COMPSCI 527- Homework 4

Due on November 20, 2014

Questions may continue on the back. Type your text. You may write math formulas by hand. What we cannot read, we will not grade.

You may talk about this assignment with others, but do not write down anything while you talk. After that, do the assignment alone. What you hand in must be your work only.

IMPORTANT: Implement all the functions in this assignment from scratch. That is, do not use anyone else's code, unless otherwise instructed. **Email your solution as a single PDF file to Ben at the beginning of class on the due date.** Remember that class will not meet on November 20.

This assignment introduces several new concepts, so be patient and careful as you read. You will hopefully learn something new. You will need the file `data.mat` and a MATLAB function, all available in a zip file on the homework page.

Image classification and regression problems take as inputs descriptors of the image or of parts of it. These descriptors are called *features*. As not all parts of an image are interesting, it is useful to develop an operator that selects image regions that are worth describing. This selection is difficult in general, because what is "interesting" is a semantic notion that depends on the application. In this assignment, we will develop a syntactic "interest operator" instead. This operator is given a window (a small square of contiguous pixels) out of a gray-level image and returns a measure of the *local distinctiveness* of the image within the window.

In this context, a window W is "locally distinctive" if nearby windows are sufficiently different from W, in a sense to be made more precise later. To formalize this notion, we first need to define what it means for two image windows to be similar or dissimilar form each other. Let

$$D(\mathbf{x},\mathbf{y}) = \sum_{\mathbf{z}\in\mathbb{Z}^2} w(\mathbf{z})\,[I(\mathbf{z}+\mathbf{y}) - I(\mathbf{z}+\mathbf{x})]^2 \quad \text{for} \quad \mathbf{x},\mathbf{y}\in\mathbb{R}^2 \tag{1}$$

be a measure of the *dissimilarity* between two windows centered at $\mathbf{x}$ and $\mathbf{y}$ in image I. In this expression, $\mathbb{Z}$ is the set of integers, the summation is over two scalar variables $\mathbf{z} = (z_1, z_2)^T$ (where the subscript T denotes transposition), and $w(\mathbf{z})$ is a discrete, truncated, normalized Gaussian function with parameter σ:

$$w(\mathbf{z}) = \begin{cases} ce^{-\frac{z_1^2+z_2^2}{2\sigma^2}} & \text{for } -h \le z_1 \le h \text{ and } -h \le z_2 \le h \\ 0 & \text{elsewhere} \end{cases} .$$

Here,

$$h = \lceil 2.5\sigma \rceil$$

is the smallest integer that is no smaller than 2.5σ, and c is a real-valued constant such that

$$\sum_{\mathbf{z}\in\mathbb{Z}^2} w(\mathbf{z}) = 1 .$$

The "window" is then an $n \times n$ square with $n = 2h + 1$. When $\mathbf{x}$ or $\mathbf{y}$ are not integer-valued, image values between pixels can be computed by bilinear interpolation. However, we do not worry about the values of $D(\mathbf{x},\mathbf{y})$ for real-valued arguments in this exercise, other than in a conceptual sense.

The meaning of $D(\mathbf{x},\mathbf{y})$ is straightforward. The truncated Gaussian $w(\mathbf{z})$ selects small areas of the image around $\mathbf{x}$ and $\mathbf{y}$. Discrepancies between corresponding pixel values in those areas are measured by the square of their difference, and weighted (through multiplication by $w(\mathbf{z})$) according to how close the pixels are from the center of each window. The weighted discrepancies are then added to each other to produce a single number $D(\mathbf{x},\mathbf{y})$. If the image values in the two small areas were identical to each other, this number would be zero, and the number increases as the contents between the two windows grow different from each other.

