style scenery as background images for our project, we decided to use deep learning technique – Convolutional Neural Networks (CNN) – for the processing. Nowadays we figured that CNN is especially good at image processing while the reason remains mysterious to us.

For implementation, we need to first set up the model and we will use a style image for obtaining the transferred style and a content image which we want to transfer the style onto. The result of the project can demonstrate how well machine learning techniques perform on image processing and how artificial intelligence can, to some extent, achieve considerable success in the field of art and creativity.

2 Methodology

We use the pre-trained VGG19 convolutional neural network model from Keras library in Python for our implementation due to its stability and reliability. The model requires a style image and a content image as inputs, and a noise image for optimization. During the optimization process, the code runs scipy-based optimization (L-BFGS) over the pixels of the generated image to minimize a loss value calculated by weighting content and style losses (the representation of “loss value” is introduced in the next section). Finally, the noise image will be optimized into a synchronized image which has content of the content image and a similar style from the style image.

2.1 Feature representation

The style or content representation of an image is measured by the output of a specific layer of the neural network. We have iterated through several layers to find the optimal one for output. In order to measure the similarity of the style/content of two images, we further define values "style loss" and "content loss" between the processed image and the input images, with less "loss" meaning the images more similar in style and content, respectively. Content loss is measured by passing both the synthesized image and the content image through some layers of the model and finding the Euclidean distance between the intermediate representations of those images; while style loss is measured similarly by comparing instead the Gram matrices of the outputs at various layers. (A Gram matrix results from multiplying a matrix with the transpose of itself and contains non-localized information about the image, such as texture, shapes, and weights - which we defined as "style" representation in our case.)

