JHU 553.493/693, Spring '20 – Homework #1

Point operations, filters, and partial differential operators.

Due on Blackboard by the end of the day on Sunday, February 16

Reading: While the class notes should be sufficient for this homework assignment, it is still useful to read Chapters 1 to 6 of *Principles of Digital Image Processing* by Burger & Burge (on Blackboard). You may skip all the parts that refer to Java implementations and ImageJ (for example, all of Chapter 2), unless they are of interest to you. The most important chapters are, for our purposes, Chapter 3 to Chapter 6.

Write, on top of the first page of your assignment: Name (<u>LAST</u>, First), the HW# (in this case, "Homework #1"), and acknowledge others with whom you may have worked (just write "*Worked with ...*"). For the computational problems where you are asked to write computer code you may choose the programming language that you prefer, such as *Python* or *Matlab*. You do *not* have to type up your homework. If you handwrite it, please scan it making sure it is readable. You should submit *one* PDF file on Blackboard (or at most two, if you choose to submit your code separately). Feel free to use in-built functions for convolution (e.g. `conv2` in *Matlab*): you do *not* have to implement it from scratch using nested '`for`' loops.

Problem 1.1. Suppose that an image has all its pixel values clustered in a small interval—say, in $[a,b] \subset [0, L-1]$, with $0 < a < b < L-1$. The image needs to be enhanced, so that the small range $[a,b]$ maps to a the range $[0, L-1]$. What operation $S : [0, L-1] \to [0, L-1]$ would you use for this purpose?

Problem 1.2 (γ-correction)**.** Remember that *gamma correction* is a point operation defined as follows:

$$J(i,j) = (L-1)\Big(\frac{I(i,j)}{L-1}\Big)^{\gamma}, \qquad \text{for } (i,j) \in \mathbb{D} = \{0,1,\dots,M\} \times \{0,1,\dots,N\},$$

where $L-1 = 2^b - 1$ is typically 255, and γ is a positive constant chosen by the user. **(a)** Using software of your choice, plot on the same figure the graphs of the function $S_\gamma : [0, L-1] \to [0, L-1]$, where $s = S_\gamma(r) = (L-1)[r/(L-1)]^\gamma$, for the following values of γ: 0.04, 0.1, 0.2, 0.4, 0.61, 1, 1.5, 2.5, 5, 10, and 25. **(b)** Fix an arbitrary value of γ: what is the *inverse* transformation S_γ^{-1}, i.e. the function such that $S_\gamma^{-1}(S(r)) = r$? **(c)** Consider now the images `image-spine.tif` and `image-airport.tif`. The first looks too dark, while the other looks too bright. Write computer code[1] to adjust the brightness of the images by applying gamma corrections with appropriate values of the parameter γ (remember that the output values must be rounded off so to be in the set $\mathbb{P} = \{0, 1, 2, \dots, L-1\}$). Show your results, and also plot the *histograms* of the original images as well as the transformed images. Provide your code.

Problem 1.3. Apply a 3×3 averaging filter to the image `image-lena-noisy.png`, which is the classic Lena[2] image with added Gaussian noise. Also try a 5×5 uniform filter, and a non-uniform averaging 5×5 filter ('Gaussian mask'). In general, what do you observe in your result if you apply larger filter masks?

Problem 1.4. The median filter (see the class notes) is the most suitable to deal with *salt-and-pepper* noise. **(a)** Show that a median filter is *nonlinear*. **(b)** Write computer code to apply a 3×3 median filter to `image-circuit.jpg`, which is an x-ray image of a circuit board corrupted by salt-and-pepper noise. **(c)** Now apply a 3×3 linear average filter to `image-circuit.jpg`. Why is the result so different?

Problem 1.5. Show that convolution is an *associative* operation, i.e. that for three arbitrary functions $f, g, h : \mathbb{Z}^2 \to \mathbb{R}$ it is the case that $f * (g * h) = (f * g) * h$.

Problem 1.6 (3-point midpoint formula for the second derivative)**.** Remember from class (and Numerical Analysis, if you have taken that course) the so-called *forward difference* and *backward difference* formulas:

[1] For implementation purposes, note that we can write $a^\gamma = e^{\gamma \ln a}$.

[2] *Lena* is the name given to a standard test image widely used in the field of image processing since 1973. It is in fact a photo of Lena Söderberg, shot by photographer Dwight Hooker, cropped from the centerfold of the November 1972 issue of Playboy magazine (!). She was a guest at the 50th annual Conference of the Society for Imaging Science and Technology in 1997.

$$f'(x) \simeq \frac{f(x+\Delta x)-f(x)}{\Delta x} \qquad \text{and} \qquad f'(x) \simeq \frac{f(x)-f(x-\Delta x)}{\Delta x}$$

for numerical differentiation. Combine the two to find the 3-point midpoint formula for the numerical computation of the second derivative of a function, namely $f''(x) \simeq (f(x+\Delta x)-2f(x)+f(x-\Delta x))/(\Delta x)^2$.

