Computer Vision: Representation and Recognition
Assignment 2

211240076, Liu Jiaxin, 211240076@smail.nju.edu.cn

May 19, 2024

1 Question 1

1.1 question1.1

To implement the function

$$function[mag, theta] = gradientMagnitude(im, sigma)$$

we have the following steps:

1. smooth the image with Gaussian std=sigma by passing it to 'imgaussfilt'
2. compute the x and y gradient values of the smoothed image using 'gradient'
3. compute the gradient magnitudes of R, G, B values respectively; then use it to compute the gradient magnitude of the RGB image by taking the L2-norm of the R, G, and B gradients
4. compute the theta values for R, G, B values respectively, then use the theta value which has the maximal gradient magnitude as the orientation at this pixel

Figure 1: figure 3096 and its result

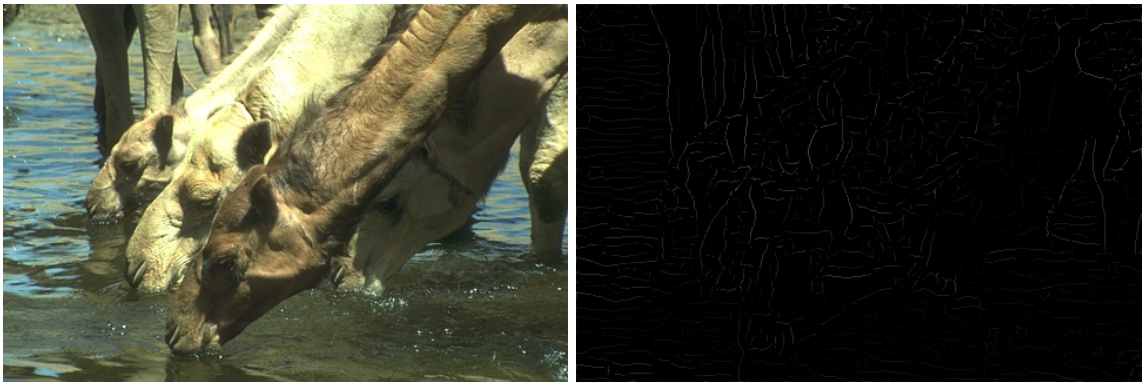

Figure 2: figure 16077 and its result

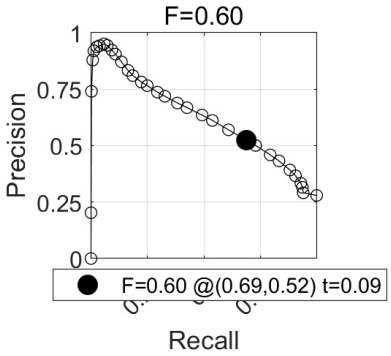

Figure 3: precision-recall plots

Method gradient: overall F-score = 0.595 average F-score = 0.629

1.2 question1.2

The bank of filters I used for part(b) is a set of steerable filters:

$$\{(cos(a) * dGx, sin(a) * dGy)\}$$

for $a = \pi * (i-1)/6, i = 1, \cdots, 6$, where (dGx, dGy) is the gradient of a Gaussian filter.