1. In this problem we get some intuition as to what the function $D(\mathbf{x},\mathbf{y})$ looks like when $\mathbf{x}$ is kept fixed and $\mathbf{y}$ is varied around $\mathbf{x}$. To do this, we set

$$\mathbf{y} = \mathbf{x} + \mathbf{d}$$

so that $\mathbf{d}$ is the *displacement* between a window centered at $\mathbf{x}$ and another one centered at $\mathbf{y}$. We then pick three qualitatively different windows centered at $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$ in a test image, and plot $D(\mathbf{x}^{(i)}, \mathbf{x}^{(i)} + \mathbf{d})$ for the 81 integer displacement vectors

$$\mathbf{d} \in R = \{\mathbf{d} \mid \mathbf{d}\in\mathbb{Z}^2 \text{ and } \mathbf{d} = (d_1, d_2)^T \text{ with } -4 \le d_1 \le 4 \text{ and } -4 \le d_2 \le 4\}$$

(the letter 'R' stands for the *range* of displacements). The set R is a square in "displacement space" and $D(\mathbf{x}^{(i)}, \mathbf{x}^{(i)} + \mathbf{d})$ for fixed $\mathbf{x}^{(i)}$ is a function from the integer grid in this space into the non-negative reals.

(a) Write a MATLAB function with header

```
function [w1, u] = gauss(sigma)
```

that computes a discrete, truncated, one-dimensional $n \times 1$ Gaussian kernel `w1` with parameter σ, such that

```
w = w1 * w1';
```

is the two-dimensional Gaussian kernel $w(\mathbf{z})$ defined above. Note that if `w1` is normalized to sum to 1, then `w` sums to 1 as well. [You may want to check this.]

The second output argument `u` is the vector `-h:h` where `h` is the half-width of the window (same as h in the problem statement above).

Hand in your code for `gauss` and show a plot of `w1` versus `u` for $\sigma = 3$. Label the axes appropriately.

(b) Write a MATLAB function with header

```
function D = dissimilarity(I, x, y, sigma)
```

that computes the dissimilarity $D(\mathbf{x}, \mathbf{y})$. The input parameter `I` is a gray-level image with pixels of type `uint8`. The vectors `x` and `y` are 2×1 vectors with *integer* coordinates (it is fine to use variables of type `double` for `x` and `y`, but the values must be integer). The positive scalar `sigma` is the parameter σ used in the definition of D. The output `D` is a scalar of type `double`. Use your function `gauss` to generate the weight mask $w(\mathbf{z})$.

The first coordinate of $\mathbf{x}$, $\mathbf{y}$, $\mathbf{d}$ is a row index, and the second coordinate is a column index.

Your code should abort with an error message if either `x` or `y` is too close to the image boundary and parts of the corresponding window would fall outside the image. It is OK to use `for` loops to scan the range R. Make sure you cast either the whole image or the relevant parts of it to `double` before doing any calculations with pixel values.

Hand in your code for the function `dissimilarity`, as well as three mesh plots of $D(\mathbf{x}^{(i)}, \mathbf{x}^{(i)}+\mathbf{d})$ for the image `shadow.jpg` provided with this assignment and for $i = 1, 2, 3$ and for $\mathbf{d}$ in the range R defined earlier. For $\mathbf{x}^{(i)}$ use the three columns of the matrix `X` in the `data.mat` file provided with this assignment (use `load data` to load `X` into your workspace). Set $\sigma = 3$ for all plots.

To display a mesh plot use the MATLAB function `mesh` (use the MATLAB `help` to figure out how `mesh` works). Before printing the mesh plot, set its viewing direction (by hand or with the `view` function) so that the plot is well visible. What MATLAB calls the y axis should be marked with label d_1 and the x axis should be marked with label d_2. The tick marks on the axes should be labeled from -4 to 4 on each axis. Do *not* hand in the code used to make the plots.

(c) The three windows centered at $\mathbf{x}^{(i)}$ for $i = 1, 2, 3$ in the previous question contain respectively a rather flat part of the image ($i = 1$), an edge ($i = 2$), and a corner ($i = 3$). You may want to visually explore these windows using MATLAB's display facilities.