Problem 1.7. The result of Problem 1.5 shows that if we want to apply two filters (with convolution kernels g and h, respectively) to a given image I one after the other, it is the case that $(I*g)*h = I*(g*h)$. In other words, we can combine the two convolution kernels into one, $f = g*h$, and then filter the image with the *new* kernel by computing $I*f$. In class we introduced several kernels for computing $\frac{\partial I}{\partial x}$, namely:

$$h_x = \begin{bmatrix} h_x(-1,-1) & h_x(-1,0) & h_x(-1,1) \\ h_x(0,-1) & h_x(0,0) & h_x(0,1) \\ h_x(1,-1) & h_x(1,0) & h_x(1,1) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix},$$

$$h_x^P = \frac{1}{6}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, \qquad h_x^S = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}.$$

The first one is derived from the 3-point midpoint formula for numerical differentiation, the second one is the *Prewitt* filter (which is obtained from h_x by averaging over 3 horizontal pixels, with equal weights), while the last one is the *Sobel* filter (which is obtained from h_x by averaging over 3 horizontal pixels, with weights 1/4, 1/2, and 1/4). We also introduced the following filter for the *second* partial derivative $\frac{\partial^2}{\partial x^2}$:

$$h_{xx} = \begin{bmatrix} h_{xx}(-1,-1) & h_{xx}(-1,0) & h_{xx}(-1,1) \\ h_{xx}(0,-1) & h_{xx}(0,0) & h_{xx}(0,1) \\ h_{xx}(1,-1) & h_{xx}(1,0) & h_{xx}(1,1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

for which we used the 3-point midpoint formula for the second derivative, which you found in Problem 1.6. However, since we can compute a second partial derivative by performing a single partial derivative *twice* in the same direction, the above reasoning suggests that we may approximate $\partial^2 I/\partial x^2$ with $I*f$: here, $f = h*g$ where h and g may be chosen independently as either h_x, h_x^P, or h_x^S. In how many ways can this be done? Also, what are the effects of choosing $f = h_x * h_x$ rather than, say, $f = h_x * h_x^P$? Compute, either by hand or using some computer code, the convolution kernel $f = h_x * h_x^S$. What is its size?

Problem 1.8 (Mixed partials)**.** Suggest a convolution kernel h_{xy} such that $\frac{\partial^2 I}{\partial x \partial y} \simeq I * h_{xy}$.
Hint: There are multiple correct answers to this question.

Problem 1.9. (Rotation invariance of the Laplacian operator.) Show that the Laplacian operation $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$ is *isotropic* (i.e. invariant under rotations, or rotationally invariant). That is, we choose an arbitrary angle θ and operate the change of coordinates $T : (x,y) \mapsto (u,v)$ given by the transformation:

$$T : \begin{cases} x(u,v) &= u\cos\theta - v\sin\theta \\ y(u,v) &= u\sin\theta + v\cos\theta \end{cases} \qquad \Longleftrightarrow \qquad T^{-1} : \begin{cases} u(x,y) &= x\cos\theta + y\sin\theta \\ v(x,y) &= -x\sin\theta + y\cos\theta \end{cases}$$

where (x,y) are the unrotated and (u,v) are the rotated coordinates. Define the composite function $F(u,v) = f(x(u,v), y(u,v))$, which is nothing but f expressed in the new coodrinates, and show that $\frac{\partial^2 F}{\partial u^2}(u,v) + \frac{\partial^2 F}{\partial v^2}(u,v) = \frac{\partial^2 f}{\partial x^2}(x(u,v),y(u,v)) + \frac{\partial^2 f}{\partial y^2}(x(u,v),y(u,v))$. Equivalently, since we have $f(x,y) = F(u(x,y), v(x,y))$, you can show that $\frac{\partial^2 f}{\partial x^2}(x,y) + \frac{\partial^2 f}{\partial v^2}(x,y) = \frac{\partial^2 F}{\partial u^2}(u(x,y),v(x,y)) + \frac{\partial^2 F}{\partial v^2}(u(x,y),v(x,y))$.
Note: Isotropy is important if you want an operation to be rotation-invariant: that is, if you apply the same operation to an image that has been rotated by a certain angle, you want it to yield the same result as the operation applied to the non-rotated image. This is significant, for example, in edge detection.

Problem 1.10 (A basic sharpening filter.)**.** Write a computer program that implements the operation $J(x,y) = I - \nabla^2 I$, in the form $J = I * h^{\mathrm{Sh}}$ with a 3×3 sharpening convolution kernel h^{Sh}. Give the form of the convolution kernel, and apply your script to the image of the North Pole of the moon (`image-moon.jpg`). You should turn in the the computer code, the input and output images, as well as a short explanation of why the method works. Make sure that your code handles saturation correctly.