The University of Saskatchewan

Saskatoon, Canada

Department of Computer Science

487/819– Computer Vision and Image Processing

Assignment 2

Date Due: October 17, 2018

Total Marks: 43

1 Submission Instructions

- Assignments are to be submitted via the Moodle class page.
- Programs must be written in Python 3.
- No late assignments will be accepted. See the course syllabus for the full late assignment policy for this class.

2 Background

The purpose of this assignment is to gain some experience with low-level image features such as edges and blobs while looking at some interesting applications of those features.

2.1 Assignment Synposis

In Question 1 you'll use implement the vector gradient algorithm for edge detection in colour images and use it to develop a measure of image sharpness that can be used to quantify how much an image has been blurred. In question 2 you'll practice using the blob detection functions in `skimage` to solve an area-of-interest detection problem, and provide a qualitative analysis of the performance.

2.2 Implementing the Vector Gradient

To implement the vector gradient we need to use the equations in Lecture Topic 5. They are repeated here. First the definitions of g_{xx}, g_{yy} and g_{xy}.

$$g_{xx} = \mathbf{u} \cdot \mathbf{u} = \mathbf{u}^T \mathbf{u} = \left(\frac{\partial R}{\partial x}\right)^2 + \left(\frac{\partial G}{\partial x}\right)^2 + \left(\frac{\partial B}{\partial x}\right)^2$$

$$g_{yy} = \mathbf{v} \cdot \mathbf{v} = \mathbf{v}^T \mathbf{v} = \left(\frac{\partial R}{\partial y}\right)^2 + \left(\frac{\partial G}{\partial y}\right)^2 + \left(\frac{\partial B}{\partial y}\right)^2$$

$$g_{xy} = \mathbf{u} \cdot \mathbf{v} = \mathbf{u}^T \mathbf{v} = \frac{\partial R}{\partial x}\frac{\partial R}{\partial y} + \frac{\partial G}{\partial x}\frac{\partial G}{\partial y} + \frac{\partial B}{\partial x}\frac{\partial B}{\partial y}$$

These will manifest themselves in your solution as 2D arrays that are the same size as the input image. The first step is to compute the horizontal and vertical partial derivatives of the red, green and blue channels. This is straightforward and can be done by applying the Sobel operators to each colour image channel. If you then stack the three horizontal filtering results back into an $N \times M \times 3$ array (N and M

are the image height and width), the result is u. Do the same for the vertical partial derivatives and you get v.

g_{xx} is the dot product of u with itself. If you construct u as described above, you can get g_{xx} using the provided function `color_dot_product()`. g_{yy} and g_{xy} can be computed similarly.

Once you have g_{xx}, g_{yy} and g_{xy}, in the form of $N \times M$ arrays, computation of the gradient direction and magnitude are as follows:

$$\theta(x,y) = \frac{1}{2}\tan^{-1}\left[\frac{2g_{xy}}{g_{xx}-g_{yy}}\right]$$

$$F_{\theta}(x,y) = \left\{\frac{1}{2}[(g_{xx}+g_{yy}) + (g_{xx}-g_{yy})\cos(2\theta(x,y)) + 2g_{xy}\sin(2\theta(x,y))]\right\}^{\frac{1}{2}}$$

These can be computed straightforwardly for all pixels at once by using `numpy` and performing each mathematical operation on the entire array, avoiding expensive loops. The results will again be 2D arrays which are the image's gradient magnitude image and gradient direction image. Note that the direction image is needed to compute the magnitude image.

2.3 The Kurtosis-based Sharpness Measure

In question 1 we'll be using a measure of image sharpness based on *kurtosis*. Kurtosis is a numeric property of a group of numbers, much like mean or standard deviation.

Given a gradient magnitude image, we can treat each gradient magnitude value in the image as one sample in a set of samples. The kurtosis of that set of samples is related to image sharpness (i.e. lack of blurriness). The kurtosis of a set of samples is calculated from statistical moments. The mean of a set of samples is the first central moment; the standard deviation is the second central moment. The kurtosis is the fourth central moment divided by the second central moment (standard deviation) minus 3. It turns out that