Figure 4: figure 3096 and its result

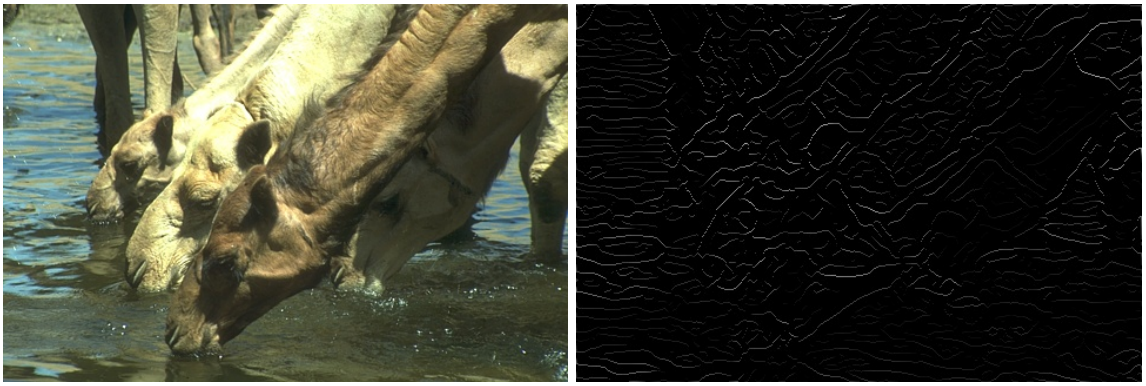

Figure 5: figure 16077 and its result

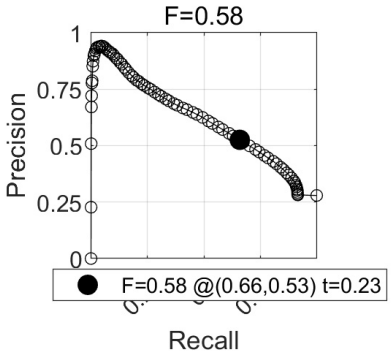

Figure 6: precision-recall plots

Method oriented: overall F-score = 0.585, average F-score = 0.624

1.3 question1.3

Idea: Use bilateral filters to smooth the image
Reason: the Gaussian filters will blur the edges, which may results in bad edge detection; bilateral filters possess the property of "Edge Preseving"

2 Question 2

2.1 question2.1

To implement the function

$$function[labelIm] = quantizeFeats(featIm, meanFeats)$$

we fetch each row pixels of $featIm$, calculate the distances to the $meanFeats$ using 'dist2' function. Then use the index whose distance is the minimal as the label (cluster membership).

2.2 question2.2

The function

$$function[textons] = createTextons(imStack, bank, k)$$

computes a texton "codebook" (i.e., set of quantized filter bank responses) based on a sample of filter responses.

It iterates through the image stack and filters each image with all filters in the filter bank. Then it samples a subset of the pixels' filter responses and puts it into the dataset "Textons".
Now it only need to do k-means clustering on the dataset using 'kmeans' and returns the clusters.

2.3 question2.3

In function

$$function[featIm] = extractTextonHists(origIm, bank, textons, winSize)$$

constructs a texton histogram for each pixel based on the frequency of each texton within its neighborhood (as defined by a local window of fixed scale winSize).
Specifically, the origin image is filtered by all the filters in the filter bank. Then we get the cluster membership of each pixel to the textons using 'quantizeFeats' that has been implemented earlier:

$$labelIm = quantizeFeats(filteredIm, textons);$$

After that, we only need to get the frequency for the window of each pixel. Here I fetch the window matrix and using 'tabulate' function to obatin the infomation.

2.4 question2.4

The function

$$function[colorLabelIm, textureLabelIm]$$

$$= compareSegmentations(origIm, bank, textons, winSize, numColorRegions, numTextureRegions)$$

comes in two parts, the first part of which simply does the k-means on the feature space of colors for each pixel.
For the second part, we're supposed to extract the texton histogram of the origin image and then do the k-means based on it. Notice that the function 'extractT extonHists' takes in a grayscale image, so we should transform the original RGB image into a grayscale image first:

$$textonHist = extractTextonHists(rgb2gray(origIm), bank, textons, winSize);$$

2.5 Results and Explanations

Note: the number of textons is set to be $k\_textons = 17$

2.5.1 results

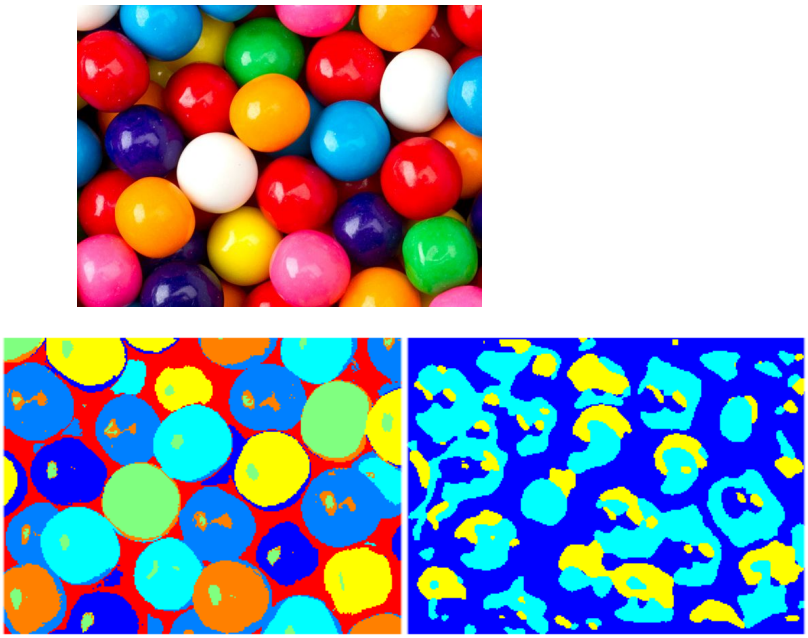

Figure 7: gumballs.jpg and results, (winSize, numColorRegions, numTextureRegions)=(12, 7, 3)

