Computer Vision: Representation and Recognition Assignment 2

171860607, Jinbin Bai, jinbin5bai@gmail.com

May 20, 2021

1 1 Edge Detection

1.1 (a)

The codes and comments are implemented in **edgeGrandient.m** and **gradientMagnitude.m**.

In **edgeGrandient.m**, our algorithm is designed in the following order.

- smooth the image with Gaussian std=sigma
- compute the x and y gradient values of the smoothed image
- compute overall gradient magnitude which is the L2-norm of the x and y gradients
- compute orientation of gradient
- compute orientation of gradient with largest magnitude over third dimension
- compute overall gradient magnitude which is the L2-norm of the R, G, and B gradients

In **gradientMagnitude.m**, our algorithm is designed in the following order.

- use gradientMagnitude to compute a soft boundary map
- rescale the boundary scores for better visualization
- two ways to perform suppression

At first, when we use no-max suppression with sigma=3, the overall and average F-score are 0.595 and 0.629. And latter we use canny suppression with sigma=3, the overall and average F-score are 0.616 and 0.652.More details are listed in table1 and table2.

	sigma=2	sigma=3	sigma=4
overall F-score	0.592	0.595	0.587
average F-score	0.632	0.629	0.619

Table 1: no-max suppression

	sigma=2	sigma=3	sigma=4
overall F-score	0.573	0.616	0.615
average F-score	0.624	0.652	0.648

Table 2: canny suppression

As a result, we find sigma=3 with canny suppression leads to the best F-score.
Here are some qualitative results and quantitative results in Figure 1 2 3.

Figure 1: figure 21077

Figure 2: fugure 24077

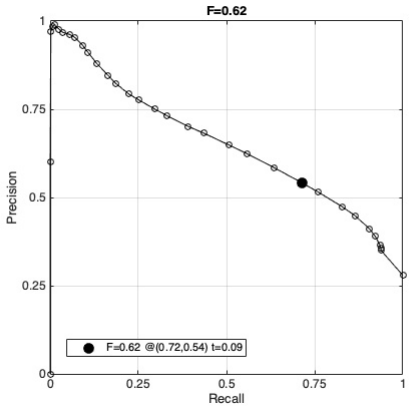

Figure 3: pr full

1.2 (b)

The codes and comments are implemented in **edgeOrientedFilters.m** and **orientedFilterMagnitude.m**.

In **orientedFilterMagnitude.m**, our algorithm is designed in the following order using four oriented filters[1].

- find two derived gaussians with respect to x and y
- create another three pair filters, m means minus
- run four differnet elongated Gaussian derivative filters on image
- way 1
  - directly compute mag, as a result there is no way to compute theta
- way 2
  - get each four mag's direction then use max function to vote, in this way we can compute theta

In **edgeOrientedFilters.m**, our algorithm is designed the same as (a).

At first, when we use way 2 with gaussian size=5, the overall and average F-score are 0.578 and 0.627. And latter we use way 1 with gaussian size=20, the overall and average F-score are 0.627 and 0.661.More details are listed in table3 and table4.size=1000 shows slower speed without improvement.

	size=5	size=10	size=20
overall F-score	0.578	0.610	0.617
average F-score	0.627	0.646	0.653

Table 3: way 2

	size=5	size=10	size=20
overall F-score	0.588	0.620	0.627
average F-score	0.636	0.654	0.661

Table 4: canny suppression

As a result, we find way 1 with gaussian size=20 leads to the best F-score.

Here are some qualitative results and quantitative results in Figure 4 5 6.

Using the code **imagesc(mat2gray(F45));** to show the filter in Figures 7. For example, for picture 42049 and 119082, the following are the results obtained by using eight filters respectively in Figure 8 9.

Figure 4: figure 119082

Figure 5: fugure 42092

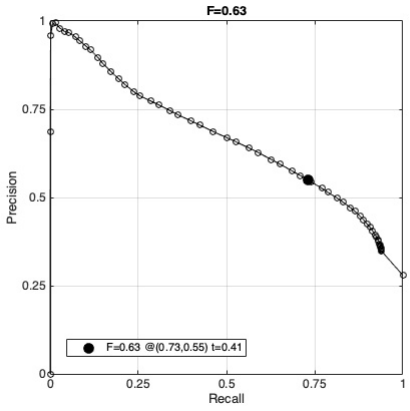

Figure 6: pr full

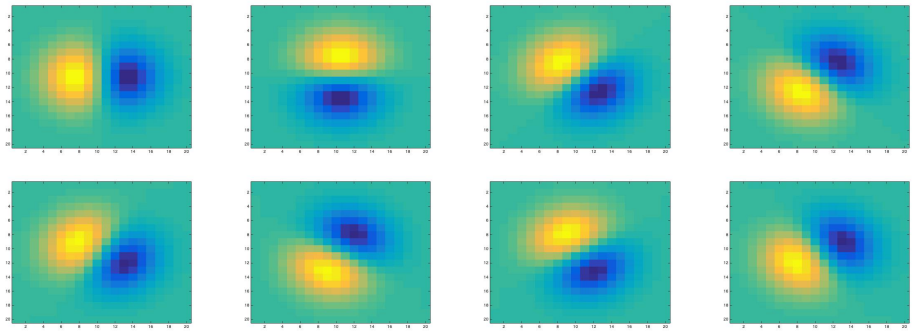

Figure 7: 4 pair filters

Figure 8: 4 pair filters

Figure 9: 4 pair filters

1.3 (c)

In darker places, the RGB value of the whole picture is very low, and in brighter places, the value is higher. The values of the three RGB channels are easy to change with the change of lighting, and it is impossible to get the specific area you need in a complex environment.

