ECEN 642, Fall 2019

Texas A&M University

Electrical and Computer Engineering Department

Dr. Raffaella Righetti

Due: 10/11/2019 (before class)

Assignment #3

Gonzalez and Woods (4th edition), projects: 3.5, 3.6, 3.7, 3.8, 3.9, 3.10

Gonzalez and Woods (4th edition), problems: 3.33, 3.44, 3.48

For 3.5, please DON'T use MATLAB command "conv2". Develop it from scratch instead.

Please clearly indicate your processing parameters (if there is any) in your solutions

For the assignments, you will need to use MATLAB. You can access it on campus through the open access lab (OAL) or remotely through the virtual open access lab (VOAL). The link below will guide you into configuring the Horizon client for VOAL remote connection step by step on your PC or mac:

https://tamu.service-now.com/tamu-selfservice/knowledge_detail.do?sysparm_document_key=kb_knowledge,6f7e0c6adbce5f84778ff5961d96199f#

After you connect to the server, you will see the MATLAB icon. If you don't, you can connect to your VOAL desktop (VOAL icon) and start MATLAB from there.

For projects from the 4th edition, photo copies of the project statements are provided. For projects from the 3rd edition, you can access them via the link: http://www.imageprocessingplace.com/DIP-3E/dip3e_student_projects.htm#02-04

(c) See if you can improve on the result in (b) by using a gamma transformation. Display your best result.

(d) If an improvement is possible (or not possible), explain why.

3.3 Histogram equalization.

(a) Write a function **g = histEqual4e(f)** for performing histogram equalization on 8-bit input image **f**. (*Hint:* Use functions **imageHist4e** from Chapter 2, and **intXform4e** from Project 3.2 as part of function **histEqual4e**).

(b) Histogram equalize the image **spillway-dark.tif** and compare the result with the result in Project 3.2(c).

(c) Histogram equalize the image **hidden-horse.tif** and compare your result with Fig. 3.25(c).

3.4 Local histogram equalization.

(a)* Use functions **imageHist4e** from Chapter 2 and **intXform4e** from Projects 3.2 as the basis for writing a function **g = localHistEqual4e(f,m,n)** for performing local histogram equalization of 8-bit grayscale image **f** based on a neighborhood of size $m \times n$. Use replicate padding to pad the image border.

(b) Apply your function to the image **hidden-symbols.tif** using a 3×3 neighborhood, display the result, and compare it with Fig. 3.32(c). (The project image is of lower resolution than the image in the book, so your details will not be as sharp. However, the general characteristics of the two images should be the same.)

(c) Repeat (b) using a 7×7 neighborhood and explain any significant visual differences between the two results.

3.5 Two-dimensional convolution.

(a)* Write or obtain a function **g = twodConv4e(f,w)** for performing 2-D convolution of image **f** and kernel **w** in the language you are using for your projects. This function should use replicate padding by default. If you are using MATLAB, use function **conv2** as a starting point. This function uses zero-padding by default but you can get around this by pre-padding **f** with replicate padding and then stripping out the excess rows and columns due to the extra replicate padding before outputting **g**. Function **conv2** requires floating point inputs. Your function should by default scale the input to the range [0, 1] using the default mode in project function **intScaling4e**. However, the function also has to have the capability of disabling this automatic scaling.

(b) Create an image of size 512×512 pixels that consists of a unit impulse at location (256, 256) and zeros elsewhere. Use this image and a kernel of your choice to confirm that your function is indeed performing convolution. Display your results and explain what you did and why.

3.6 Lowpass filter kernel.

(a) Write a function **w = gaussKernel4e(m,sig,K)** that uses Eq. (3-55) to generate a normalized Gaussian lowpass kernel of size $m \times m$. If **K** is not included in the function call it should default to 1.

(b)* Filter the image from Project 3.5 with a Gaussian lowpass kernel large enough so that the maximum value of the filtered image is approximately 0.005 of the maximum value of the original image. Display the filtered image. (*Hint:* The image will appear black unless you scale it. Use the **'full'** mode in project function **intScaling4e** to scale the intensity values.)

3.7 Lowpass filtering.

(a) Read the image **testpattern1024.tif** and lowpass filter it using a Gaussian kernel large enough to blur the image so that the large letter "a" is barely readable, and the other letters are not.

(b)* Read the image **testpattern1024.tif**. Lowpass filter it using a Gaussian kernel of your specification so that, when thresholded, the filtered image contains only part of the large square on the top, right. (*Hint:* It is more intuitive to work with the negative of the original image.)

(c) Read the image **checkerboard1024-shaded.tif** and reproduce the results in Example 3.18, keeping in mind that the above image is of size 1024×1024 pixels, so the checkerboard squares are 64×64 pixels. (*Hint:* to obtain images like the ones in the example, scale the shading pattern and the processed image to the full [0, 1] intensity range—you can use project function **intScaling4e** for this.)

3.8 Unsharp masking and highboost filtering.

(a)* Read the image **blurry-moon.tif** and sharpen it using unsharp masking. Use a Gaussian lowpass kernel of your choice for the blurring step. Display your final result.

(b) Improve the sharpness of your result using highboost filtering. Display the final result.