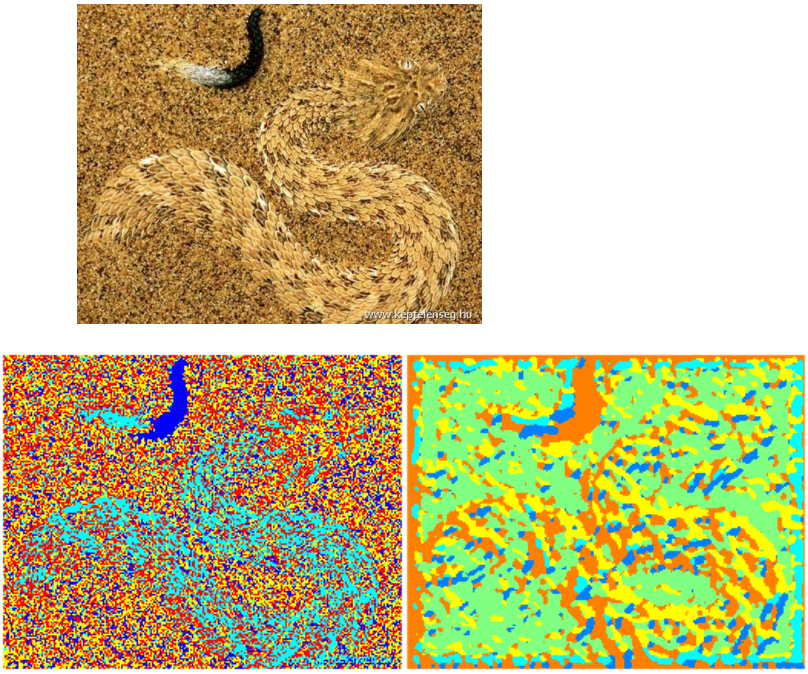

Figure 8: snake.jpg and results, (winSize, numColorRegions, numTextureRegions)=(6, 4, 5)

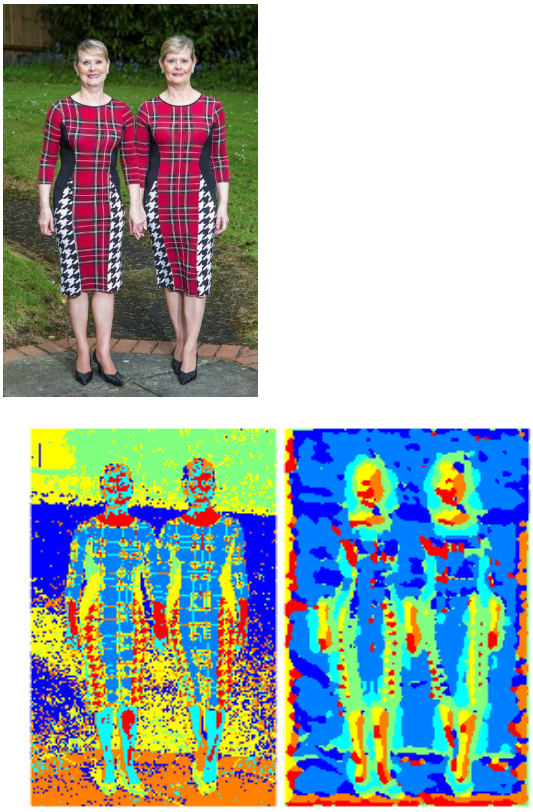

Figure 9: twins.jpg and results, (winSize, numColorRegions, numTextureRegions)=(9, 7, 7)