In what way do the three functions $D(\mathbf{x}^{(i)}, \mathbf{x}^{(i)} + \mathbf{d})$ differ from each other qualitatively? More specifically, the function $D(\mathbf{x}, \mathbf{x} + \mathbf{d})$ is nonnegative and is zero for $\mathbf{d} = \mathbf{0}$, so in some sense it is supposed to look like a bowl (a convex paraboloid) in the vicinity of $\mathbf{d} = \mathbf{0}$. How well-defined is the bottom of the bowl in each of the three different situations? In answering this question, pay attention also to the actual values of D (the values on the vertical axis of your plots), and keep in mind that the image is noisy. Because of this, small fluctuations in the value of D are not significant.

2. We now use the intuition developed with the previous exercise to come up with a definition of "interestingness." An image window is "interesting" if the bottom of the bowl formed by $D(\mathbf{x}, \mathbf{x} + \mathbf{d})$ as a function of $\mathbf{d}$ is well defined. When this is the case, there is enough going on in the window around $\mathbf{x}$ that sliding the window in any direction makes the window look different from its original version. In particular, a flat part of the image is not interesting, nor is a window along a straight edge. A corner is interesting, and so is an image detail with rich textural content (for instance, a polka-dot pattern, gravel, or a fine checkerboard).

To measure this notion of "interestingness," one could in principle build the bowl for each window in the image and then somehow measure how well defined the bottom of the bowl is. A more efficient way to achieve the same result is to note that $D(\mathbf{x}, \mathbf{x}+\mathbf{d})$ must look like a paraboloid in an infinitesimally small region around $\mathbf{d} = \mathbf{0}$, because in such a small region one can approximate the function D by its Taylor series truncated after the second-order term. The derivation that follows computes the smallest curvature of this second-order term, and calls this curvature the "distinctiveness" (or "interestingness") of the window centered at $\mathbf{x}$.

Imagine fixing $\mathbf{x}$ and seeing $D(\mathbf{x}, \mathbf{x} + \mathbf{d})$ as a function of $\mathbf{d}$ only. If you take $\mathbf{d} = \mathbf{0}$, the two windows centered at $\mathbf{x}$ and $\mathbf{x} + \mathbf{d}$ are then the same window, and

$$D(\mathbf{x}, \mathbf{x}) = 0 \; .$$

In addition, since D cannot get any smaller than zero, $\mathbf{d} = \mathbf{0}$ is also a minimum of $D(\mathbf{x}, \mathbf{x}+\mathbf{d})$ with respect to $\mathbf{d}$, and therefore the gradient there is zero:

$$\left.\frac{dD(\mathbf{x}, \mathbf{x}+\mathbf{d})}{d\mathbf{d}}\right|_{\mathbf{d}=\mathbf{0}} = 0 \; .$$

If you move away from $\mathbf{d} = \mathbf{0}$ in any direction, the function D cannot decrease. It either stays constant, which means that the window at $\mathbf{x}+\mathbf{d}$ looks exactly the same as the window at $\mathbf{x}$, or it increases. In other words, the function $D(\mathbf{x}, \mathbf{x}+\mathbf{d})$ with $\mathbf{x}$ fixed looks like the bottom of a bowl in a small neighborhood of $\mathbf{d} = \mathbf{0}$: The gradient at the bottom is zero, and the curvature of the bowl may be different as you move in different directions away from $\mathbf{d} = \mathbf{0}$, but is always nonnegative.

The *distinctiveness* of the window at $\mathbf{x}$ is defined as the curvature of the bowl at $\mathbf{d} = \mathbf{0}$ in the direction in which it curves the least. Specifically, we say that a window centered at $\mathbf{x}$ has distinctiveness $\lambda_{\min}(\mathbf{x})$ if

$$\min_{\{\mathbf{u} \in \mathbb{R}^2 \;\mid\; \|\mathbf{u}\|=1\}} \frac{1}{2} \left.\frac{d^2 D(\mathbf{x}, \mathbf{x}+a\mathbf{u})}{da^2}\right|_{a=0} = \lambda_{\min}(\mathbf{x}) \; . \tag{2}$$

