Homework 1
M1522.001000 Computer Vision (2016 Spring)
Due: Wednesday March 23, 11:59PM

The goal of this homework is to explore Texture Classification, Feature Extraction and Transformation.

1 Submitting Your Assignment

Your submission for this assignment consists of answers to two theory questions, the code for your MATLAB implementation, a short writeup questions through 1 to 3 and interesting observations you made or things you did differently while implementing the assignment. You need to zip the following items, name it as `M1522001000_HW1_(your student ID).zip`, and send it to TA's email(`ta.cv@vision.snu.ac.kr`).

Your submitted zip file should include the files arranged in this layout:

- `theory.txt` or `theory.pdf`
- Folder `result`
  - `TextureLibrary.mat`, `Histogram.mat`
  - `reference_sift.png`, `test_sift.png`, `reference_match.png`, `test_match.png`, `affine_result.png`
  - `writeup.doc` or (or `.pdf`): the writeup describing your experiments
- Folder `matlab`
  - `extractResponseVectors.m`, `createTextonLibrary.m`, `createHistogram.m` , `classify.m`
  - `sift_match.m`
  - You can also include any extra helper functions.

Refer to Section 3 for the details of the above files.

Your zip file should be sent before the due(Wednesday March 23, 11:59PM). Later than that, you will use one late day.

2 Theory Questions

Question 1 (10 points)

(a) What kind of function f convolves with any other function g to give g back?

(b) Suppose that the derivatives of an image I are denoted by I_x and I_y. Design an algorithm that, at each location (x, y), smoothes the image by σ_1 in the direction of the dominant edge at (x, y) and by σ_2 in the perpendicular direction. Simply write down the formulas involved at each location (x, y) to compute the

output $I'(x, y)$ from I, I_x and I_y, assuming that you already have computed I_x and I_y.

Question 2 (10 points)

(a) The Hough transform is a line detector. Can you come up with a transform for circles? What is the representation of a family of circles in this space?

3 Programming & Writeup Questions

Throughout this part of assignments we will not only ask you to program, but also to answer questions/make observations that relate to the code you will write. These questions will help you to you understand the concepts that you implement in code. These questions will be numbered Q1 through Q4 and will be in **bold**. Please hand in both **code** and **answers**. For your and our convenience, all your code should be implemented in given `YOURCODE` directory, and all code should use relative file path.

3.1 Texture Classification (55points)

In this section of the assignment, we will implement some interesting parts of the following paper:

```
Cula, O. G. and K. J. Dana: 2004,
3D Texture Recognition Using Bidirectional Feature Histograms.
International Journal of Computer Vision 59(1).
```

Although the paper looks long and scary, do not fear. We will indicate explicitly those sections of the paper that you actually need to read. Please remember to read those sections before you attempt the relevant part of the assignment.
Texture classification is an interesting problem in vision since many objects we would like to recognize have distinctive textures (such as zebras, pedestrian crossings and chess boards). The first section of the paper gives a broad outline of the most recent work in texture classification. Reading this part is extremely optional, but is recommended, especially if your research is related to texture. This paper uses the CURET database, which you may want to look at if you have some free time. It is at `http://www1.cs.columbia.edu/CAVE/curet/`. In this assignment, some of the images will also come from the BRODATZ database, which is also online at `http://www.ux.his.no/tranden/brodatz.html`
On our website you have access to a zip file that contains 7 directories, that you will use in this assigment:

- a directory of *random* textures
- 5 classes of *training* textures (grass, straw, seeds, chips and canvas) and
- a directgory of *testImages* textures for the 5 classes

The word 'texton' is both powerful and vague. Its origins are in early vision science studies which showed that certain biological systems were sensitive to fundamental building blocks of textures. The kinds of textons that you will extract from textures are the responses of the image to a scale invariant filter.
You will now take a bunch of training images and create a large texton database. You should follow the paper for guidelines. We will provide a short description of Gabor Filters which you will need in order to build the texton database.

