CS270 Homework1

Deadline 2020/4/6

Notes:

10 points (2 points for algorithm description, 4 points for code implementation, 4 points for performance of image processing) for each question, 3 questions, a total of 30 points. The preliminary results are just for reference. Please try to achieve the best performance as you can. **Discussions are encouraged and plagiarism is strictly prohibited, source code should not be shared in any form.** Please submit your homework to spring2020cs270@163.com with subject(CS270+ID+name+hw1) example, CS270_2019123321_张三_hw1.

Question1.

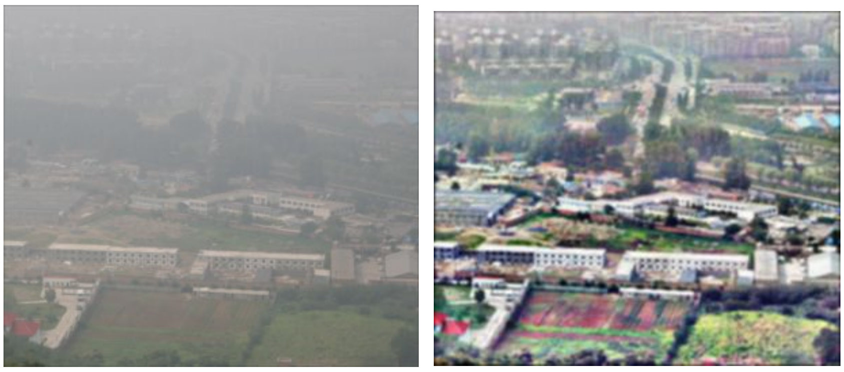
In this question, we first request that you can improve the quality of one given image using image enhancement methods achieved by your own programs. For example, you could improve the image contrast by adjusting the histogram, or try some image filters. As shown in Fig. 1, (a) shows the original image which looks like a little foggy; (b) gives one preliminary result which seems clearer after improving the contrast.

Then, we need you to impair the image quality on one normal image (see Fig. 2), which is contrary to the above. You would make the clear image foggy after changing the contrast. It is helpful for you to better understand the process of image enhancement.

Tasks:

(1) Please describe your algorithms in words or flowcharts.

(2) Achieve the process by your own codes and try not to use library functions of MATLAB.



(a) Original image (b) Improved image

Figure 1: Examples of image enhancement.

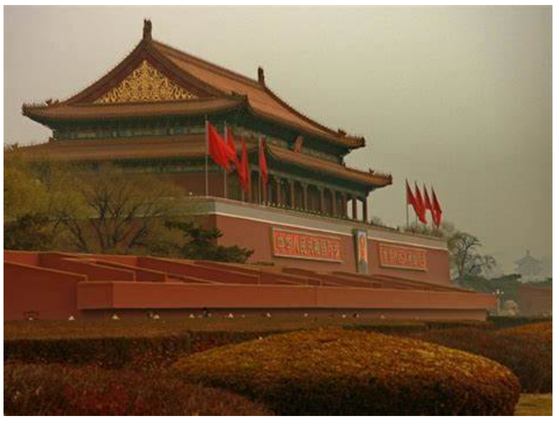


Figure 2: Input image to be blurred.

Question2.

Image stitching is the process of combining multiple photographic images with overlapping fields of view to produce panoramic or high quality views. Detecting features, feature matching, image registration and blending make up the whole stitching process.

Here we got two village photos taken by drone. figure 1 has a wider perspective while the houses in figure 2 is clearer. We would like to combine the two figures to get a photo with wide perspective and clear houses. Fortunately, we only need to focus on image registration in this homework. You can select and match feature points manually with the help of Matlab build-in function. **However, you have to implement image registration by yourself. Low scores would be given if using *fitgeotrans()* or similar functions to Fit geometric transformation.**

Tips:

Since image registration would be covered in lecture8, don’t miss it if you are unfamiliar with image registration. Reference results are provided in figure3,4,5. You may use *cpselect()* to achieve select and match feature points.

Tasks:

1. Please describe your algorithm in words or flowcharts and show us your equations and equations’ solution which used to perform geometric transformations.
2. Implement image registration. The code needs to be provided for review.
3. Show your results. Your results should not be worse than the reference results.