Let us parse this definition. Imagine starting at the bottom of the bowl at $\mathbf{d} = \mathbf{0}$. Now pick an arbitrary direction away from that point, and keep track of that direction with a unit vector $\mathbf{u}$ on the plane. As the positive, real number a increases away from 0, the point

$$\mathbf{y} = \mathbf{x} + a\mathbf{u}$$

moves away from $\mathbf{x}$ in the direction of $\mathbf{u}$. The function D has zero gradient, so to first order it does not change. However, the second derivative may be positive, which means that the window centered at $\mathbf{y}$ is different from the starting window centered at $\mathbf{x}$, and $D(\mathbf{x}, \mathbf{x}+a\mathbf{u})$ increases, at least initially. Measure the rate of increase with one half[1] of the second derivative in the definition (2). Along some directions $\mathbf{u}$, the second derivative at $a = 0$ is smaller than along others. Pick the direction $\mathbf{u}$ that gives the smallest increase. One half of the second derivative in that direction is the distinctiveness of the window at $\mathbf{x}$. At this point, you may want to go back to your plots of $D(\mathbf{x}, \mathbf{x}+\mathbf{d})$ and check your intuition.

You will now prove that $\lambda_{\min}(\mathbf{x})$ is the smaller of the two eigenvalues of the 2×2 matrix

$$A(\mathbf{x}) = \sum_{\mathbf{z} \in \mathbb{Z}^2} w(\mathbf{z}) \, \nabla I(\mathbf{z}+\mathbf{x}) [\nabla I(\mathbf{z}+\mathbf{x})]^T \tag{3}$$

where

$$\nabla I(\mathbf{x}) = \begin{bmatrix} \frac{\partial I}{\partial x_1} \\ \frac{\partial I}{\partial x_2} \end{bmatrix}$$

is the gradient of the image I at $\mathbf{x} = (x_1, x_2)^T$. I will take you through the proof, so do not worry: All you need to do is to fill the blanks.

The expression (1) for D contains the argument $\mathbf{z}+\mathbf{y}$, which for $\mathbf{y} = \mathbf{x}+\mathbf{d}$ becomes $\mathbf{z}+\mathbf{x}+\mathbf{d}$. We start by approximating the image at $\mathbf{z}+\mathbf{x}+\mathbf{d}$ by its Taylor series around $\mathbf{z}+\mathbf{x}$, truncated to the linear term in $\mathbf{d}$:

$$I(\mathbf{z}+\mathbf{x}+\mathbf{d}) \approx I(\mathbf{z}+\mathbf{x}) + [\nabla I(\mathbf{z}+\mathbf{x})]^T \mathbf{d} \; .$$

In this way, $D(\mathbf{x}, \mathbf{x}+\mathbf{d})$ is approximately a quadratic function of $\mathbf{d}$ for small values of $\mathbf{d}$:

$$D(\mathbf{x}, \mathbf{x}+\mathbf{d}) \approx Q(\mathbf{x}, \mathbf{x}+\mathbf{d}) = \sum_{\mathbf{z} \in \mathbb{Z}^2} w(\mathbf{z}) \left\{ I(\mathbf{z}+\mathbf{x}) + [\nabla I(\mathbf{z}+\mathbf{x})]^T \mathbf{d} - I(\mathbf{z}+\mathbf{x}) \right\}^2 = \sum_{\mathbf{z} \in \mathbb{Z}^2} w(\mathbf{z}) \left\{ [\nabla I(\mathbf{z}+\mathbf{x})]^T \mathbf{d} \right\}^2 \; .$$

(a) Compute one half of the partial derivatives of $Q(\mathbf{x}, \mathbf{x}+\mathbf{d})$ with respect to $\mathbf{d}$. Since $\mathbf{d}$ has two components (call them d_1 and d_2), your result will be a vector with two derivatives:

$$\mathbf{q}(\mathbf{x}, \mathbf{d}) \;\; = \;\; \frac{1}{2} \frac{\partial Q(\mathbf{x}, \mathbf{x}+\mathbf{d})}{\partial \mathbf{d}} \;\; = \;\; \frac{1}{2} \begin{bmatrix} \frac{\partial Q(\mathbf{x}, \mathbf{x}+\mathbf{d})}{\partial d_1} \\ \frac{\partial Q(\mathbf{x}, \mathbf{x}+\mathbf{d})}{\partial d_2} \end{bmatrix}$$

[Hint: If you are familiar with differential calculus applied to vectors, this is a one-line derivation. If not, just compute the two derivatives separately, and then package the result into a 2×1 vector.]