$$\textit{sharpness} = \log(K+3)$$

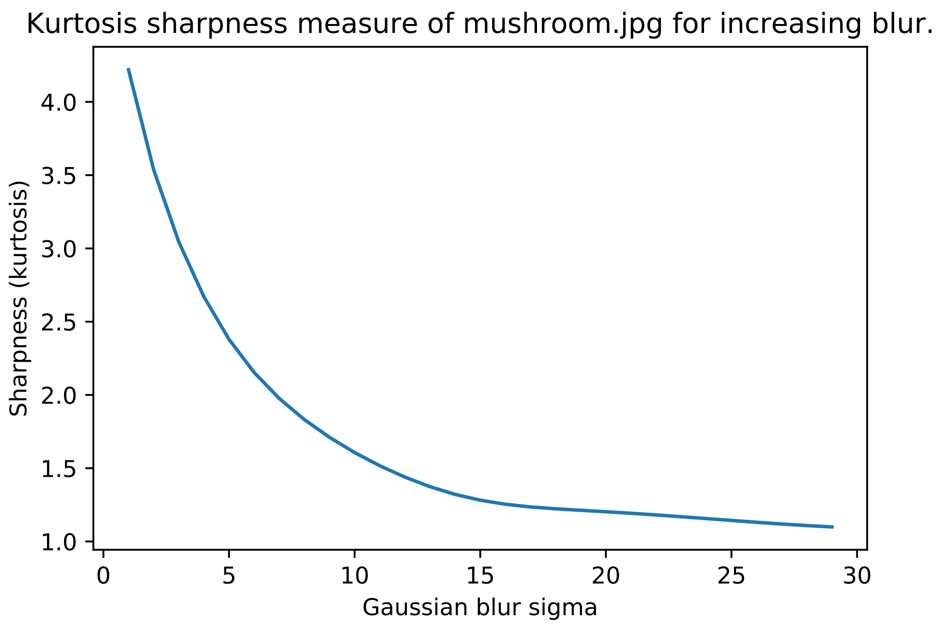
where K is the kurtosis of an image's set of gradient magnitudes is a reasonable measure of sharpness (the $+3$ is to offset the -3 in the definition of kurtosis so we aren't taking logarithms of negative numbers).

You can compute the kurtosis of a set of samples using the `scipy.stats.kurtosis()` function (requires importing the `scipy.stats` module). To compute a measure of how blurry an image is, you can't quite just pass the entire gradient magnitude image to `scipy.stats.kurtosis()`. The function expects a one-dimensional array, so you have to use the `numpy.reshape()` function to rearrange the gradient magnitude image into a one-dimensional array of length MN rather than a 2D array with N rows and M columns. Then you can pass the 1D array to the `scipy.stats.kurtosis()` function to obtain K. Once you have that, just add 3 and take the logarithm. And that's our sharpness measure.

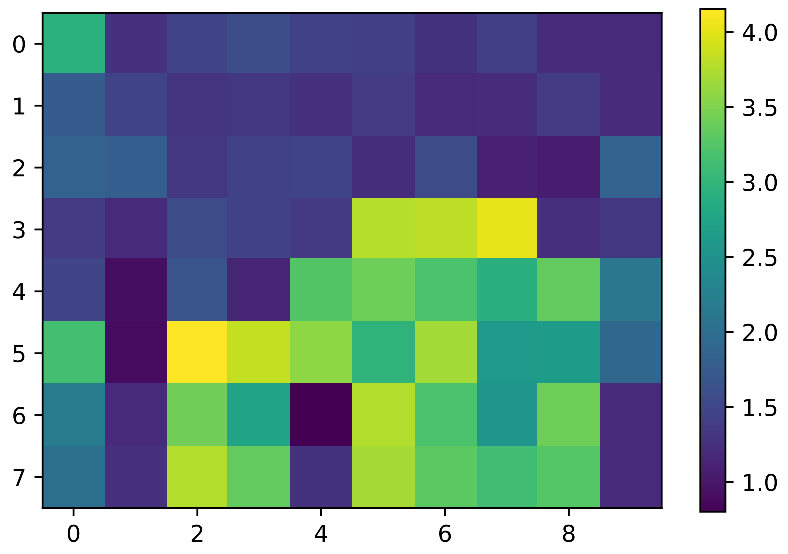
3 Problems

Question 1 (32 points):

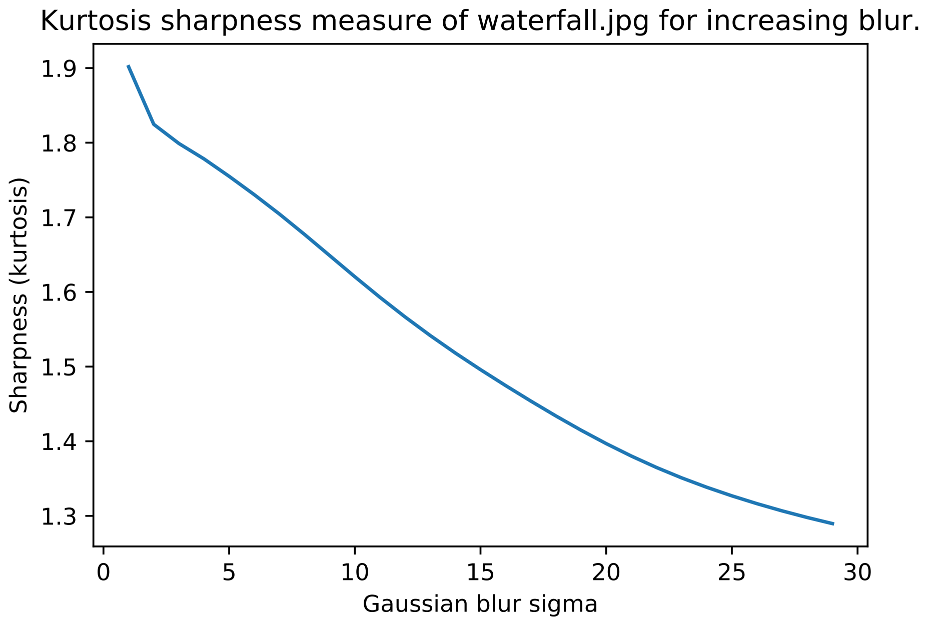
Detailed instructions are provided `asn2-q1.ipynb`. Sample outputs are given below.

