```
# -*- coding: utf-8 -*-
"""
Course: CS 4365/5354 [Computer Vision]
Author: Jose Perez [ID: 80473954]
Assignment: Exercise 3
Instructor: Olac Fuentes
Last Modification: September 7th, 2016 by Jose Perez
"""
from timeit import default_timer as timer
from PIL import Image
from numpy import *
from pylab import *
from scipy.ndimage import filters

import sys

# Print iterations progress
def printProgress (iteration, total, prefix = '', suffix = '',
    """
    Call in a loop to create terminal progress bar
    @params:
        iteration   - Required  : current iteration (Int)
        total       - Required  : total iterations (Int)
        prefix      - Optional  : prefix string (Str)
        suffix      - Optional  : suffix string (Str)
        decimals    - Optional  : positive number of decimals :
        barLength   - Optional  : character length of bar (Int
    """
    formatStr       = "{0:." + str(decimals) + "f}"
    percents        = formatStr.format(100 * (iteration / floa
    filledLength    = int(round(barLength * iteration / float(
    bar             = '' * filledLength + '-' * (barLength -
    sys.stdout.write('\r%s |%s| %s%s %s' % (prefix, bar, percer
    sys.stdout.flush()
    if iteration == total:
        sys.stdout.write('\n')
        sys.stdout.flush()

def question1(img):
    im = array(Image.open(img).convert('L'))
```

```
    Image.fromarray(im).save(img + '_gray.png')

    # Sobel
    start_time = timer()
    imx_sobel = zeros(im.shape)
    filters.sobel(im, 1, imx_sobel)

    imy_sobel = zeros(im.shape)
    filters.sobel(im, 0, imy_sobel)

    magnitude_sobel = sqrt(imx_sobel**2+imy_sobel**2)

    end_time = timer()
    print('[Sobel] Duration: ' + str(end_time - start_time))

    # Save sobel images
    Image.fromarray(imx_sobel).convert('RGB').save(img + '_sob
    Image.fromarray(imy_sobel).convert('RGB').save(img + '_sob
    Image.fromarray(magnitude_sobel).convert('RGB').save(img +

    # Prewitt
    start_time = timer()
    imx_prewitt = zeros(im.shape)
    filters.prewitt(im, 1, imx_prewitt)

    imy_prewitt = zeros(im.shape)
    filters.prewitt(im, 0, imy_prewitt)

    magnitude_prewitt = sqrt(imx_prewitt**2+imy_prewitt**2)

    end_time = timer()
    print('[Prewitt] Duration: ' + str(end_time - start_time))

    Image.fromarray(imy_prewitt).convert('RGB').save(img + '_p
    Image.fromarray(imx_prewitt).convert('RGB').save(img + '_p
    Image.fromarray(magnitude_prewitt).convert('RGB').save(img

def question2(img, n):
    im = array(Image.open(img).convert('L'))
```

```
    devImage = im

    count = 0
    total = len(im) * len(im[0])

    start_time = timer()
    for row in range(len(im)):
        for col in range(len(im[row])):
            count += 1
            observations = []

            for i in range(n):
                for j in range(n):
                    new_row = row + i
                    new_col = col + j

                    # Check bounds
                    if 0 <= new_row < len(im) and 0 <= new_col
                        observations.append(im[new_row][new_co

            devImage[row][col] = std(observations)
        if count % 3 == 0:
            printProgress(count, total)

    end_time = timer()
    print('[Problem 2, O(rcn^2)] Duration: ' + str(end_time - 

    Image.fromarray(devImage).convert('RGB').save(img + '_std.

    import cPickle as pickle
    f = open(img + '_std_raw.pkl', 'wb')
    pickle.dump(devImage, f)
    f.close()

    imshow(devImage)
    show()

def question3(filename, n):
    img = Image.open(filename)
    im = array(img.convert('L'))
```

```
start_time = timer()

im_sq = im ** 2

integral = im.cumsum(axis=0).cumsum(axis=1)

D = integral[:,:]
A = integral[:, :]
B = integral[:,:]
C = integral[:,:]
integral_sum = A + D - B - C
integral_mean = integral_sum / (n)

integral_sq = im_sq.cumsum(axis=0).cumsum(axis=1)
D_sq = integral_sq[:,:]
A_sq = integral_sq[:, :]
B_sq = integral_sq[:,:]
C_sq = integral_sq[:,:]
integral_sq_sum = A_sq + D_sq - B_sq - C_sq
integral_sq_mean = integral_sq_sum / (n)

deviation = sqrt(integral_sq_mean - (integral_mean * integ

end_time = timer()
print('[Problem 3, O(rc)] Duration: ' + str(end_time - sta


imshow(deviation)
show()
```