(b) Find a matrix $A(\mathbf{x})$ such that your previous answer has the form

$$\mathbf{q}(\mathbf{x}, \mathbf{d}) = A(\mathbf{x}) \, \mathbf{d} \; .$$

Show all your steps clearly. [Hints: (1) Depending on how you wrote the expression for $\mathbf{q}(\mathbf{x}, \mathbf{d})$, it may be useful to use the fact that if s is a scalar and $\mathbf{f}$ is a vector, then $s\mathbf{f} = \mathbf{f}s$. (2) The matrix in this question is not called $A(\mathbf{x})$ by coincidence.]

[1] The factor $1/2$ is introduced merely for convenience in later calculations.

(c) The *Hessian* of $Q(\mathbf{x}, \mathbf{x}+\mathbf{d})$ is the 2×2 matrix of second derivatives of Q with respect to $\mathbf{d}$:

$$H = \frac{\partial^2 Q(\mathbf{x}, \mathbf{x}+\mathbf{d})}{\partial \mathbf{d} \partial \mathbf{d}^T} = \begin{bmatrix} \frac{\partial^2 Q(\mathbf{x},\mathbf{x}+\mathbf{d})}{\partial d_1^2} & \frac{\partial^2 Q(\mathbf{x},\mathbf{x}+\mathbf{d})}{\partial d_1 \partial d_2} \\ \frac{\partial^2 Q(\mathbf{x},\mathbf{x}+\mathbf{d})}{\partial d_2 \partial d_1} & \frac{\partial^2 Q(\mathbf{x},\mathbf{x}+\mathbf{d})}{\partial d_1^2} \end{bmatrix}$$

and can be found by differentiating the two entries of $\mathbf{q}(\mathbf{x}, \mathbf{d})$ each with respect to d_1 and d_2. That is, if the two entries of $\mathbf{q}$ are q_1 and q_2, we have

$$\frac{1}{2}H = \begin{bmatrix} \frac{\partial q_1}{\partial d_1} & \frac{\partial q_1}{\partial d_2} \\ \frac{\partial q_2}{\partial d_1} & \frac{\partial q_2}{\partial d_2} \end{bmatrix} .$$

Use your answer to the previous question to show that

$$\frac{1}{2}H = A(\mathbf{x}) .$$

[Hint: Again, if you know how to differentiate with respect to vectors, this is an instant derivation. If not, just spell out the product $A(\mathbf{x})\mathbf{d}$ in terms of the entries of $A(\mathbf{x})$ and $\mathbf{d}$ and compute the four derivatives separately.]

Since $D \approx Q$, the Hessian of D is approximately $A(\mathbf{x})$. It is not too hard to show that one half of the second derivative in equation (2) is then

$$\frac{1}{2} \left. \frac{d^2 D(\mathbf{x}, \mathbf{x}+a\mathbf{u})}{da^2} \right|_{a=0} \approx \mathbf{u}^T A(\mathbf{x}) \mathbf{u}$$

(the equality would be exact if the Hessian of D were exactly $A(\mathbf{x})$). Also, the expression $\mathbf{u}^T A(\mathbf{x})\mathbf{u}$ is minimized over all unit vectors $\mathbf{u}$ when

$$\mathbf{u} = \mathbf{v}_{\min} ,$$

the eigenvector of $A(\mathbf{x})$ that corresponds to the smaller ($\lambda_{\min}(\mathbf{x})$) of its two real eigenvalues, and

$$\mathbf{v}_{\min}^T A(\mathbf{x}) \mathbf{v}_{\min} = \lambda_{\min}(\mathbf{x}) .$$