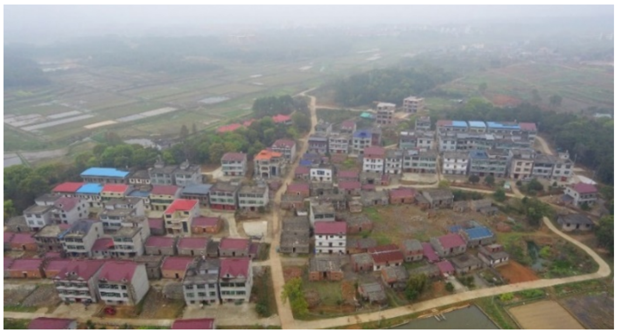


Figure 2

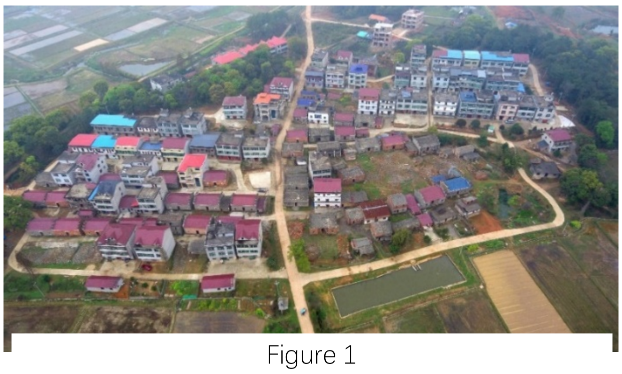


Figure 1

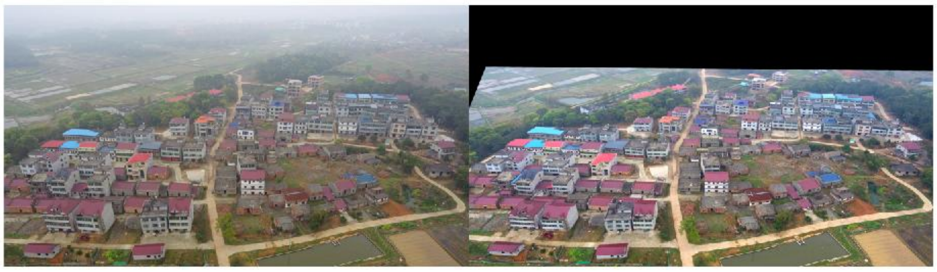


Figure 3--figure1 keep the same, figure2 Registration. montage({frame2,registered})

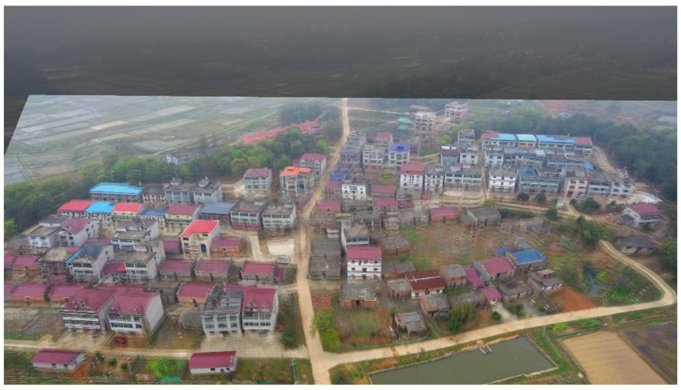


Figure 4--blend figure1 and Registered Images. imshowpair(frame2,registered,'blend')

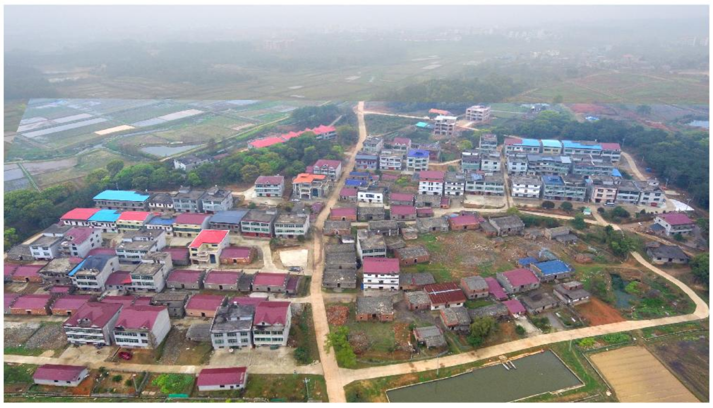


Figure 5--replace the houses in figure1 with registered figure

Question3.

1. Gray-level co-occurrence matrix

The gray-level co-occurrence matrix (GLCM) was introduced by Haralick in 1973 and often used for texture analysis. The GLCM can be specified in a matrix of relative probability $P(i,j)$ with which two neighboring pixels occur on the image, one with gray level i and the other with gray level j.