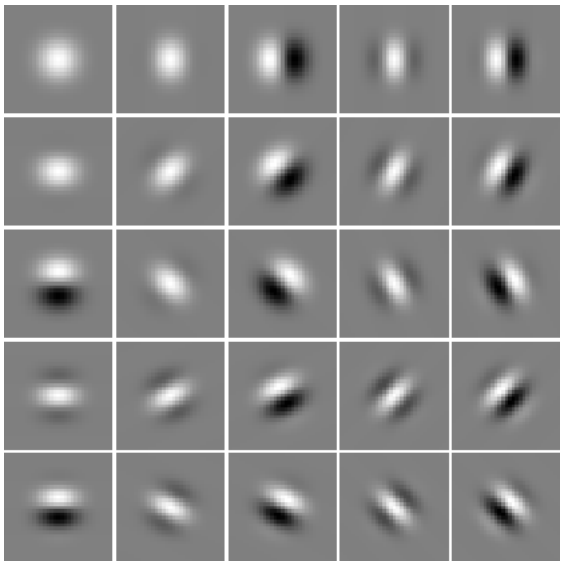

Figure 1: An example Garbor filter bank.

Part 1 Texton Library

Read Section 2 Intro

Q1) Why would you need a texton database? (2 sentences)

Part 2 Gabor Filters

Gabor filters have become popular as it has been shown recently that simple cells in the primary visual cortex can be modeled by Gabor functions. We must emphasize that we are, in fact, giving you everything that you will need to build the Gabor filter in Matlab. We are not asking any questions relating to the actual equations below, and you will not need to do anything more other than use them in the rest of the assignment.
A 2D Gabor function $g(x, y)$ is a 2D Gaussian modulated with a complex exponential (sinusoidal) and is given as:

$$g(x,y) = (\frac{1}{2\pi\sigma_x\sigma_y})exp\{-\frac{1}{2}(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}) + i2\pi Wx\}$$

A Gabor filter bank is a series of Gabor filters at various scales and orientations (which means they have different values for the inputs in the function above).

Applying each of these to the image gives a response at each pixel, for each filter in the bank.
The constant W determines the frequency bandwidth of the filters, use $W = 0.52$. The above representation of a Gabor function simply combines in a single function the even and odd Gabor functions. Gabor filters can be seen as 2D wavelets. Let $g(x, y)$ be the mother wavelet, then the self-similar filter bank can be obtained by appropriate dilations and rotations of $g(x, y)$ through the generating function as:

$$g_{mn}(x, y) = a^{-m} g(x', y'), \quad a > 1, \quad m = 0, \cdots, S-1 \quad n = 0, \cdots, K-1$$
$$x' = a^{-m}(x \cos\theta + y \sin\theta) \quad \text{and} \quad y' = a^{-m}(-x \sin\theta + y \cos\theta)$$

where a is the scale parameter, angle $\theta = n\pi/K, S$ is the total number of scales and K is the total number of orientations. We can define a certain family of filters by choosing low and high frequencies, U_l and U_h to define the band within which we are interested in working. We must also choose a scale factor, a, and the variances along the x and y axes, σ_x and σ_y. The Gabor filters can be shown to form a nonorthogonal basis, which implies that there is redundant information in the filtered images. It is possible to design the Gabor filter bank $g_{mn}(x, y)$ such that the tiling of the frequency space reduces these redundancies. This design yields the following equations for the necessary parameters:

$$\begin{aligned}
a &= \left(\frac{U_h}{U_l}\right)^{\frac{1}{S-1}} \\
\alpha &= \frac{a+1}{a-1} \\
\sigma_x &= \left(\frac{1}{2\pi}\right)\alpha\left(\frac{\sqrt{2\ln 2}}{U_h}\right) \\
\sigma_y &= \left(\frac{1}{2\pi}\right)\left[\tan\left(\frac{\pi}{2K}\right)\left(\frac{1}{2\pi}\right)\sqrt{\frac{\alpha^2-1}{(\sigma_x)^2}}\right]^{-1}
\end{aligned}$$

Note: There is nothing inherently intuitive about the following parameter values - they are just provided here to help you implement the Gabor filter.
For design purposes, you can choose S to be any number between 4 and 10 (we chose 10) and K to be between 8 and 12 (we chose 10). The values of center frequencies U_h and U_l can be chosen as 0.4 and 0.1 respectively.
What you need to do now is to write the following function:

```
[vectors] = extractResponseVectors(Image)
```

The extractResponseVectors function should take any Image and apply the Gabor filter bank onto it. *vectors* is an `MxN` matrix, where `N` is the number of Gabor filters in the bank and `M` is the total numbers of pixels in an image.
When you see the Garbor function $g(x, y)$ you can notice that the function has defined on complex plane. Don't panic. You can split $g(x, y)$ by real part and complex part, and calculate filter for each part. After that, just calculate the magnitude and you could get the filtered result.

