# CSE 152 Introduction to Computer Vision
# Homework 1

**Instructions:**

- Total points: 100
- **Due: 11:59 pm, Monday, Nov 5, 2018**
- For the implementation part, we've provided you with a number of functions and scripts in the hopes of alleviating some tedious or error-prone sections of the implementation. **Please implement your part under the comment line "Write your code here", and stick to the headers, variable names, and file conventions provided.**
- For the theory problems, please justify your solutions by necessary derivations or explanations.
- Please pack your codes and write-up into a single file named **YourPID_hw1.zip** and submit it to descartes.ucsd.edu. **And also submit your write-up (pdf) to Gradescope.**
- A complete submission consists of the following files:
  - cnn.m
  - filters.m
  - frequency.m
  - local_feature.m
  - prepareData.m
  - MNISTModel.mat
  - A write-up (.pdf format)
  - \img (image folder)

  **DO NOT** include the `data` folder created by `prepareData.m` in your submission. **Make sure that we can run your code without any modification, otherwise you are at risk of losing points.**
- **Start early!** Please familiar yourself with MATLAB and allow enough time for debugging.

# 1 Convolutions and Correlations

The file you will work on in this section is `filters.m`.

## 1.1 Implementation [10 points]

In this section, you will implement your own convolution and correlation in MATLAB. You should implement the following two functions in filters.m:

```
function y = my_conv2d(I,f)
function y = my_corr2d(I,f)
```

Both of them take an image $I$ and a filter $f$ as inputs and output an image $y$ that has the same shape as the input image. Your implementation will be compared with MATLAB function `conv2` and `filter2`. Show an input image and two corresponding output images in your write-up. You are provided with example images in folder `img` to test your implementation, but you are free to include any image you like in your write-up.

**Note**: The convolution and correlation formulas showed in the slides are not exactly the same as the built-in functions in MATLAB. You may need to add an offset to the original image like this:

$$h[m,n] = \sum_{k,l} f[k,l]I[m-k+\Delta, n-l+\Delta]$$

## 1.2 Commutative Property [5 points]

Recall that the convolution of an image $f : \boldsymbol{R}^2 \to \boldsymbol{R}$ and a kernel $h : \boldsymbol{R}^2 \to \boldsymbol{R}$ is defined as follows:

$$(f * h)[m,n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[i,j] \cdot h[m-i, n-j]$$

Or equivalently,

$$(f * h)[m,n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} h[i,j] \cdot f[m-i, n-j] = (h * f)[m,n]$$

Show that this is true (i.e. prove that the convolution operator is commutative: $f * h = h * f$). Please answer it in your write-up.

## 1.3 Linear and Shift Invariance [5 points]

Let $f$ be a function $\boldsymbol{R}^2 \to \boldsymbol{R}$. Consider a system $f \xrightarrow{\text{S}} g$, where $g = (f * h)$ with some kernel $h : \boldsymbol{R}^2 \to \boldsymbol{R}$. Show that $S$ defined by any kernel $h$ is a Linear Shift Invariant (LSI) system. In other words, for any $h$, show that $S$ satisfies both of the following:

- $S[a \cdot f_1 + b \cdot f_2] = a \cdot S[f_1] + b \cdot S[f_2]$

- If $f[m,n] \xrightarrow{\text{S}} g[m,n]$, then $f[m-m_0, n-n_0] \xrightarrow{\text{S}} g[m-m_0, n-n_0]$

Please answer it in your write-up.

# 2 Fourier Transform

The file you will work on in this section is `frequency.m`.

## 2.1 1D Fourier Series [3 points]

In this question, you are expected to compute the 1D Fourier series to convert an input function from spatial domain to frequency domain by filling out the corresponding part of `frequency.m`. The Fourier transform needs to be rearranged by shifting the zero-frequency component to the center of the array. You are allowed to call built-in MATLAB functions. Figure 1 shows the example results for function $cos(\frac{2\pi x}{10})$. Please save your resulting figure for function $sin(\frac{2\pi x}{5}) + cos(\frac{2\pi x}{10})$ and include it in your write-up.

