Final Project: Implementing a real-time Image Processing Demo with a GUI

K-Means Clustering: Image Contour on human's head and body

Faculty advisor:
Prof. Ming-Ting Sun

Course:
EE568 Digital Image Processing

Student:
Sheng-Hao Chen
2027314

University of Washington
Seattle, WA-98105

March 2021

Winter Quarter Final Project Report

Objective

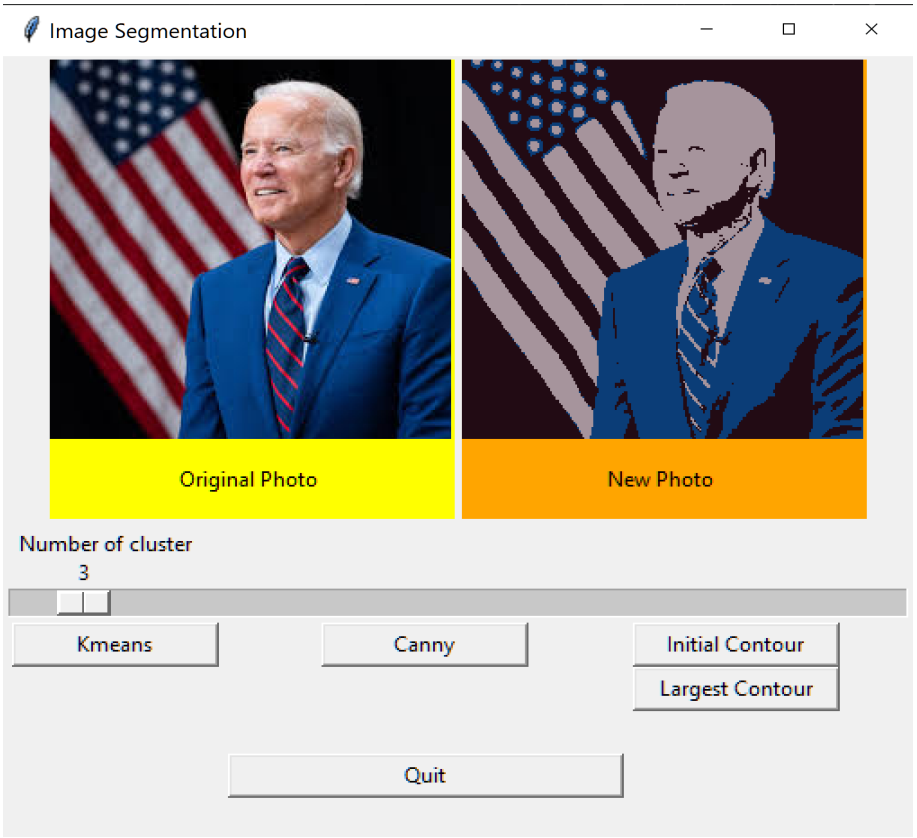
Computer vision is quite a big topic in the modern world, because the improvement of hardware and optimized algorithms help people start to use the techniques to enhance people's life. In the topic, I try to use K-mean clustering pictures to look for obvious targeted areas. For a human, I sucessly find his/her head and body and contour the areas. The results generated by canny edge detection and threshold determination after clustering by K-means.