Part 3 The Texton Library

In this section you will use the *random* textures directory that we have given you.

Read Section 2.1 of the paper
Each pixel in an image will give a vector of responses to the Gabor filter bank. The length of this vector will be the number of filters you create in the bank.
The next thing you will write is a script:

`createTextonLibrary`

In this script, you will use the `extractResponseVector` command you made to extract a vector of responses to the Gabor filter, from every pixel in each image contained in the *random* image directory. The point of the texton library is to approximate all the variety of textures that occur in the real world with a finite set of textons. You have already written code to extract response vectors from an image. Now you will run this function on all the images in the random directory to create a giant matrix of response vectors. Each row of the matrix is a response vector that you extracted using the Gabor filter, from some pixel in some image in the random directory. You should now cluster this giant matrix of intensity vectors using the BUILT-IN Matlab command:

```
[clusterresult clustercenters] =
      kmeans(GiantFeatureMatrix,NumberofClusters,EmptyAction,drop)
```

The EmptyAction bit means that if you input too large a k, then redundant cluster centers are dropped. *clustercenters* are the centers of the groups that kmeans has found in your GiantFeatureMatrix. Our main hope/assumption is that these centers capture what it means to be a texture that belongs to that class. Specifying a good value of k is actually a very hard question in machine learning. We suggest using a k between 8-25. If you change the number of clusters, then you will affect the whole classification. Therefore a design decision for you is the value of $NumberofClusters$.

For both your and our convenience, store your database in a Matlab data file called

`TextureLibrary.mat`

When you load this file into Matlab, you should get a matrix of textons in the workspace called `TextonLibrary`. PLEASE KEEP the variable name as directed. This matrix should be an `MxN` matrix, with `M` being the number of clusters you use and `N` is the number of filters in the filter bank.

Part 4 Using Training Images *Read Section 2.2 and Skim Section 2.3 of the paper*

Once you have the texton library, you can start to train images. Training images are also located in the zip file that you will download. There are 5 classes and the directories are named after each class. Each directory is filled with textures of

a certain kind. For example the trainGrass directory has grass images.
When you first read a training image, you should extract the response vectors at each pixel from it, using `extractResponseVectors`. What you will do now is to use the information in the paper to create the following script:
`createHistogram`

In this script, you will open a class directory (say Grass), and compute all the response vectors for every image in that class. You will NOT need to store all of the response vectors you compute. Instead you will keep a record of which *textons* from the *texton library* were important in describing that class. This record is a *histogram*. Generally, a histogram is represented as a 2-dimensional graph. The y-axis is usually some measure of count. Note that you can use the Matlab function named `hist` if you wish, but it is not necessary.

Q2) What is the x-axis of the histogram in this assignment? (1 sentence)

For your own convenience, and for grading purpose, create a Matlab data file called

`Histograms.mat`

When you load this file into Matlab, you should get many 1-dimensional vectors (`Nx1`) loaded into the workspace. These vectors are the histograms. Each histogram should be named after a class. For example there will be a `Grass_histogram` which refers histogram of canavas and a `Straw_histogram` which refers histogram of straw.

Part 5 Classification *Read Section 3 of the paper*
After the above section you will be able to take a single test image and create its histogram by using the texton library. Create a function:

```
[class] = classify(im)
```

This function creates a histogram for the input image, and figures out which class this image belongs to, using the texton library. The output should be a string of the directory name of the class. For example, if the classify result is Grass, the return value should be 'Grass'.

Q3) Run the classify function you made on the test class directories provided and report the result of each images and overall accuracy. To assist you with this classification, we have provided a file called 'testClassifier.m'