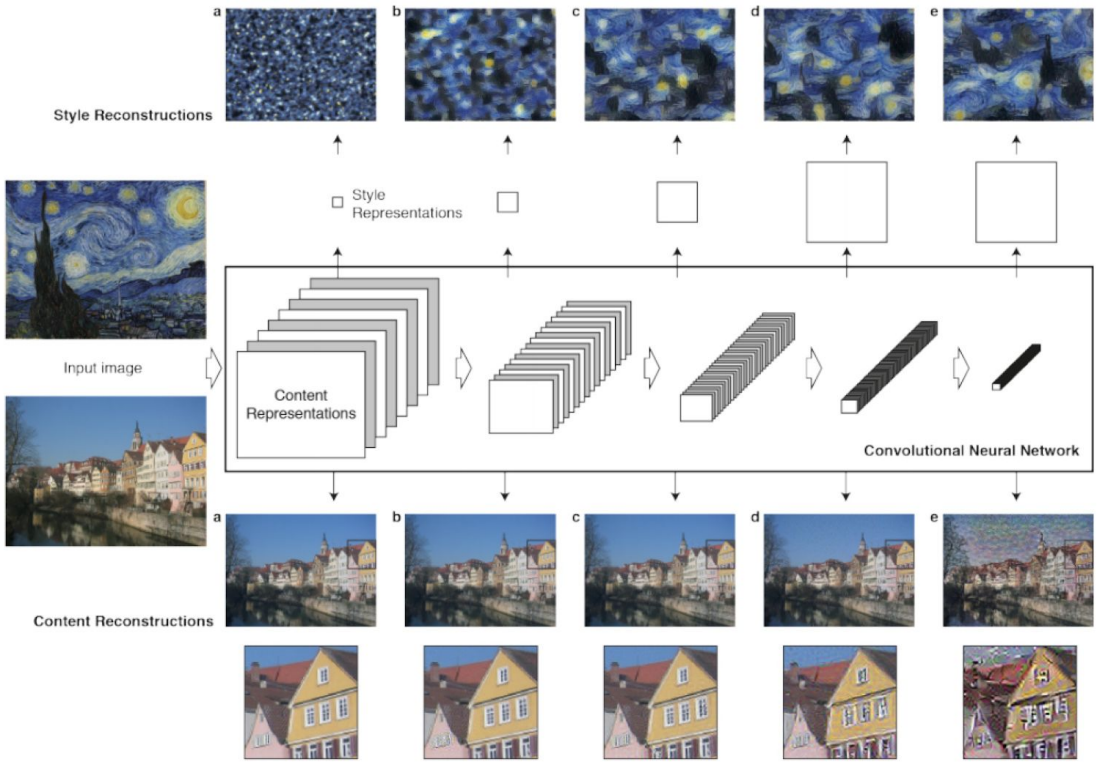

Figure 7 Content and Style Construction in CNN

2.2 Data collection

The data set consists of two types of images: style image and content image. We collected approximately one hundred representative paintings and selected 40 of them as the style image data set for our project. We then assigned them into 8 different categories: abstract art, Chinese paitings, Japanese Ukiyo-e, portraits, still, sceneries, buildings and illustrations. For content images, we include two portraits and two sceneries since we observed that the results might differ when we used different images as content image. We applied 40 style images for each content image and we ran 5 iterations of optimization for each pair. Hence, we obtained 800 photos in total in out final data set.

3 Result

In summary, we found that convolutional neural network performed very well on this task due to its outstanding performance on analyzing visual imagery. We have acquired decent synthesized images using CNN model with tuned parameters.

Figure 8 Kanagawa Waved Deering

Figure 9 Le Rêve Robert Downey Jr.

Since CNN utilized Max Pooling as downsampling strategy, when the image contains large contrastive color blocks in different areas, CNN will perform the best. In our examples, when the style image has big blocks of colors such as Le Rêve by Picasso in Figure 9, we observed the most decent results. On the contrary, if the style image is too colorful with small blocks of various colors as in Figure 10, the synthesized image will contain messy strokes and the layout of the colors will be chaotic.

Figure 10 Fragmented Deering

According to our observation, we recommend selecting style images with simple outlines and big blocks of colorful blushes for artistic style transfer and avoiding using paintings with many small blocks of colors such as Broadway Boogie-Woogie by Piet Mondrian in Figure 10.

Result and Analysis

After we implemented the three distinct functions according to the design overview, we can integrate them together as introduced in the project description to achieve the goal statement.

The final product is an integrated program. After the launching of the program, users can view the real-time video on the computer screen as Figure 11. The program will also show the segmented version of the video below.

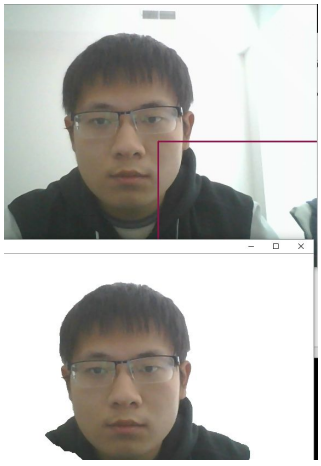

Figure 11 Program Interface

Users can then show their gestures in the right-bottom red box for different instructions: specifically, the gestures “Two fingers” and “Three fingers” change the background image, and the gestures “Four fingers” and “Five fingers” change the style of the current background image.

Following are some results obtained from the program:

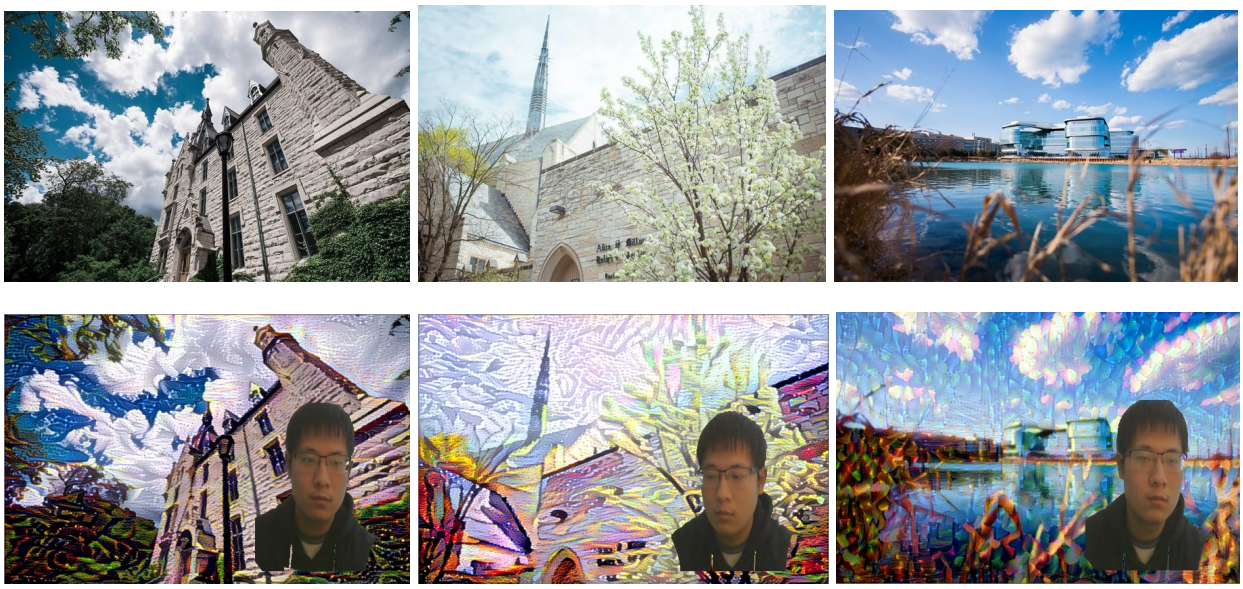

Figure 12 Segmentation with Artistic Style Images Background

The upper rows are the original background images. The lower rows are the result images of the program.

As a result, we observe that our program was very successful in segmenting users from the background and in identifying users' gestures as different instructions to interact with the program. However, there are still aspects we can improve for the final results. First, the user is "floating" in the integrated images. Second, we set the frame rate of the video to be very slow (1 frame per second) to give the program time to process user gestures, which makes the program very slow.

Future Steps

To further improve the program, we want to work on the following aspects. First, we want to improve FPS to be at least five frames per second. To achieve this and ensure the correctness of gesture recognition, we have to improve the robustness of finger recognition. Second, we want to make the combination of the user's foreground image and background image more convincing and beautiful. To implement this improvement, we can either modify the segmentation process to cut the lower part of the image where there are no or very little foreground pixels, or modify the pasting process to find a better place to paste the foreground image to the artistic background.

Above summarized the preliminary improvement to the projects. More advanced topics might be to reduce the processing time of artistic style transfer or to improve the algorithm of segmentation. To better improve this specific project, I hope the course could include more contents on video processing.

References

1. https://blog.csdn.net/zouxy09/article/details/8534954
2. https://blog.csdn.net/zouxy09/article/details/8535087
3. http://www.demodashi.com/demo/12968.html
4. Course Machine Problem
5. A Neural Algorithm of Artistic Style by Gatys, Ecker and Bethge (2015)

Course Feedback

This course has given me the preliminary understanding on the different fields and the basic concepts of Computer Vision. Moreover, I learned about the mathematical principles behind the important concepts in image and video processing. Overall, This course serves as a satisfactory introduction to Computer Vision for students who are interested in this topic.

Professor Wu is passionate about the topics covered in class and he always used some interesting examples to attract us and illustrate the concepts. Moreover, he was very kind and patient when we sought for advice on our final projects. He gave valuable suggestions to us and inspired us on the direction of the project.

On the aspect of the course organization, machine problems have very clear purpose and they are greatly designed to help us refresh the memory as well as better understand the knowledge we learned from the class.

One of the possible improvement is that this course includes a large amount of math. For someone like me who does not have a very strong background in Mathematics, sometimes I might feel difficult to catch up with course speed. Hence, I hope there could be weekly recitation sessions that can help students to learn about the principles. Also, since I have course conflict with TA's office hours this quarter, I am not able to get help right away sometimes. I think it would be great if the course can have a Piazza website where we can post our questions and get answers.

Personally, I really like this course and I will recommend it to my friends who have interests in Computer Vision. The course itself include a wide range of topics in depth and it also provides students the capability to implement their whimsical ideas using techniques they learned from the course.

Appendix A: Code

This appendix outlines the code used in the project.

The main program basically consists of two part: a) the interaction program including segmentation function/hand poses, b) A program that obtains the painting-style background images.

Part I: Segmentation and Finger Recognition

The environment we used is Python 2 and OpenCV 3. For full code please visit: https://github.com/sally9805/Smart-Camera

Part II: Artistic style transfer

The environment we used is Python 2 in Jupyter Notebook.

Please refer to the source code here: https://sally9805.github.io/Neural-Artistic-Style-Transfer/Final_project_Style_transfer.ipynb