Instructions

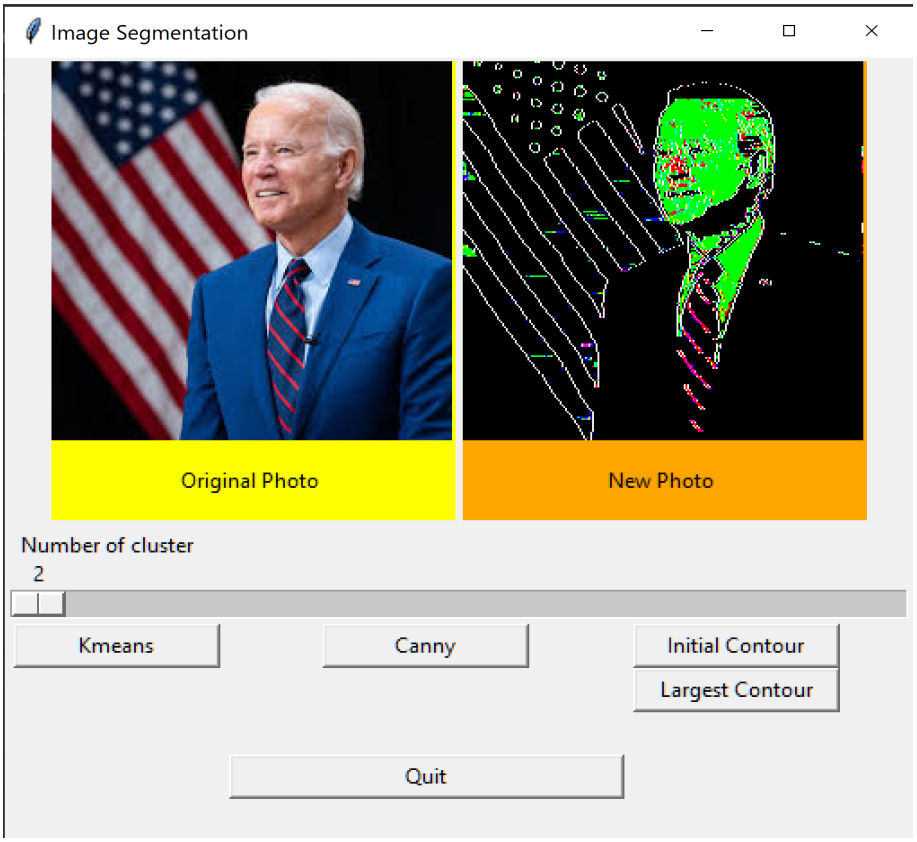
Execution: The program is completed by **Python** scripts. All the data are in the same documentary. Switch to the documentary and type command "**python find_contour_kmeans.py**", then the window will jump out and you can start to use it.

The interface includes 2 images, one scroll bar, 5 buttons. The image at the left is the original image and the image at the right is the new image after processing. The scroll bar is under the 2 images. We can use a scroll bar to adjust the number of clusters. After sliding the bar and getting a value, we can click one of the four buttons, **K-means, Canny, Initial Contour, Largest Contour**.(Please click only once after sliding the bar) The new image will change to the corresponding image by your choice. Please refer to below four figures with title image segmentation to know the effects.
The last button is "**quit**" which we can click to close the program and the window.