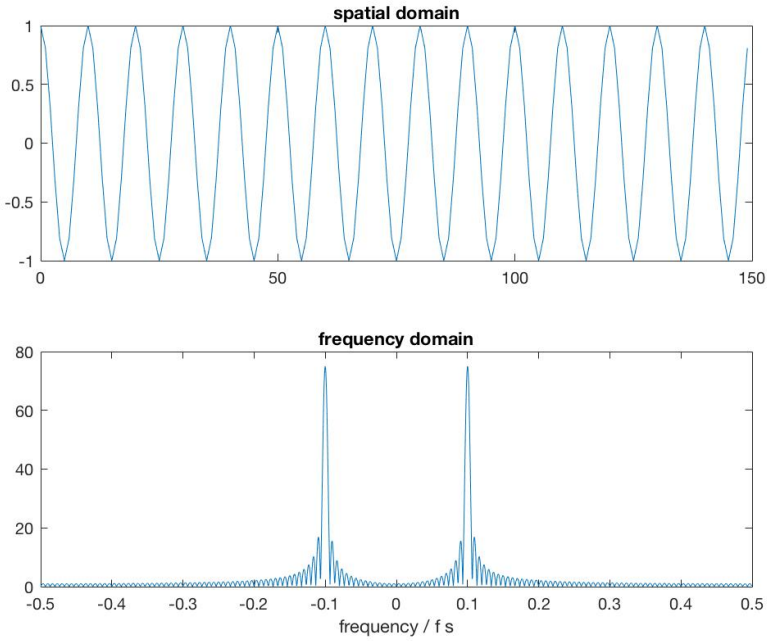


Figure 1: 1D Fourier Transform for $cos(\frac{2\pi x}{10})$

## 2.2 2D Fourier Transform [3 points]

Now you will implement 2D Fourier transform to convert an image from spatial domain to frequency domain by filling out necessary lines in `frequency.m`. The Fourier transform needs to be rearranged by shifting the zero-frequency component to the center of the array. You are allowed to call built-in MATLAB functions. Figure 2 shows the example results for image `dog.jpg`. Pick an arbitrary image other than the example one, save the resulting figure produced by your code, and include it in your write-up.

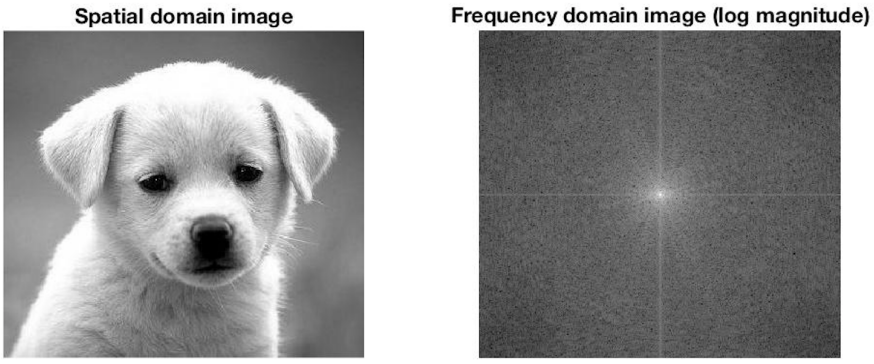


Figure 2: 2D Fourier Transform for dog.jpg

## 2.3 Sampling and Aliasing [8 points]

In the lecture of filters and frequency analysis, we have introduced how to subsample an image and fix aliasing. In this question, you are asked to first downsample an image, then fix the aliasing issue using the algorithm introduced in the class. (See slides P84-P93 for reference.) The implementation can be done by filling out the corresponding part in `frequency.m`. You are not allowed to call `imresize()` directly, but you can use other built-in MATLAB functions. Run your code on `zebra.jpg` and show the resulting figure in your report.

## 2.4 Low-pass Filtering and High-pass Filtering [12 points]

In this question, you will implement a procedure to do low-pass filtering and high-pass filtering on an input image. Both of the low-pass filter and high-pass filter should be **applied in frequency domain**. (See slides P76-P80 for reference) One low-pass filter you are recommended to try is Gaussian filter. Images after filtering are shown in spatial domain. You are allowed to call built-in MATLAB functions to do Fourier transform or create filters. Run your code on an image and include the resulting figure in your write-up.

# 3 Local Feature Descriptors and Matching

In this part, you will be implementing keypoint detector, feature descriptors and feature matching. Keypoint detectors find particularly salient points, such as corners, in an image upon which we can apply a feature descriptor. Once we have descriptors, we can use them to extract local features for each keypoint, and match keypoints between images by measuring feature similarities. The file you will work on in this section is `local_feature.m`.