Fig. 1 (a) gives one image with a gray-level of 3 ($n=3$), which means that the gray values of the image can only be $1,2,3$ (see Fig. 1 (b)). We first construct a pair of points for one pixel (x,y) and its neighboring one $(x+a,y+b)$. If the gray value of (x,y) is i and that of $(x+a,y+b)$ is j, we get a combination of gray values (i,j). For $n=3$, the total number of combinations of gray values is 3×3 which is the dimension of the GLCM (see Fig.1 (c)). To simplify the problem, we set $a=1,b=0$ and get the point pair (x,y) and $(x+1,y)$. We count the number of occurrences of each combination $(1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3)$, and the calculated result is the GLCM. For example, we have the combination $(1,2)$ for (x,y) with a gray value of 1 and $(x+1,y)$ of 2. It appears three times in the image (see Fig. 1 (b)), so the value of the GLCM in the 1st row and the 2nd column is 3. Finally, to calculate the probability $P(i,j)$, we need to use the GLCM to divide the sum of all the elements in the GLCM.

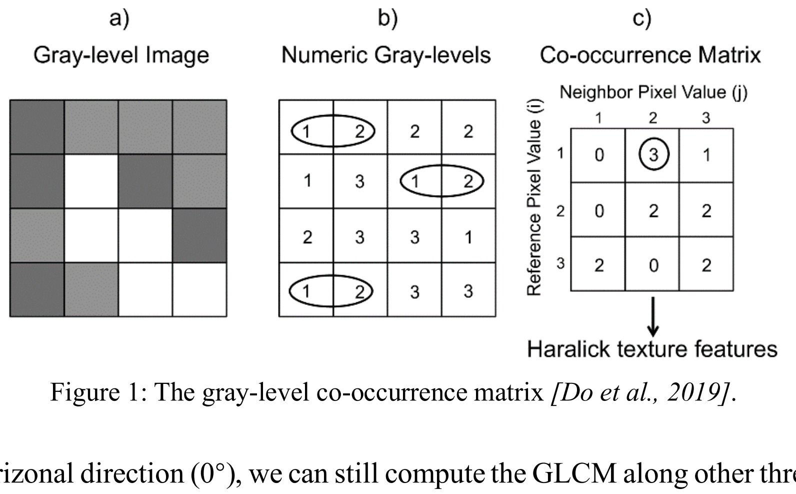


Figure 1: The gray-level co-occurrence matrix *[Do et al., 2019]*.

Except the horizonal direction (0°), we can still compute the GLCM along other three directions: 45°, 90° and 135° (see Fig. 2).

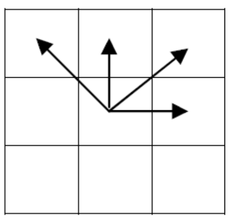


Figure 2: The four directions of adjacency for calculating the gray-level co-occurrence matrix *[Sebastian et al., 2012].*

After creating the GLCMs, we can derive several statistics from them using the different formulas as shown below. These statistics provide information about the texture of an image. Contrast, a measure of the intensity contrast between a pixel and its neighbor,

$$Contrast = \sum_i \sum_j (i-j)^2 P(i,j). \tag{1}$$

Correlation, a measure of how correlated a pixel is to its neighbor,

$$\begin{aligned} Correlation &= \sum_i \sum_j \frac{P(i,j)(i-\mu_i)(j-\mu_j)}{\sigma_i \sigma_j} \\ \mu_i &= \sum_i \sum_j i \cdot P(i,j) \\ \mu_j &= \sum_i \sum_j j \cdot P(i,j) \\ \sigma_i &= \sum_i \sum_j P(i,j) \cdot (i-\mu_i)^2 \\ \sigma_j &= \sum_i \sum_j P(i,j) \cdot (j-\mu_j)^2 \end{aligned} . \tag{2}$$

