ECE4580 Homework #8

Due: Mar. 6, 2014

Problem 1. (50 pts) The Laplacian of Gaussian filter is a circular, symmetric filter that seeks out a specific image structure in images. This structure is blob-like, which will be explored in this homework.

(a) The closed-form equation for the Laplacian of Gaussian operator is

$$LoG = \Delta G_\sigma = \frac{\partial^2}{\partial x^2} G_\sigma + \frac{\partial^2}{\partial xy^2} G_\sigma = \frac{(x^2 + y^2 - 2\sigma^2)}{2\pi\sigma^6} e^{-(x^2+y^2)/2\sigma^2}$$

Generate an $x - y$ grid with the range $[-10,\ 10]$ in both coordinates, then evaluate the Laplacian of Gaussian on the grid. Plot the resulting values in Matlab using the mesh function. Why would what you see relate to blob detection?

(b) Using the `dots` image provided, identify what the Gaussian σ value should be that best captures the blobs. What is the radius of the blob? How do they relate? Can you figure out or read online what the relationship between σ and the blob radius is?

(c) Using the `sunflower` image provided, apply the Laplacian of Gaussian filter to it and try to get the best response possible. What value of σ did you use and why?

(d) Sometimes the Laplacian of Gaussin (LoG) is used as a feature detector (here the image feature is a blob). For example, one can detect then try to match blobs in a stereo image pair to identify point correspondences for stereo depth. You will need to identify a threshold for the LoG. Flesh out the function called `findBlobPts.m` which is provided on t-square. There is a function called `runBlobs.m` that you can flesh out and run. The code has comments that should help you out (in particular, pay attention to the normalization comment).

Run it on either the dots or the sunflower image, on a TSRB or campus image, as well as one of the images from images zip file (which also contains images I got from google using keywords). Vary the window size (and hence blob scale) until you get results that you like. What kinds of points does it find? What window size did you use to get the results for the images?

Note: Get to know the Matlab functions `meshgrid` *and* `mesh`. *They will be helpful for evaluating the LoG operator from part (a). Also, you should be able to exploit the fact that Matlab can perform operations on entire arrays to avoid for loops. This is helpful for the squared terms and exponential part. Try to get part (a) as close to 4-5 lines of code as possible. If you can do that, you are on the right track as far as Matlab coding simplicity. In Matlab, you should be able to write code that almost looks like the equation.*

Files: There should be some Matlab files to help you out. The one called `viewBlobScore.m` *can be used for the (a)-(c). If desired, a code stub called* `blobs.m` *can be used for these parts to avoid code repetition. For part (d), the code stubs are the* `runBlobs.m` *and* `findBlobPts.m` *files.*

Problem 2. (20 pts) Many times in computer vision, it is very difficult to find the optimal value of a function by plotting the function because the vector to optimize has high dimensions. The only values one has access to are the gradients. This problem puts you in the same kind of predicament. I have created a 2D function whose minmimal value is to be estimated using gradient descent. You know that the space is limited to $x \in [0, 10]$ and $y \in [0, 10]$. Load the file `gradient.mat` and use the function **gradF** which gives the gradient of the function $F : \mathbb{R}^2 \to \mathbb{R}^2$ to find the minimal value of $F(\vec{x})$ on the domain $[0, 10] \times [0, 10]$.

Note: The argument to `gradF` *is a 2x1 position vector. The return value is a 2x1 motion vector.*

Problem 3. (25 pts) With the homework, you will find a Matlab file called `templates.mat` which has two templates `temp1` and `temp2` plus three images `I1`, `I2`, and `I3`. Your job is to find the template locations in the three images. You are to turn in, not only the three locations, but also your parameters (`c`, `nsteps`, and `pos`), and the lines of code that you have modified in the findTemplate program. Note that the gradient function has been provided to you as `gradTempMatch.m` so all you have to do is work out the gradient descent optimization. Turn in the program so that it works for one of the cases. It is OK to have output to figures since the output is part of the original code stub (but do not open anything else beyond what the code stub already does).

Play around with the initial condition and try to estimate how close to the target you need to be for the gradient descent to work. Find an initial condition that you think should work but doesn't and turn in the script for that case with `loseTempMatch.m` as the script name. In the homework submission document, specify the initial point and the final (converged) point. Also turn in the plots of the functioning and failing gradient descent attempts.

Note: You can also play around with the blurring/smoothing of the image to see what effect that has on the convergence of the gradient descent.

Problem 4. (30 pts) Each group has a baby step to perform for the following Thursday. Your homework is to complete that baby step and report on it. You should roughly reiterate the baby step, discuss how it was accomplished, demonstrate through images (if possible) functionality, then briefly note any observations (including cases where it may not work so faithfully, or what is needed to work very well). If your project involves video and cannot be demonstrated with still images, then present it to your group contact by Friday.