COMPSCI 527 — Homework 1

Due on September 10, 2015

Work on this assignment either alone or in pairs. You may work with different partners on different assignments, but you can only have up to one partner for any one assignment. You may not talk about this assignment with others until all of you have handed in their work. See Mechanics→Homework on the class web page for details on the homework policy.

Read and comply with the following instructions carefully. We will take points off for not doing so.

This assignment comes with a LaTeX template file `template.tex` that helps you get started. Before you do anything for this assignment, rename the file `template.tex` to `first_last_hw1.tex` where `first` and `last` are your first and last name. If either name has spaces or special characters (such as apostrophes or diacritical marks), please skip those. Please do not use capital letters, and skip any middle names. So for instance Zoë Linda O'Connor will have file `zoe_oconnor_hw1.tex`. If you work with a partner, use a similar format, but with the pattern `first1_last_1_first2_last2_hw1.tex` where your names are sorted alphabetically by last name (or by first name if the two last names are identical).

Open the resulting file you made in LaTeX, change `XXXYourName(s)` with your name or names, and compile the file—that is, run LaTeX on it to produce a `.pdf` file.

The `.tex` file has problem headers and `\newpage` statements that place page breaks between problems. Please do not change headers or page breaks. This will help us read your work more easily. You may add page breaks if this is useful.

The template file also contains some text, formulas, matrices, and code to show you how those are written in case you are not familiar with LaTeX. Please remove all this extra LaTeX input from your file (you have a copy of it in `template.tex` anyway), so we don't have to read our own code and text. Instructions are given in the `.tex` file itself for two different ways to remove the extra input.

Hand in only the PDF file that results from compiling your `.tex` file. Make sure the output is formatted correctly and that all lines of text and code show up in the PDF file.

Some questions require writing simple MATLAB code. Please intersperse concise comments with your code whenever useful, but do not go overboard with comments. Try to use matrix notation whenever possible, so as to avoid explicit `for` loops over images. These are inefficient in MATLAB.

1. The MATLAB statement

```
F = conv2(A, H, 'full');
```

computes the convolution of array `A` and kernel `H` for all relative displacements for which the two arrays overlap at all. More specifically, the result of this call is equivalent[1] to the following operations. If `H` has size $r_H \times c_H$ (rows by columns), then pad array `A` with p_W zeros on the left, p_E on the right, p_N above, and p_S below, where

$$p_W = p_E = c_H - 1 \quad , \quad p_N = p_S = r_H - 1$$

to produce a new image `B`. So if `A` has size $r_A \times c_A$, then `B` has size

$$r_B \times c_B \quad \text{where} \quad r_B = r_A + 2(r_H - 1) \quad \text{and} \quad c_B = c_A + 2(c_H - 1) \; .$$

The result of `conv2` is then the convolution of `B` and `H` computed everywhere these two arrays overlap fully, in the sense that the smaller array is completely contained in the larger one. For instance, if `H` is 2×3, then the first (that is, top-left) entry of `F` contains the value

$$f_{11} = h_{11} a_{11}$$

(remember that convolution entails flipping both rows and columns of the kernel `H`).

(a) Convolution is a linear operation, so there must be a matrix `Mf` that depends on `H` and transforms a vector form of `A` into a vector form of `F`, that is, such that the predicate

```
all(F(:) == Mf * A(:))
```

is true (the MATLAB colon operator arranges the entries of a matrix as a column vector in column-major order).

What is the size of `Mf`?

(b) Write out the matrix `Mf` for the case $r_A = 3$, $c_A = 4$, and for

$$\mathtt{H} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix} \; .$$

The entries in `H` correspond to the order in which they are listed in `H(:)`. *Please leave all zero entries blank in your answer.* [Hint: Be patient and make clean drawings for yourself. Do *not* hand in your drawings.]

[1]This is not a description of the actual code.

(c) The MATLAB statement

```
S = conv2(A, H, 'same');
```

computes a smaller convolution than the one obtained with the `'full'` option. Specifically, `S` has the same size as `A`, and padding to produce the image `B` from `A` is as follows:

$$p_W = \left\lfloor \frac{c_H - 1}{2} \right\rfloor \quad , \quad p_N = \left\lfloor \frac{r_H - 1}{2} \right\rfloor \quad , \quad p_E = c_H - p_W - 1 \quad , \quad p_S = r_H - p_N - 1 \; .$$

The result of `conv2` with the `'same'` option is then the convolution of `B` and `H` computed everywhere these two arrays overlap fully. For instance, if `H` is 2×3, then the first entry of `S` contains the value

$$s_{11} = h_{22}a_{11} + h_{21}a_{12} + h_{12}a_{21} + h_{11}a_{22} \; .$$

Just as before, there must be a matrix `Ms` that depends on `H` and such that

```
all(S(:) == Ms * A(:))
```

What is the size of `Ms`?

(d) `Ms` is the same as `Mf` but with some rows erased. Write out the matrix `Ms` for the case $r_A = 3$, $c_A = 4$, and for

$$\mathtt{H} = \left[\begin{array}{ccc} 1 & 3 & 5 \\ 2 & 4 & 6 \end{array}\right] .$$

Please leave all zero entries blank in your answer.

(e) Write a MATLAB function with header

```
function M = convMatrix(H, sA, type)
```

where `sA` is the size of `A` (as obtained with `size(A)`) that computes `Mf` if `type` is `'full'` and `Ms` if `type` is `'same'`. Your code should not make any assumptions about the sizes of `A` or `H`, other than these are non-empty, 2-dimensional arrays.