Energy, the sum of squared elements in the GLCM,

$$Energy = \sum_i \sum_j (P(i,j))^2. \tag{3}$$

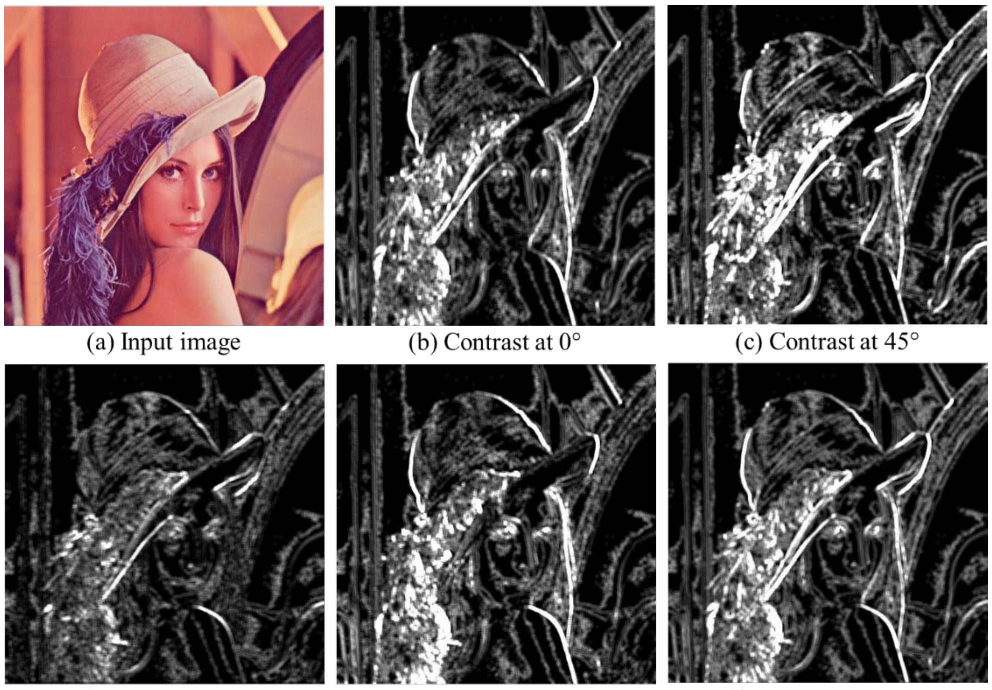
Homogeneity, a value that measures the closeness of the distribution of elements in the GLCM to the GLCM diagonal,

$$Homogeneity = \sum_i \sum_j \frac{P(i,j)}{1+(i-j)^2}. \tag{4}$$

Using the calculated results of the above formulas, we can generate different feature images from the GLCM. We set a fixed-size window to traverse the entire image and compute Contrast, Correlation, Energy and Homogeneity in the window. We use these features to replace the gray values of the center pixels, and then we can get different feature images corresponding to different GLCM features.

2. Homework

We request that you can calculate the GLCMs of the enclosed image "Lenna.png" by MATLAB and illustrate the four feature images corresponding to Contrast, Correlation, Energy and Homogeneity, respectively. Specifically, you should first use a sliding window with the size of 7×7 to traverse the image. Within each window, you will calculate the GLCMs along 4 directions (0°, 45°, 90° and 135°) with an Chebyshev distance 1 to construct the point pair. For example, $((x,y),(x+1,y))$ for 0°, $((x,y),(x+1,y+1))$ for 45°, $((x,y),(x,y+1))$ for 90° and $((x,y),(x-1,y+1))$ for 135°. The number of the gray level is 256 for the input image. To save computing, you can convert it to 8 gray level. Then, you need to generate 4 features by formulas and compute the average of features at different directions as the final value to be presented in the feature image. Finally, 4 feature images with the same size of the input image should be given. Fig. 3 show the feature images at 4 directions and the average result.



(a) Input image (b) Contrast at 0° (c) Contrast at 45°

(d) Contrast at 90° (e) Contrast at 135° (f) Average Contrast

Figure 3: The Contrast image at 4 directions and the average result.

Tips:

(1) There are two functions "graycomatrix" and "graycoprops" in MATLAb which can calculate the GLCM and 4 features simply. However, we highly recommend that you can achieve the process by your own codes. It is not only helpful for you to understand the neighborhood operation of image and feature extraction, but also important to give you a higher evaluation in this homework. Besides, if you decide to use "graycomatrix", you still need to consider how to generate the feature image by the sliding window.

(2) The input image is color image, so you need to convert it to gray image before you start to calculate the GLCM (G denotes gray).

(3) For the gray level, the input image is 256. We recommend that you convert it to 8. For example, you can just use a simple mapping from 0~255 to 1~8. Why choosing 1~8? It is because the function "graycomatrix" can set the range between 1 and 8. It will keep consistent in case some students use the MATLAB function. It is not mandatory to do this, so you can select 0~7 or keep the original 0~255 (the dimension of the GLCM will be 256×256).

(4) If you don't know how to deal with border pixels when selecting window, try "padarray".

(5) If you decide to write your own codes to calculate the features, please remember to normalize the GLCM before the calculation. The range of $P(i,j)$ should be between 0 and 1.