So where the light and dark changes are relatively large, we have some color spaces that are more suitable for image processing-HSV and YUV.[2]

Equation1.1 shows conversion formula from RGB to HSV (if H<0 then H=H+360). And equation1.2 shows conversion formula from RGB to YUV.

$$\begin{aligned} H_1 &= \cos^{-1}\left\{\frac{0.5[R-G+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right\} \\ H &= \begin{cases} H_1 & \text{if } B \leqslant G \\ 360^\circ - H_1 & \text{if } B > G \end{cases} \\ S &= \frac{\max(R,G,B)-\min(R,G,B)}{\max(R,G,B)} \\ V &= \frac{\max(R,G,B)}{255} \end{aligned} \tag{1.1}$$

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.332 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{1.2}$$

In matlab, what we need to do is only add **im=rgb2hsv(im);** to achieve conversion. As a result, new F-scores are listed in Table 5 below and better than SOTA in (a) and (b).

	(a)	(b)
overall F-score	0.642	0.634
average F-score	0.665	0.659

Table 5: from RGB to HSV

2 Image segmentation with k-means

2.1 (a)

In **quantizeFeats.m**, our algorithm is designed in the following order.

- get some dimensions
- loop through each column and find the cluster for that column of pixels using dist2.m
  - create a matrix of vectors for each pixel in a column
  - save the cluster where the pixel is closest to in labelIm

2.2 (b)

In **createTextons.m**, our algorithm is designed in the following order.

- get number of images and filters
- loop through each image
  - loop through each filter and record the filter responses
- select samples randomly, one sample pixel per 2000 pixels

2.3 (c)

In **extractTextonHists.m**, our algorithm is designed in the following order.

- get some dimensions
- loop through each filter and apply it to the image
- get labelIm with quantizeFeats function
- calculate frequency of each texton in window

2.4 (d)

In **compareSegmentations.m**, our algorithm is designed in the following order.

- get some dimensions
- caculate colorLabelIm
- caculate textureLabelIm

2.5 (e)

In **egmentMain.m**, our algorithm is designed in the following order.

- read the images and load the filter bank
- create image stack with gray images
- generate texton codebook
- try some parameters to finish 3 tasks

The result will be showed in part (f).

2.6 (f)

As we show in figure 10,11,12 and 13 (the first three images are provided and the last one is we chose) , we list original images, color labeled images and texture labeled images with their parameters.

As we show in figure 14 and 15, we compare different parameters with twins image. As a result, we find that when we use small winsize, texture labels show detailed structure and texture, such as lines on clothes and face details. And when we use big winsize, the background will be smoother. In conclusion, if you need more texture details, smaller winsize is more suitable for you. If you just want to know the outline of the image and want to make the background smoother and ignore details, bigger winsize is more suitable for you.

On the other hand, different directions of filters convey different directions of texture information. For instance, when we only use vertical filters, the texture information that we get only contains vertical texture information.

