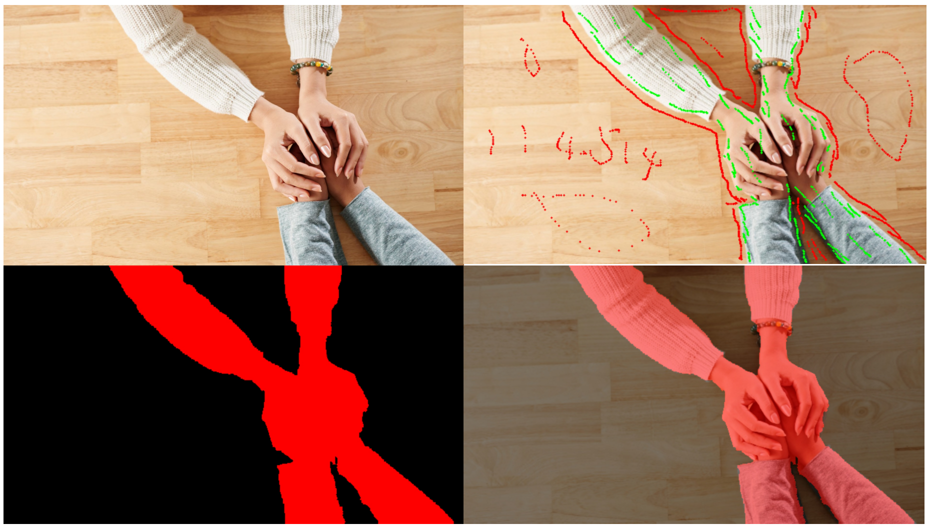
CS270 Homework 3 Report

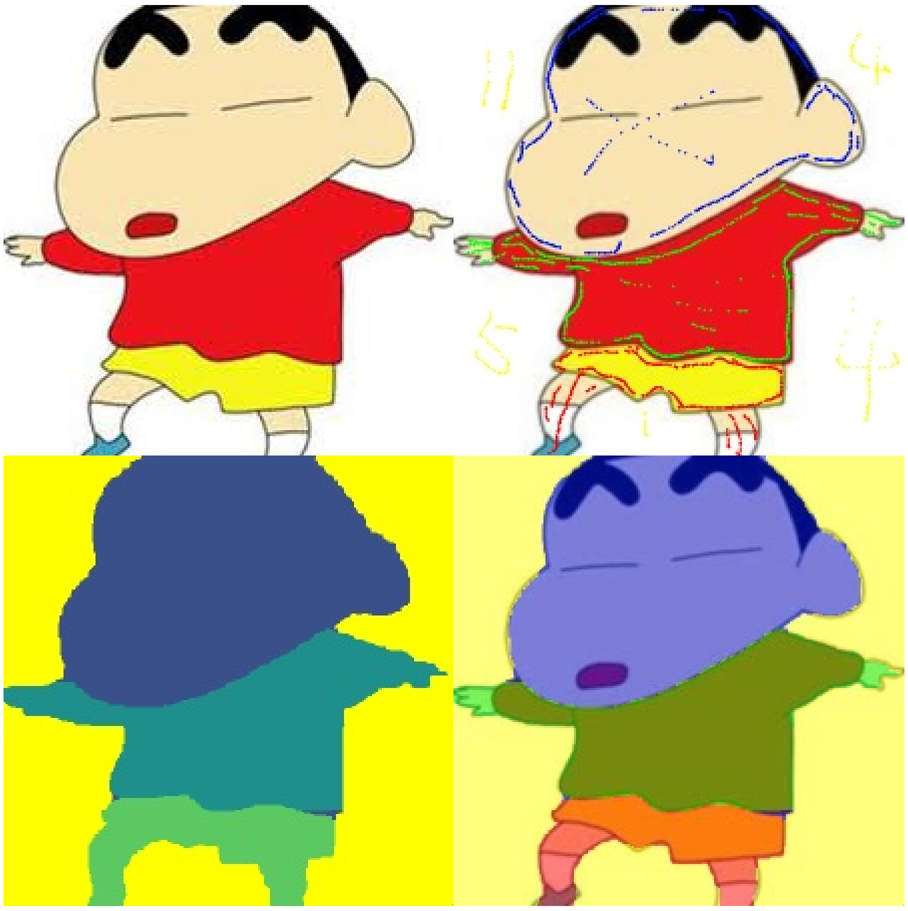
任怡静 2018533144

Question 1: Graph Cut for Image Segmentation (50 points)