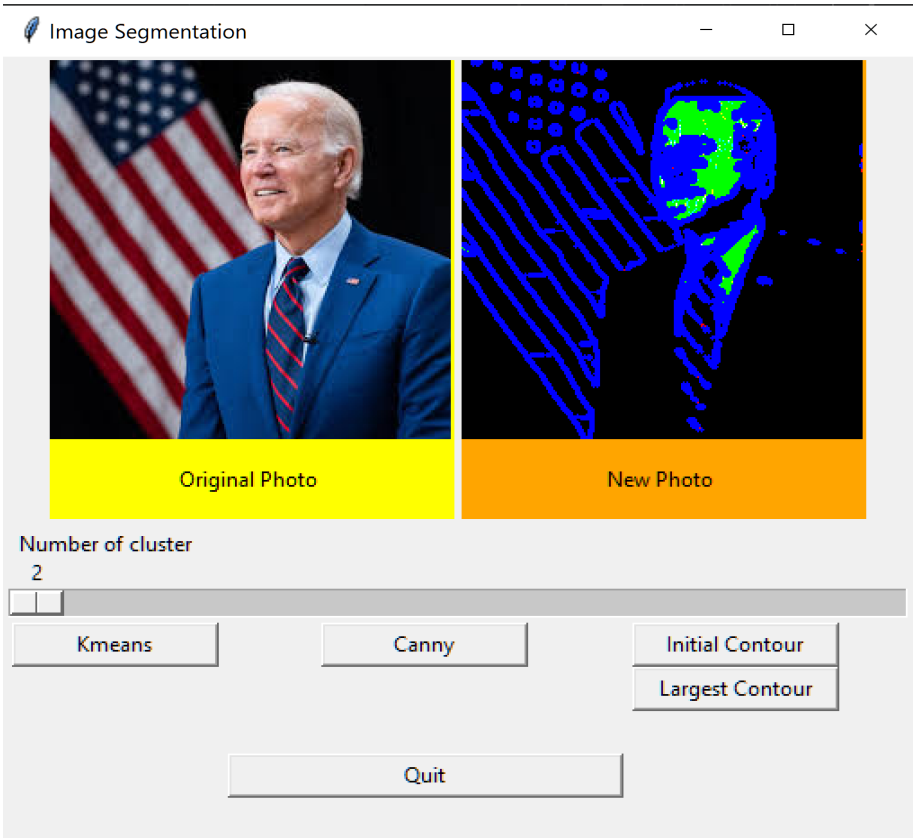
K-means



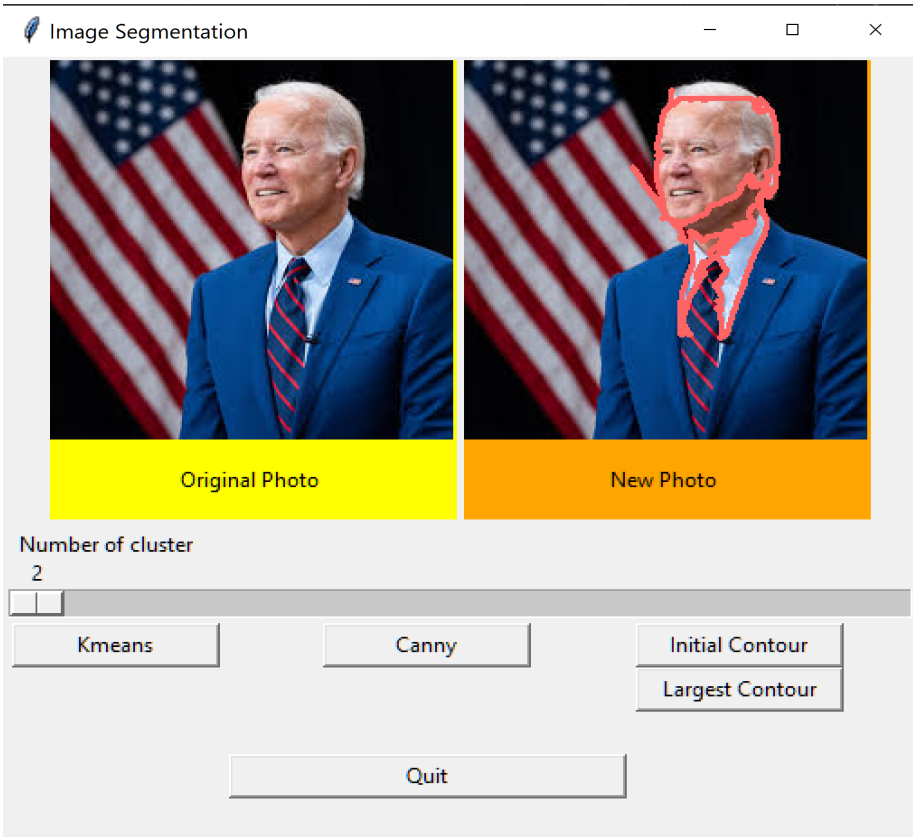
Canny Edge Detection



Initial Contour



Final Contour



Implementation

GUI: Python tkinter libary

Using tkinter in Python gives me an efficient way to reach my goal to design an interface. There some steps below