You need not prove any of these statements. Together, they show that the distinctiveness $\lambda_{\min}(\mathbf{x})$ of a window at $\mathbf{x}$ is the smaller eigenvalue of the matrix $A(\mathbf{x})$ defined in equation (3). This matrix, which is only approximately the Hessian of the dissimilarity D, is called the *Gramian* of D weighted by $w(\mathbf{z})$. You may recognize this expression as the *image structure tensor* defined in the textbook in section 13.2.2. The selection criterion in equation (13.15) of the book is slightly different from $\lambda_{\min}(\mathbf{x})$, but the idea is very similar.

We can now compute $\lambda_{\min}(\mathbf{x})$ everywhere in an image, and say that a window is *interesting* (or *distinctive*) if $\lambda_{\min}(\mathbf{x})$ is greater than some predefined threshold τ.

3. The function `grad` provided with this assignment computes the gradient of an image. If you say

```
g = grad(I);
```

then `g` is a cell array with two entries: `g{1}` is an image that contains the component of the gradient in the vertical direction at every pixel and `g{2}` contains the component in the horizontal direction. The function `grad` also takes an optional argument for the parameter σ of the kernel, but we will use the default value in this exercise.

(a) Use the function `grad` and the MATLAB builtin function `conv2` with the `'same'` option to write a function with header

```
function lambdaMin = smallEigenvalue(I, sigma)
```

that computes an image with the smallest eigenvalue $\lambda_{\min}(\mathbf{x})$ of the Gramian $A(\mathbf{x})$ defined in equation (3) at every pixel in image `I`. The second argument `sigma` is the parameter σ used in the definition of $w(\mathbf{z})$. There is no need to cast `I` to `double`, because `grad` already does that for its own computations. Use your function `gauss` to compute the weighting mask `w1` such that `w = w1 * w1'`.

If you are surprised that `conv2` shows up here, note that expression (3) for $A(\mathbf{x})$ is a convolution. Specifically, the product

$$P(\mathbf{x}) = \nabla I(\mathbf{x}) [\nabla I(\mathbf{x})]^T = \begin{bmatrix} a(\mathbf{x}) & d(\mathbf{x}) \\ d(\mathbf{x}) & b(\mathbf{x}) \end{bmatrix}$$

is a 2×2 symmetric matrix. So your code will first call `grad` to compute the two components of ∇I everywhere in the image, then use matrix operations to compute the three images $a(\mathbf{x})$, $b(\mathbf{x})$, $d(\mathbf{x})$ in $P(\mathbf{x})$. The three distinct entries of

$$A(\mathbf{x}) = \begin{bmatrix} p(\mathbf{x}) & r(\mathbf{x}) \\ r(\mathbf{x}) & q(\mathbf{x}) \end{bmatrix}$$

are the convolutions of the three distinct entries of $P(\mathbf{x})$ with the Gaussian kernel w. **Make sure you use the separability of the Gaussian to make these convolutions efficient.**

Finally, you can compute the image `lambdaMin` by recalling that the smaller eigenvalue of $A(\mathbf{x})$ is

$$\lambda_{\min}(\mathbf{x}) = p(\mathbf{x}) + q(\mathbf{x}) - \sqrt{[p(\mathbf{x}) - q(\mathbf{x})]^2 + 4r^2(\mathbf{x})}\ .$$

Use matrix operations to do this computation at once over the entire image, without explicit `for` loops.

Hand in your code and the image `lambdaMin` obtained on the image `shadow.jpg` provided with this assignment. To save ink, invert the image so that large values are mapped to black and small values to white. You can do this as follows:

```
clf
imagesc(lambdaMin)
colormap gray
cmap = 1 - colormap;
colormap(cmap);
axis image
axis off
```

(b) Comment on your image `lambdaMin`. In particular, what do interesting windows in `shadow.jpg` look like?