**COMP 558: Assignment 1**
**Available: Saturday, October 5th, 2013**
**Due Date: Friday, October 25th, 2013 (before midnight) via mycourses.**

**Notes:** You are encouraged to become familiar with the **MATLAB** environment which is currently installed on Unix machines in the CS labs and to use it for the experimental parts of the assignment. Morteza has already had a tutorial on its use and some helpful code is posted on mycourses. Hints on assignment 1 will be provided once you get going and have further questions. I EXPECT EVERYONE TO SUBMIT ORIGINAL WORK FOR THIS ASSIGMENT. This means that if you have consulted anyone or any sources (including source code), you must disclose this explicitly. Anything you submit reflects your own work. Your submission should be in the form of an electronic report (PDF), which includes a description of what you did, answers to the specific questions, and a presentation and discussion of your results. Submit code that you have written to generate your results as a separate .zip file.

## Question 1: Convolution and Edge Detection (10 marks)

For this question you will test your implementations in parts e) and f) on the two images for edge detection located at:
`http://www.cim.mcgill.ca/~morteza/COMP558.php`.
A common strategy to detect edges which is consistent with biological processing is to convolve the image with oriented filters which look like second derivatives of 2D Gaussians in one direction, and to find zero crossings of the result. This is motivated by the idea that edges correspond to maxima of a first derivative or zero-crossings of a second derivative. The convolution with a Gaussian has to do with the idea of limiting frequencies (or smoothing out "noise") in the input.

a) The expression of a normalized 2D Gaussian centered at the origin $(0, 0)$ is given by

$$G(x, y) = \frac{1}{\sqrt{2\pi\sigma^2}} \left( e^{-\frac{x^2+y^2}{2\sigma^2}} \right)$$

Calculate the first and second partial derivatives of $G(x, y)$ with respect to the variable $x$ and write down the results. This corresponds to a directional derivative of $G(x, y)$ in the horizontal direction, i.e., with $y$ held fixed.

b) Using Matlab and appropriate plotting functions related to mesh and surface plots (e.g. mesh or surf) create visualizations of $G(x, y)$ and the first two partial derivatives you calculated above for $\sigma = 1, 2, 4$. In your plots you can set the range of $x$ and $y$ to each be $(-3\sigma, 3\sigma)$. This is because 98% of the area under a Gaussian is within 3 standard deviations of its center.

c) Now consider the operation of rotating the above 2D functions about $(0, 0)$ by an angle $\theta$. This can be done in a very easy fashion by using the fact that if you write the

point $(x, y)$ as a column vector a rotation by $\theta$ can be obtained by premultiplying by a $2 \times 2$ matrix whose terms involve $\sin(\theta)$ and $\cos(\theta)$. Using this strategy write a Matlab function to create a rotated copy of $G(x, y)$ and its first two partial derivatives and demonstrate that it works correctly by visualizing the rotated copies, for a few values of $\theta$, for a chosen value of $\sigma$.

d) Now consider the second derivative kernel you have created above in a) (prior to the rotation step) but premultiply it by a Gaussian oriented along the $y$ axis so as to limit its spatial extent in the $y$ direction. By a Gaussian oriented along the y-axis I mean one that has no dependence on $x$, i.e., it has the form $G(x, y) = \frac{1}{\sqrt{2\pi\sigma_1^2}}\left(e^{-\frac{y^2}{2\sigma_1^2}}\right)$. By this premultiplication you will be creating something that looks very much like the receptive field corresponding to a simple cell, following which you can also create rotated copies of the result using the answer to part c) you have developed. Illustrate this for a chosen value of the $\sigma, \sigma_1$ parameters.

e) Now consider 8 orientations $\theta = 0, \pi/8, \pi/4, 3\pi/8, \pi/2, 5\pi/8, 3\pi/4, 7\pi/8$. Over a range of standard deviations $\sigma = 1, 2, 4, 8$ convolve the input images with the kernel from part d) but rotated by $\theta$ and detect zero-crossings of the result. You can pick a suitable (but fixed) value for $\sigma_1$. For each $\sigma$ show the detected zero crossings at each direction $\theta$ superposed on the original image. (In other words, for each image you should have 4 figures, one for each $\sigma$). Comment on and discuss the results you obtain.

f) You are now going to consider a different approach to edge detection, which is to directly convolve the image with the Laplacian of a 2D Gaussian and detect zero crossings of the result. In class and in the Marr-Hildreth paper we presented the form of the 2D Laplacian of a Gaussian in polar coordinates. For each of the $\sigma$'s used earlier, i.e., $\sigma = 1, 2, 4, 8$ show your results overlayed on the original image. As in part e) for each image you should have 4 figures, one for each $\sigma$. Comment on and discuss the results you obtain, in comparison to the ones obtained in part e). Keep in mind that the Marr-Hildreth edge detection strategy rests on the condition of linear variation holding.

## Question 2: Shape from Shading (10 marks)

Using the illumination direction $\vec{i}$ as an input variable (you will have to play with particular choices) and the integrability constraints for $p$ and $q$ covered in class, implement the discrete form of the Euler-Lagrange Equations to recover the depth map from the shaded images provided for this question which will be located at: `http://www.cim.mcgill.ca/~morteza/COMP558.php`.

You can use the following discrete approximation of the Euler-Lagrange equations that

uses finite differences over pixel coordinates $(i, j)$:

$$-4p_{i,j} + p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1} = -\frac{1}{\lambda}\left(E_{i,j} - R(p_{i,j}, q_{i,j})\right.\frac{\partial R}{\partial p} \tag{1}$$

$$-4q_{i,j} + q_{i+1,j} + q_{i-1,j} + q_{i,j+1} + q_{i,j-1} = -\frac{1}{\lambda}\left(E_{i,j} - R(q_{i,j}, q_{i,j})\right.\frac{\partial R}{\partial q} \tag{2}$$

Experiment with different values of the regularization parameter $\lambda$. What is the impact of changing it? Display your findings and results using suitable functions in Matlab.

**Hint**: To implement the shape from shading algorithm, you will first need to initialize $p$ and $q$. Explore different possibilities and see if it gives you a faster and/or better convergence.