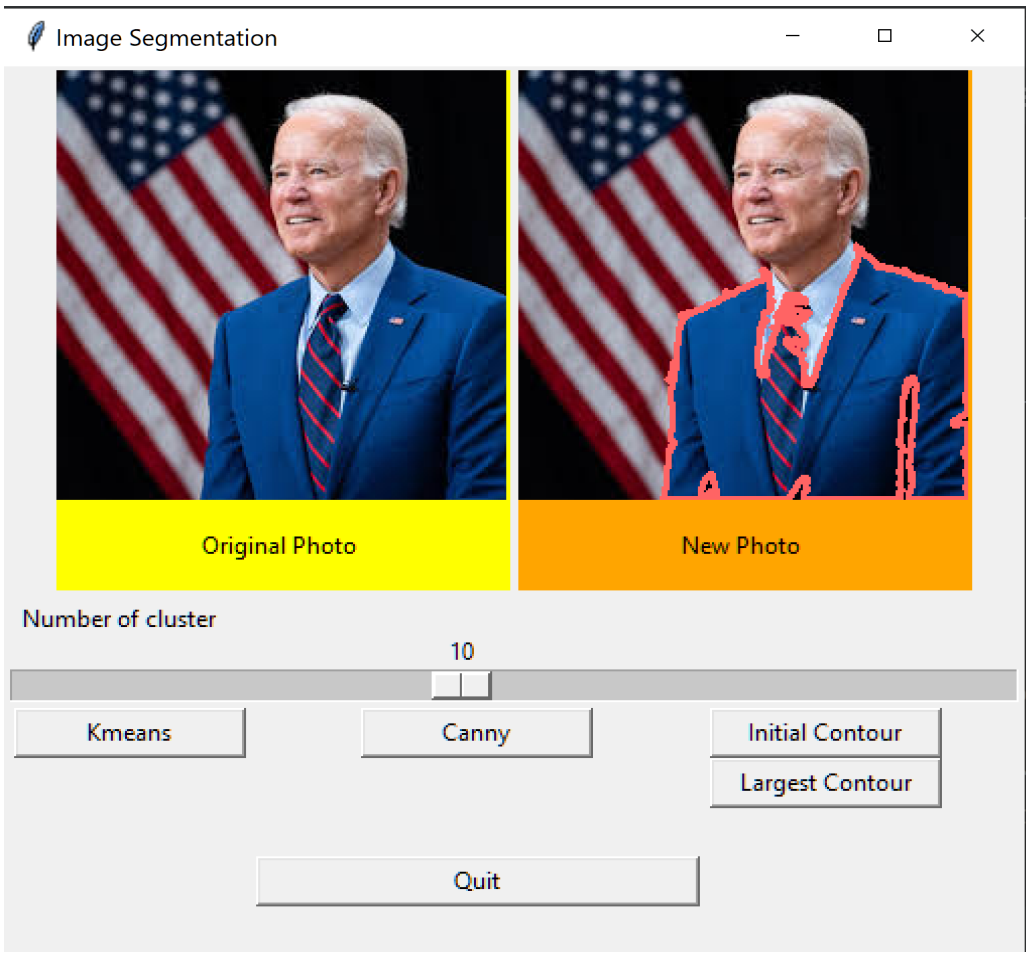
1. Set up a window interface and title.
   self.window = window
   self.window.title(window_title)
2. Create a frame that can fit the original and new image.
   self.frame1 = tk.Frame(self.window, width=self.width, height=self.height, bg='blue')
   self.frame1.pack()
3. Create canvases for original(LEFT) and new(RIGHT) image.
   *self.canvas0 = tk.Canvas(self.frame1, width=self.width, height=self.height * 1.2, bg='yellow')*
   self.canvas0.pack(side=tk.LEFT)
4. Use PIL to convert the NumPy ndarray to a "PhotoImage".
   self.photoOG = PIL.ImageTk.PhotoImage(image=PIL.Image.fromarray(self.cv_img))
5. Add images to canvas.
   self.canvas0.create_image(self.width // 2, self.height // 2, image=self.photoOG)
6. Create another frame for button space.
7. Create a slide bar(**tk.scale**) and buttons. The slide bar should use a command to call the image processing function.
   self.scl_cluster = tk.Scale(window, from_=2, to=20, orient=tk.HORIZONTAL, showvalue=1, command=self.kcluster, length=500, label="Number of cluster")
8. Use **tk.button** to create buttons and use **button.grid** to position the buttons.
   self.button_k = tk.Button(self.frame2, text="Kmeans", width=15, command=self.kcluster) self.button_k.grid(row=0, column=0, padx=5)
9. **self.window.mainloop(),** an important part which let the interface show up**.**

Image Processing

There is one class "**kmeanskernal**". In the class there is an initial function to set up default parameters and GUI design code. The following functions are **kcluser, canny, iniContour, bigContour, and contour. "kcluser"** is the task of grouping pixel into two or more groups based on the value of the pixel. Here we can set up iteration times and the number of data. Default is 10. The **canny** function is to find the intensity gradient of the image. filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction and vertical direction. Gradient direction is always perpendicular to edges, so I use it to find the edges. Moreover, I use **cv2.contour** methods to draw contours. In this way, it would draw too many contours. To find the largest contour, I use cv2.threshold to split data by pixel intensity with a certain max and min value. Then, find the largest contour and output it(**button: largest contour**).

