HW1

September 13, 2017

1 CSE327 Homework 1

Due date: 11:59 pm on Sep 27, 2017 (Wednesday)

1.1 ## Prerequisite

- **Install Python**: you can download it at https://www.python.org/downloads/ or use Anaconda (a Python distribution) at https://docs.continuum.io/anaconda/install/. Please use Python of version 2.7.x because some of the packages we use may not support the latest version of Python. Below are some materials and tutorials which you may find useful for learning Python if you are new to Python.
- https://docs.python.org/2.7/
- https://www.learnpython.org/
- http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html
- http://www.scipy-lectures.org/advanced/image_processing/index.html
- **Install Python packages**: install Python packages: `numpy`, `matplotlib`, `opencv-python` using pip, for example:

  ```
  pip install numpy matplotlib opencv-python
  ```

  Sometimes you may need to use command `pip2` instead of `pip`
- **Install Jupyter Notebook**: follow the instruction at http://jupyter.org/install.html to install Jupyter Notebook and get yourself familiar with it. *After you have installed Python and Jupyter Notebook, please open the notebook file 'HW1.ipynb' with your Jupyter Notebook and do your homework there.*

1.2 ## Example

Before you get down to do your homework, please read through the following example where we apply image thresholding to an image. This example uses OpenCV as the image processing toolkit, you are free to use other toolkit at your convenience.

```
In [3]: # import packages
        import cv2
        import numpy as np
        import matplotlib.pyplot as plt
        from IPython.display import display, Image

        print 'OpenCV version = ' + cv2.__version__
```

```
OpenCV version = 3.3.0
```

```
In [4]: # function for image thresholding
        def imThreshold(img, threshold, maxVal):
            assert len(img.shape) == 2 # input image has to be gray

            height, width = img.shape
            bi_img = np.zeros((height, width), dtype=np.uint8)
            for x in xrange(height):
                for y in xrange(width):
                    if img.item(x, y) > threshold:
                        bi_img.itemset((x, y), maxVal)

            return bi_img
```

```
In [5]: # read the image for local directory (same with this .ipynb)
        img = cv2.imread('snow.jpg')

        # convert a color image to gray
        img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        # image thresholding using global tresholder
        img_bi = imThreshold(img_gray, 127, 255)

        # Be sure to convert the color space of the image from
        # BGR (Opencv) to RGB (Matplotlib) before you show a
        # color image read from OpenCV
        plt.figure(figsize=(18, 6))
        plt.subplot(1, 3, 1)
        plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        plt.title('original image')
        plt.axis("off")

        plt.subplot(1, 3, 2)
        plt.imshow(img_gray, 'gray')
        plt.title('gray image')
        plt.axis("off")
        plt.subplot(1, 3, 3)
        plt.imshow(img_bi, 'gray')
        plt.title('binarized image')
        plt.axis("off")

        plt.show()
```

1.3 ## Description

In this homework you will work on basic image processing using Python. Be sure your Python has Image Processing Toolbox (OpenCV for example). However, for this homework do not use OpenCV unless explicitly instructed to. Download the image files (hw1_images.tgz) from Blackboard into your directory and then load image files using OpenCV with the imread command (see above example), and make sure to set your Jupyter Notebook's current directory to the directory containing the file. Please do use relative path for this homework.

There are seven images in total. The lena-noise.bmp is the same as the lena.bmp but corrupted with Gaussian noise. The barbara_noise.bmp is similarly corrupted barbara.bmp, and mandrill_noise.bmp is corrupted mandrill.bmp.

1.4 ## Problems

- **Problem 1 {15 pts}:** Write a function in Python that takes two arguments, a width parameter, w, and a variance parameter, s, and returns a 2D array containing a Gaussian kernel of the desired dimension and variance. The peak of the Gaussian should be in the center of the array. Make sure to normalize the kernel such that the sum of all the elements in the array is 1. Use this function and the OpenCV's `filter2D` routine to convolve the lena and lena_noise arrays with a 5 by5 Gaussian kernel with sigma of 1. Repeat with a 11 by 11 Gaussian kernel with a sigma of 3. Comment on the difference between the two images. Try the same thing to road and road_noise, barbara and barbara_noise, and to mandrill and mandrill_noise. Include your code and results in your Jupyter Notebook file.

```
In [1]: # write your code for Problem 1 here
```

- **Problem 2 {15 pts}:** The Gaussian kernel is separable, which means that convolution with a 2D Gaussian can be accomplished by convolving the image with two 1D Gaussian, one in the x direction and another in the y direction. Repeat the 2 convolutions you did in question 1 using this scheme. You can still use `filter2D` to convolve the images with each of the

1D kernels. Verify that you get the same results that you did in question 1 by computing the maximum difference between the arrays produced with the two methods. Include your code and results in your Jupyter Notebook file. (You don't have to turn in the images for this part)

```
In [1]: # write your code for Problem 2 here
```

- **Problem 3 {15 pts}:** Implement in Python the Sobel edge detector and apply it to the input image building.bmp. Then apply an appropriate threshold to the edge strength values to select what you think are the most important edges. Include your code and results in your Jupyter Notebook file.

```
In [2]: # write your code for Problem 3 here
```

- **Problem 4 {5 pts}:** Convolve an 11 by 11 Gaussian of sigma = 1 with the discrete approximation to the Laplacian kernel (i.e., [0 1 0; 1 -4 1; 0 1 0] or [1 1 1; 1 -8 1; 1 1 1]). Plot this 2D function using the `Matplotlib` function `plot` . Turn in this 3D plot using the `Matplotlib` function `plot_surface`. Do you see why this is referred to as the Mexican hat filter? Include your code and results in your Jupyter Notebook file.

```
In [3]: # write your code for Problem 4 here
```

- **Problem 5 {15 pts}:** Implement in Python an edge detector based on locating the zero crossings of the Laplacian. More specifically, write a routine that first convolves the input image with the Mexican hat kernel you produced in question 4 then searches the output array for locations where the value changes from positive to negative between neighboring pixels and the magnitude of the change is greater than some specified threshold. Test your program on the building image and find a threshold value which seems to yield acceptable results. Include your code and results in your Jupyter Notebook file.

```
In [ ]: # write your code for Problem 5 here
```

- **Problem 6 {35 pts}:** Implement in Python the first two steps of the Canny edge extraction algorithm: CANNY_ENHANCER(Gaussian filtering & finding magnitude and orientation of gradient) and NONMAX_SUPPRESSION. Apply these steps to all of the images then apply an appropriate threshold to the edge strength values to select what you think are the most important edges. Compare the outputs from the original images and the noisy ones. Also compare the outputs for images between the zero crossing and Canny detectors. Include your code and results in your Jupyter Notebook file.

```
In [ ]: # write your code for Problem 6 here
```

- **Problem 7 {15 pts extra credit}:** Implement in Python the third step of the Canny algorithm HYSTERESIS_THRESH. Select two appropriate thresholds for the road and road_noise image, assuming that you would want to use the results in a self driving car application. Include your code and results in your Jupyter Notebook file.

```
In [ ]: # write your code for Problem 7 here
```

1.5 ## Submission

Submission will be via Blackboard. The only file you need to submit is this .ipynb notebook file with your answers. **NO** need to submit the resulting images separately.