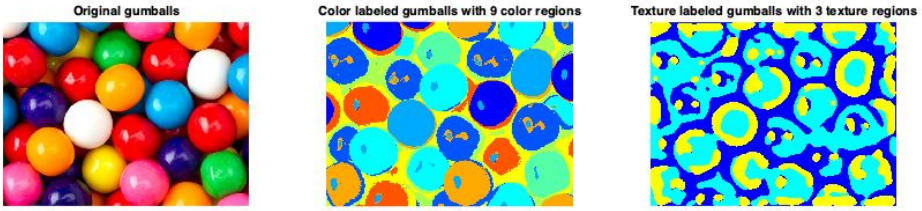

Figure 10: gumballs with winsize=12

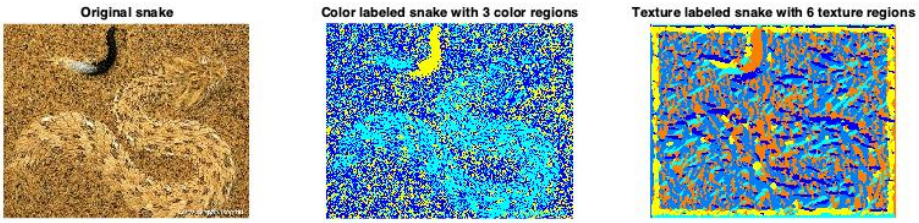

Figure 11: snake with winsize=5

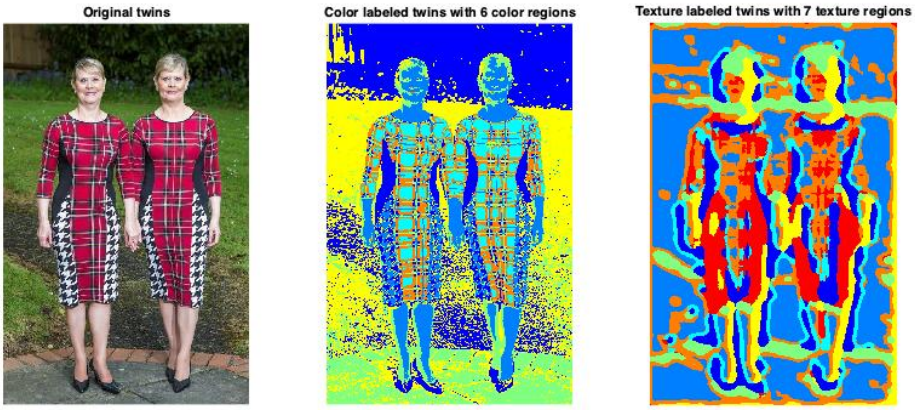

Figure 12: twins with winsize=15

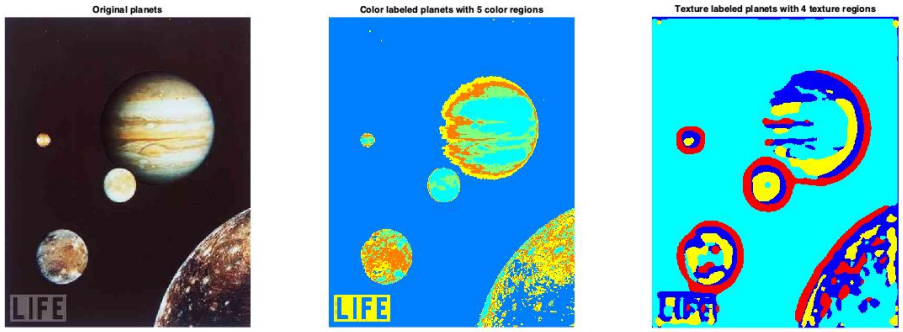

Figure 13: planets with winsize=9

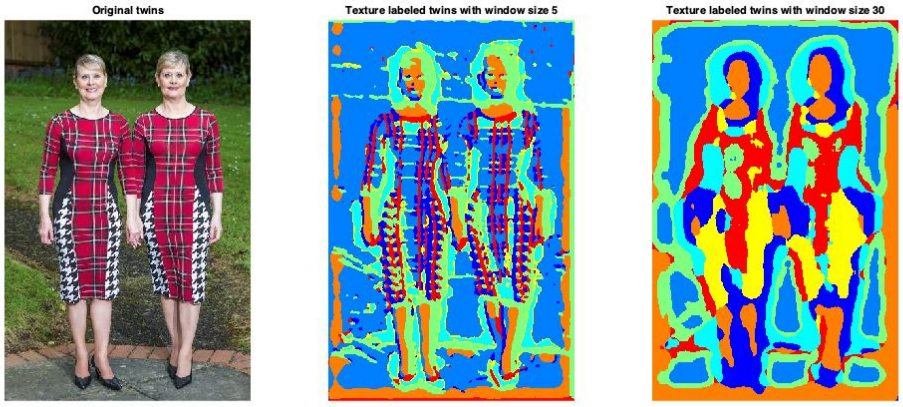

Figure 14: twins with different winsize

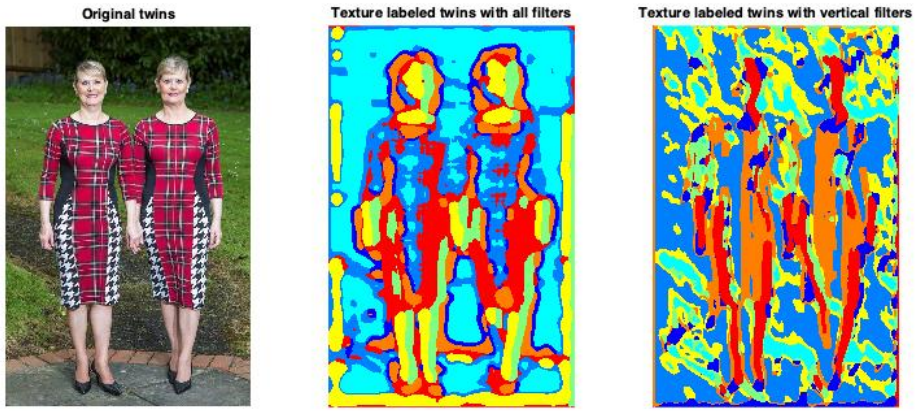

Figure 15: twins with different filter bank

reference

[1] W.T. Freeman and E.H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[2] D.R. Martin, C.C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549, 2004.