Results

I use New United States president **Joe Biden** photo as an example(also **Donald John Trump and Chou Tzu-Yu photos** attached). When I click K-means, we can see the output image change to a couple of image blocks based on how many clusters I choose. If the clusters are large, the image would be similar to the original image, because it can be presented by more colors. By clicking **Canny**, we can see edges, but it also depends on how many clusters we choose. If the number of clusters are large, the edge lines are more complicated. For clicking **Initial Contour,** we can see **the thicker edges after canny function is applied.** Finally, we can try different clusters to find the president's head or body after searching the largest contour.



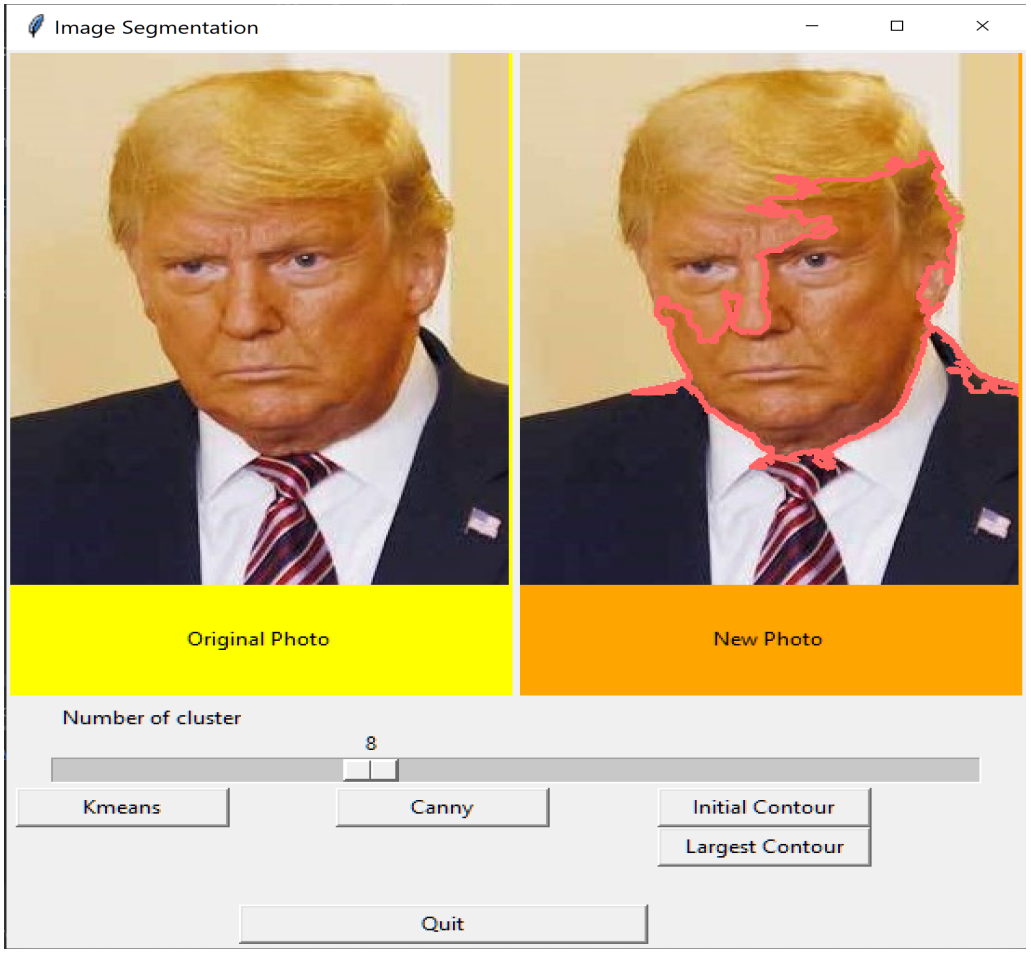
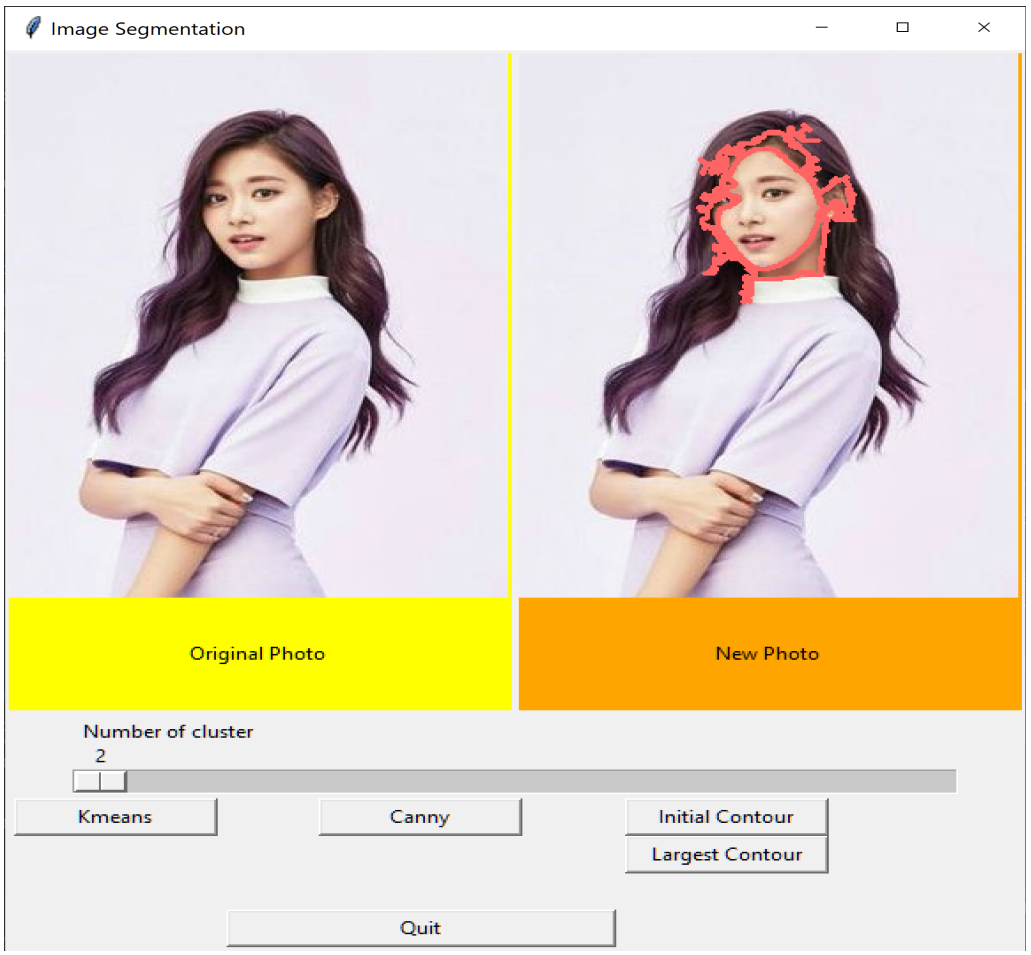