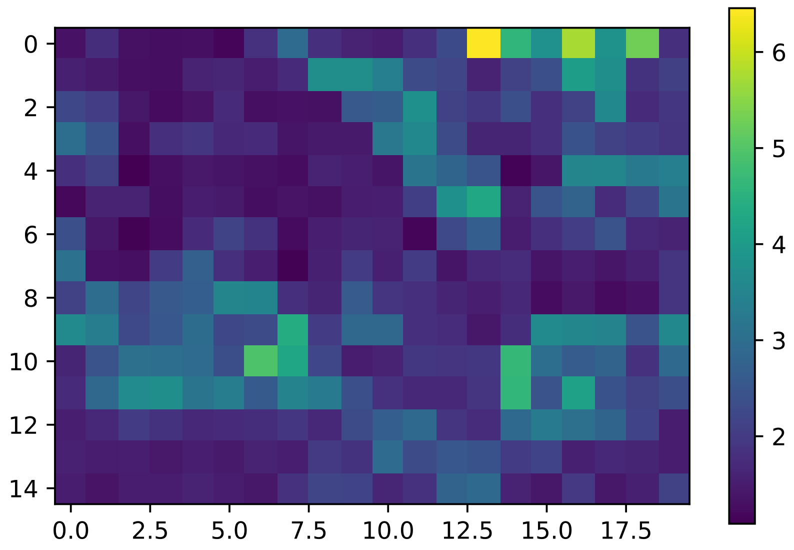


Sample Output for Step 2



Sample Output for Step 3

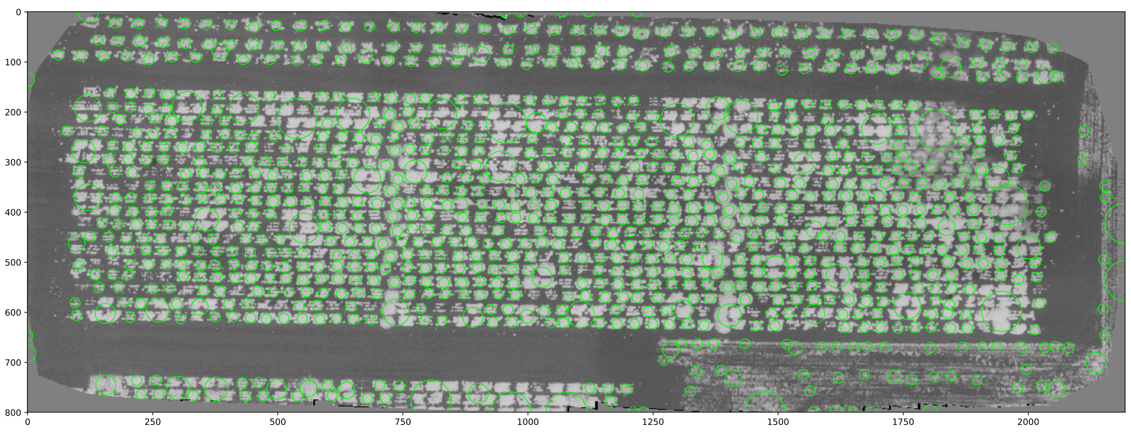




Sample Output for Step 4

Question 2 (11 points):

Detailed instructions are provided in `asn2-q2.ipynb`. Sample output is given below.



The grading rubric can be found on Moodle.

Ackonwledgement

Thanks to Dr. Kirstin Bett, University of Saskatchewan for providing the lentil field image. Used and distributed with permission.

4 Files Provided

`asn2-qX.ipynb`: These are iPython notebooks, one for each question, which includes instructions and in which you will do your assignment.

5 What to Hand In

Hand in your completed iPython notebooks, one for each question.