Several valid ways to write this code are possible, depending on what patterns you see in the examples you built for your answers to the questions above. It is also possible to write this code without regard to the discussion above. More succinct (and correct) code will get higher grades. Include your code into your PDF file within a `verbatim` environment or by using the `\VerbatimInput` command of the `fancyvrb` package, as shown in the template `.tex` file.

(f) Show the six matrices $M\{1\}, \ldots, M\{6\}$ you obtain with the following code snippet (also available in the template `.tex` file):

```
C = [1 1; 1 2; 2 1; 2 2; 3 2; 3 3];
M = cell(size(C, 1), 1);
for k = 1:size(C, 1)
    M{k} = convMatrix(reshape(1:prod(C(k, :)), C(k, :)), [2 3], 'same');
    latexArray(M{k}, sprintf('M\\{%d\\}', k), sprintf('../M%d.tex', k), '%d');
end
```

Please leave zero entries blank in your answer, and write out two matrices per line. [Hint: It may be faster and less error prone to write MATLAB code that produces the LaTeX source for these matrices, rather than writing the matrices by hand. A sample function `latexArray` that does this is provided with the assignment. Feel free to customize that if you like.]

(g) Write a single MATLAB instruction that takes an array (image) `A` and the matrix `M` resulting from the call

```
M = convMatrix(H, size(A), 'same');
```

and produces the same result as

```
S = conv2(A, H, 'same');
```

without using `conv2`. The MATLAB function `reshape` is useful here.

(h) Check your previous answer using as inputs the matrices

```
H = reshape(1:6, [2 3]);
A = reshape(1:12, [3 4]);
```

Specifically, let

```
S = conv2(A, H, 'same');
S2 = <your instruction here>
```

and show the values of `S` and `S2`.

2. The Sobel operator computes an approximate image derivative in the horizontal direction. Its kernel is

$$K = \frac{1}{8}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} .$$

(a) Show that the Sobel operator is separable. Allocate any constants in the most meaningful way to the two factors K_1 and K_2 that K separates into.

(b) State accurately what the two factors do, that is, what their effect is as each of them is convolved with an image.

(c) Write out the matrices $4M_{K_1}$ and $2M_{K_2}$ where M_{K_1} and M_{K_2} are the two matrices that correspond to K_1 and K_2 (as per problem 1) for a 3×3 input image and the `'same'` option:

```
MK1 = convMatrix(K1, [3 3], 'same');
MK2 = convMatrix(K2, [3 3], 'same');
```

Please leave all zero entries blank.

(d) If M_K is the matrix for the Sobel kernel K, how can you compute M_K from M_{K_1} and M_{K_2}? Just give a formula.

(e) Knowing what you know about convolution, the square matrices M_s that arise from convolution as shown in problem 1 must—quite surprisingly—enjoy a special property that is not generally shared by square matrices. What is that property?

3. For this problem, assume that images are square, and that image size is a power of 2 in either dimension. A simpler way to compute an image pyramid than using Gaussian filters is to use a 2×2 box filter with kernel

$$B = \frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

and subsample by a factor of 2 (that is, take every other pixel in the row and the column direction). So the pyramid definition is

$$\begin{aligned} P_0 &= I \\ P_\ell &= \text{down}(P_{\ell-1}) \quad \text{for} \quad \ell = 1, \ldots, L \end{aligned}$$

where I is the input image and $Z = \text{down}(X)$ computes

```
Y = conv2(X, B, 'same');
Z = Y(2:2:end, 2:2:end);
```

Let us call this a *block pyramid*. The code above is inefficient, because it first computes all the pixels in `Y` and then throws away 3/4 of them when subsampling.

(a) Write an efficient version of a function with header

```
function p = blockPyramid(img)
```

that takes a black-and-white image of proper size and type `double` and returns a cell array `p`. The entries of the cell array are the images in the full pyramid for `img`, with `p{1}` being the input image itself.

Your function should be efficient in that it should not compute values it then throws away, nor should it explicitly loop over the pixels in the image (use array operations instead). Your function should also check that the input image is black-and-white (that is, it is a two-dimensional array), square, and has a power of 2 as each of its dimensions. Show your code.

(b) The function `compositePyramid` provided with this assignment takes a pyramid as input and arranges its levels into a single composite image for display purposes. It rescales each pyramid level independently from the others for better visibility. Use this function and the MATLAB function `imwrite` to make an image file `p.png` with the composite image for the block pyramid of the image `wave.png` provided with this assignment. Read the image as

```
img = double(imread('wave.png'));
```

to convert it to double floating-point format. Use the LaTeXfunction `\includegraphics` from the `graphicx` package to include the `p.png` figure in your PDF file. Do *not* include `wave.png`.

(c) Write a function with header

```
function gp = gMag(p)
```

that takes a pyramid `p` and uses the Sobel filter to compute a new pyramid `dp` whose levels are the gradient magnitudes of the levels of `p`.

Use the MATLAB function `imfilter` with the `'replicate'` option instead of using `conv2`. Exploit the separability of the Sobel filter for greater efficiency. Show your code.

[Note: We saw in class that Gaussian filters and their derivatives are the “proper” convolution kernels for differentiation. However, Gaussians are also the better way for building pyramids. In this exercise, we build a pyramid with box filters, so using Gaussians for differentiation would be overkill.]

(d) Show the pyramid `gMag(p)` on the pyramid `p` you built above, using the same display technique.