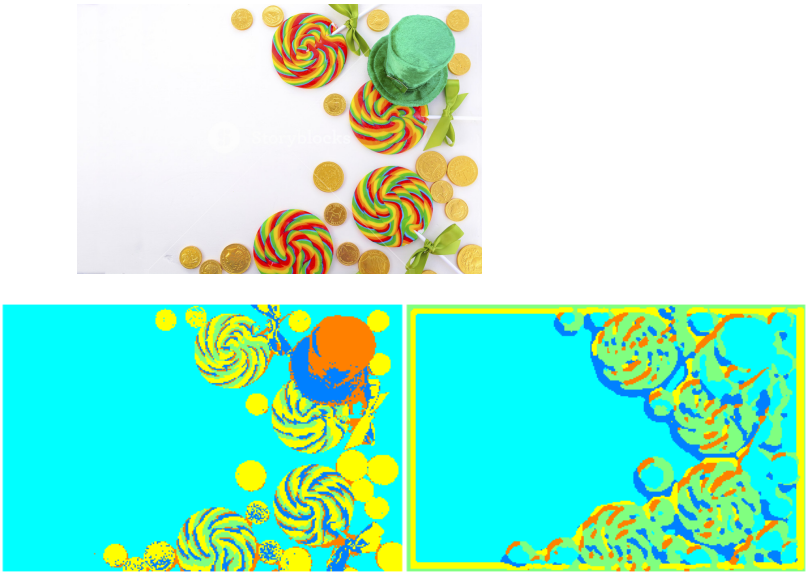

Figure 10: coins.jpg and results, (winSize, numColorRegions, numTextureRegions)=(10, 5, 5)

2.5.2 Explanations

Choose two window sizes for texture representation:

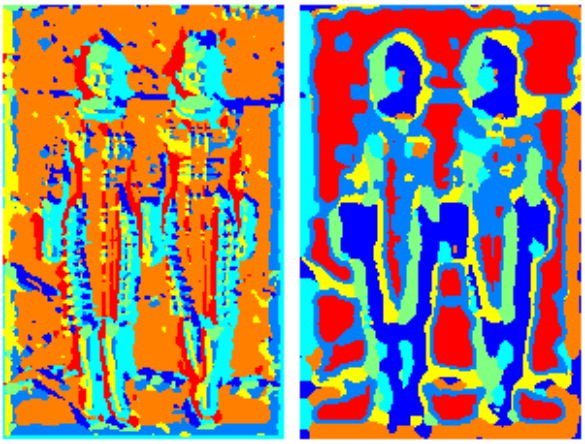

Figure 11: two texture segmentation results of twins.jpg, winSize is 5 and 22

we can easily find that small window sizes lead to detailed texture while large window sizes lead to less details. (Notice: Larger winSizes were chosen such as 35 and 50, but they didn't lead to convergence in kmeans.)
Run the texture segmentation results with another filter bank which is the subset of the given filter bank. The filter bank contains all the horizontal filters:

$$subbank = bank(:, :, 1 : 6 : end);$$

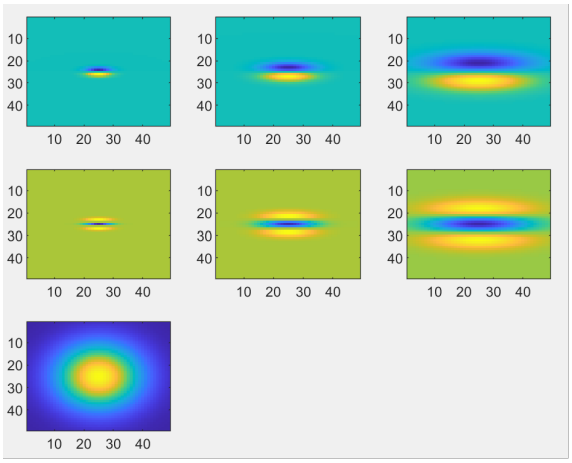

Figure 12: horizontal filter bank

We get:

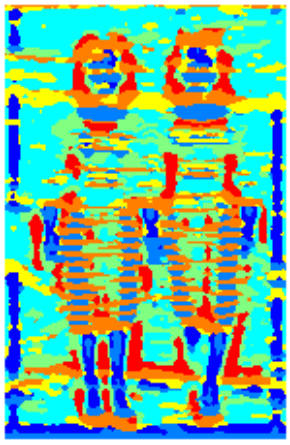

Figure 13: texture segmentation results of twins.jpg, with horizontal filter bank

Compared to the previous texture segmentation result, we can find that only horizontal textures are extracted. So we conclude that filters with different orientation are needed for the texture segmentation.