Image Segmentation
Original Photo
New Photo
Number of cluster
8
Kmeans
Canny
Initial Contour
Largest Contour
Quit



Future work

The program is still under optimization. The call function method I designed is not smart enough. If I click the same button again and again, it would execute again by the previous result. If I would like to acquire an accurate result, please bear in mind, just click the function button once after sliding the bar, don’t click buttons continuously. In the future, I will fix the bug and add more elements like squaring the person's head and showing its coordinate real-time.

Reference:

1. “OpenCV - K-Means Clustering”
   https://docs.opencv.org/3.4/d1/d5c/tutorial_py_kmeans_opencv.html
2. “OpenCV - Canny Edge Detection”
   https://docs.opencv.org/master/da/d22/tutorial_py_canny.html
3. “OpenCV - Image Thresholding”
   https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html
4. “OpenCV - Contours : Getting Started”
   https://docs.opencv.org/master/d4/d73/tutorial_py_contours_begin.html
5. “K-Means Clustering Explained: Algorithm And Sklearn Implementation”
   https://programmerbackpack.com/k-means-clustering-explained/
6. “Finding contours using opencv”
   https://programmerbackpack.com/k-means-clustering-for-image-segmentation/
7. “tkinter — Python interface to Tcl/Tk”
   https://docs.python.org/3/library/tkinter.html
8. “United States President”
   https://www.whitehouse.gov/