## 3.1 Harris Corner Detector and Feature Matching [25 points]

1. Complete the implementation of function `harris_corners(I)`, which takes an image and outputs a response map. The response map is computed according to the corner response function.

2. Design and implement your own simple feature descriptor. One design choice you are recommended to try is to build a gradient histogram for the local patch.
   First, you need to complete the function `simple_descriptor(patch)`. Then, implement `describe_keypoints(I, keypoints)` to get local feature descriptors for all keypoints. (`simple_descriptor()` is called in `describe_keypoints()`)

3. Complete the function `match_descriptors(desc1, desc2)`. In this function, you will match the keypoints according to your descriptors. You can use any distance function mentioned in the class. Run your code on the provided images `img/building_1.jpg` and `img/building_2.jpg` and include the visualization results in your write-up.

## 3.2 SURF Features [4 points]

In the class, we have talked about SIFT feature descriptor. However, MATLAB doesn't have a built-in SIFT function because it is patented. Instead, you can try another local feature SURF. In this part, you need to find corresponding points between two images using SURF features. You are not asked to implement SURF by yourself, so feel free to call any relevant built-in functions. Run your code on the provided images `img/building_1.jpg` and `img/building_2.jpg` and include the visualization results in your write-up.

# 4 Filters in Convolutional Neural Networks

In this part, you will get a chance to see what CNN learns on a simple dataset. You are provided with MNIST dataset and a simple pretrained CNN on it. MNIST is a handwritten digit dataset which contains 60,000 training examples and 10,000 testing examples. All of the examples are $28 \times 28$ gray images with category labels 0 to 9. The file you are working on for this section is `cnn.m`. Before getting started, make sure that

1. Install MATLAB Deep Learning toolbox by running the following command in MATLAB:
   matlab.addons.supportpackage.internal.explorer.showSupportPackages('ALEXNET','tripwire')

2. Run the first three lines of `cnn.m` to load images and labels in training set and test set, load network model and print out the details of each layer. If everything works well, you should see the same results as in Figure 3.

```
Preparing MNIST data...
MNIST data preparation complete.

ans =

  15x1 Layer array with layers:

     1   'imageinput'    Image Input             28x28x1 images with 'zerocenter' normalization
     2   'conv_1'        Convolution             16 3x3x1 convolutions with stride [1  1] and padding [1  1  1  1]
     3   'batchnorm_1'   Batch Normalization     Batch normalization with 16 channels
     4   'relu_1'        ReLU                    ReLU
     5   'maxpool_1'     Max Pooling             2x2 max pooling with stride [2  2] and padding [0  0  0  0]
     6   'conv_2'        Convolution             32 3x3x16 convolutions with stride [1  1] and padding [1  1  1  1]
     7   'batchnorm_2'   Batch Normalization     Batch normalization with 32 channels
     8   'relu_2'        ReLU                    ReLU
     9   'maxpool_2'     Max Pooling             2x2 max pooling with stride [2  2] and padding [0  0  0  0]
    10   'conv_3'        Convolution             64 3x3x32 convolutions with stride [1  1] and padding [1  1  1  1]
    11   'batchnorm_3'   Batch Normalization     Batch normalization with 64 channels
    12   'relu_3'        ReLU                    ReLU
    13   'fc'            Fully Connected         10 fully connected layer
    14   'softmax'       Softmax                 softmax
    15   'classoutput'   Classification Output   crossentropyex with '0' and 9 other classes
```

Figure 3: Layer details of the pretrained CNN on MNIST

## 4.1 Filter Visualization [7 points]

Complete the provided code to visualize the 16 filters in the `conv1` layer and report your visualization results in your write-up. Select two filters and explain in your report what properties do each of the filter functions pick up.

## 4.2 Network Activation Visualization [3 points]

First, select 5 images you like from test set. Then, using the two filters you selected in section 4.1 to extract activations on each image, and show the visualization results in your write-up.

## 4.3 Image Retrieval [14 points]

First, select 3 images you like from test set. Then for each testing image, retrieve 5 most similar images from the first 1000 images in the training set. In your implementation, use Euclidean distance on the activations of `conv3` layer as the similarity metric.

## 5 Questionnaire [1 point]

Approximately how many hours do you spend on this homework? Please answer it in your write-up.