- Please describe your algorithms in words or flowcharts. (15’)
  - **The Seed Collecting**
    - This part used the OpenCv built-in functions to catch mouse operations to append coordinates to seed lists (**foreground_seeds** and **background_seeds**) and illustrate seeds on image
    - The image size is adjusted for drawboards so that user can draw accurately
  - **The Cut Graph Algorithm**
    - I first calculate a graph from seeds obtained in the previous step, which assign 0 to background_seeds' corresponding coordinates in **graph**, 1 to foreground_seeds' and 0.5 for the rest.
    - Then according to the **graph**, I can construct the node list and edge list for **maxflow graph**. I append **node** (node_flatcoord, capacity_towards_source, capacity_towards_sink) into the **nodeList**, and **edge** (curr_index, neighbor_index, capacity) into the **edgeList**. It follows the rule that if the point in the **graph** is 1 then I append (node_flatcoord, 0, MAXIMUM), 0 append (node_flatcoord, MAXIMUM, 0) and (node_flatcoord, 0, 0) to the rest. I calculate and append two edges into **edgeList** for each points in the **graph**, whose capacity using $\frac{1}{1+Euclidean(I(x,y),I(x+1,y))}$ and $\frac{1}{1+Euclidean(I(x,y),I(x,y+1))}$
    - Then I use built-in maxflow package to construct the graph **g** that first connect all nodes to source and sink, then for nodes which has edges to each other (have edge in **edgeList**) add edges between them. And do the maxflow() algorithm.
    - Then I obtain the mask that the foreground is valued 1 and background 0, so that I can generate the mask and overlays based on **g**.get_segment(index)
  - **The multi-segments division**
    - I reuse the cut graph algorithm. And to divide multiple parts, I create 4 lists to collect four kinds of seeds, select one to be the foreground and others combined to be the background, loop this procedure for all four lists as they all become once the foreground. Then I check the mask before filling in color to see if it is occupied by other colors before and only fill in the non-occupied parts.
    - I modified the GUI to fit for four kinds of seeds by pressing '1', '2', '3', '4' in the keyboard
- As shown in figure 1 you need to perform graph cut method to segment the foreground and background of the given image q1_1.jpeg. (15’)
  - First original image, second seed record image, third mask image, fourth mask-origin overlayed image



- Modify your graph cut program for multi-class segmentation. You need to segment the given image q1_2.jpeg to four parts, and show the segmentation results via transparent painting overlayed with the origin image as shown in Figure 2. (20’)
    - First original image, second seed record image, third masks image, fourth masks-origin overlayed image



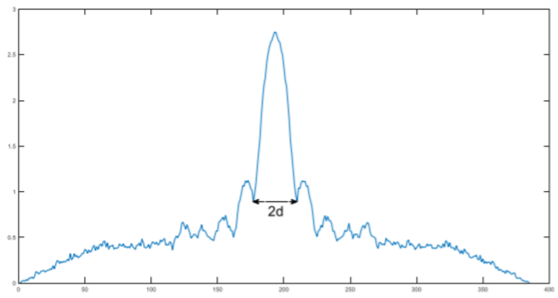
Question 2 Image Blending

- **Pyramid-Based Image Blending**
    - There are two versions of registrations, one is by reading two .mat feature files that contains man's and girl's feature and use code to match them, it performs well in blending chins. Another one uses a pre-defined matrix to do the transform, which do better in mouths and skins.
    - I think the pre-defined version is better

- **Poisson-Based Image Blending**

Question 3 Image Deblurring

- How do you implement your algorithm? Describe it by flow charts or words. (6 pts)
    - I first transform the image into **square shape** using resize function, then do FFT and some post process to get a picture with clear stripes.
    - Estimate θ with **Hough transform**, and estimate L with the projected plot of the rotated stripe figure (stripes perpendicular to horizontal line)
    - Reconstruct the blur matrix by `fspecial('motion',L,theta)`, then use **weiner** or **CLS** filter to deblur the image
- Describe implement details of your code. (6 pts)
    - For **DetectMotionParameters**:
        - As described above, I did **FFT** for the **squared image** and get the clear stripes image following these steps:
            - First do $fft\_image_{(i,j)} = 15 * log(abs(fft\_image_{(i,j)}))$
            - Then **binarize** the image, use **morph** to close up disconnected short lines and cut out connected lines that are not supposed to be connected.
            - Use **edge detection** to check for the stripes we need
            - Use **Hough** transform to find lines' θ = 180 - hough_theta
            - Rotate the image by θ to get the stripes perpendicular to horizontal, then project it to get the plot, and estimate $L = N/d$ with the highest hill.



    - For **Weiner Filter**:
        - In a word, the Weiner filter follows the following calculation in frequency domain

$$G(f) = \frac{1}{H(f)}\left[\frac{|H(f)|^2}{|H(f)|^2 + \frac{N(f)}{S(f)}}\right] = \frac{1}{H(f)}\left[\frac{|H(f)|^2}{|H(f)|^2 + \frac{1}{SNR(f)}}\right]$$

        - where $\frac{1}{SNR(f)}$ is $\frac{noise\ constant}{VAR(blurred\ image)}$, $H(f)$ is the estimated blur matrix in frequency domain
        - $\hat{X}(f) = G(f) * Y(f)$, where $Y(f)$ is the blurred image in frequency domain, and $\hat{X}(f)$ is the recovered image in frequency domain, recover $\hat{x}(f)$ by using **abs(IFFT)**
    - For **Constraint Least Square Filter**:

        - In a word, the Weiner filter follows the following calculation in frequency domain

$$\hat{F} = \frac{\lambda H^* G}{\lambda H^* H + P^* P} = \left[\frac{H^*}{||H||_2^2 + \gamma ||P||_2^2}\right] G \quad (5)$$

        - where $P = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$, H is the estimated blur matrix in frequency.
        - $\hat{F}$ is the recovered image in frequency domain, recover $\hat{f}$ by using **IFFT**

- The parameters L and theta that you get from each figure.(3*2pts)
    - The first $L = 20.75, \theta = 258$
    - The second $L = 21.5094, \theta = 214$
    - The third $L = 40, \theta = 124$
- Your selected filtering method and the recovered images, as well as the Structural Similarity (SSIM) of the restored image and the original image . (3*4pts)
    - For picture 1, I select Constrained Least Square method since it over-performs the Weiner filter (first CLS, second Weiner, original on the left, recovered on the right)
    - For picture 2, I select Constrained Least Square method since it over-performs the Weiner filter (first CLS, second Weiner,original on the left, recovered on the right)
    - For picture 3, I select Constrained Least Square method since it over-performs the Weiner filter (first CLS, second Weiner,original on the left, recovered on the right)