3.9 Highpass filtering.

(a)* Read the image **blurry-moon.tif** and sharpen it using the Laplacian kernel in Fig. 3.51(c). Compare your result with the result in Project 3.8(a), and discuss any differences and similarities between the two images.

(b) Read the image **chekerboard1024-shaded.tif** and filter it with the Sobel kernel in Fig. 3.56(d). Then, filter the original image again using the kernel in Fig. 3.56(e). Display your results and explain why they look the way they do.

(c) Filter the checkerboard image with one of the Sobel kernels and then filter the result with the other kernel. Display your result and explain why it looks the way it does.

(d) Obtain the magnitude of the gradient [Eq. (3-74)] of the checkerboard image using the Sobel kernels. Display and explain the result.

3.10 Using kernels generated by filter-design software.

(a)* File **lpkernel1D.txt** is a comma-delimited text file containing a 129-coefficient, one-dimensional filter kernel similar to the kernel shown in Fig. 3.60(a). This kernel was generated using MATLAB's Signal Processing Toolbox. Read this file and plot it.

(b)* Construct a 2-D lowpass kernel from the data in (a) and filter the image **testpattern1024.tif** with it. Display your result.

(c) Create a 2-D highpass filter kernel from the lowpass kernel in (b) and filter the image **testpattern1024.tif** with it. Display your result.

(b) Because G is separable and circularly symmetric, it can be expressed in the form $G = \mathbf{v}\mathbf{v}^T$. Assume that the kernel form in Eq. (3-55) is used, and that the function is sampled to yield an $m \times m$ kernel. What is $\mathbf{v}$ in this case?

3.32* Show that the product of a column vector with a row vector is equivalent to the 2-D convolution of the two vectors. The vectors do not have to be of the same length. You may use a graphical approach (as in Fig. 3.36) to support the explanation of your proof.

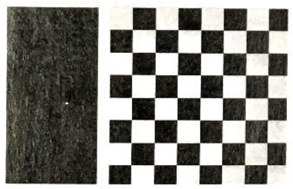
3.33 Given K, 1-D Gaussian kernels, $g_1, g_2, \ldots, g_K$, with arbitrary means and standard deviations:

(a)* Determine what the entries in the third column of Table 3.6 would be for the product $g_1 \times g_2 \times \cdots \times g_K$.

(b) What would the fourth column look like for the convolution $g_1 \bigstar g_2 \bigstar \cdots \bigstar g_K$?

(*Hint:* It is easier to work with the variance; the standard deviation is just the square root of your result.)

3.34 The two images shown below are quite different, but their histograms are the same. Suppose that each image is blurred using a 3×3 box kernel.



(a)* Would the histograms of the blurred images still be equal? Explain.

(b) If your answer is no, either sketch the two histograms or give two tables detailing the histogram components.

3.35 An image is filtered four times using a Gaussian kernel of size 3×3 with a standard deviation of 1.0. Because of the associative property of convolution, we know that equivalent results can be obtained using a single Gaussian kernel formed by convolving the individual kernels.

(a)* What is the size of the single Gaussian kernel?

(b) What is its standard deviation?

3.36 An image is filtered with three Gaussian lowpass kernels of sizes 3×3, 5×5, and 7×7, and standard deviations 1.5, 2, and 4, respectively. A composite filter, w, is formed as the convolution of these three filters.

(a)* Is the resulting filter Gaussian? Explain.

(b) What is its standard deviation?

(c) What is its size?

3.37* Discuss the limiting effect of repeatedly filtering an image with a 3×3 lowpass filter kernel. You may ignore border effects.

3.38 In Fig. 3.48(b) the corners of the estimated shading pattern appear darker or lighter than their surrounding areas. Explain the reason for this.

3.39* An image is filtered with a kernel whose coefficients sum to 1. Show that the sum of the pixel values in the original and filtered images is the same.

3.40 An image is filtered with a kernel whose coefficients sum to 0. Show that the sum of the pixel values in the filtered image also is 0.

3.41 Often, you will see in the literature kernels whose elements are powers of two, of the form shown in the three kernels below. At times, these kernels are referred to as Gaussian [implying circularly-symmetric Gaussian, as in Eq. (3-55)]; at other times they are called approximations to Gaussians.

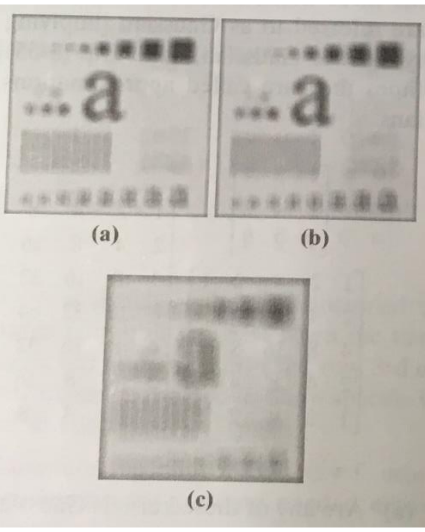
$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 4 & 2 & 1 \\ 2 & 4 & 8 & 4 & 2 \\ 4 & 8 & 16 & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 4 & 8 & 4 & 2 & 1 \\ 2 & 4 & 8 & 16 & 8 & 4 & 2 \\ 4 & 8 & 16 & 32 & 16 & 8 & 4 \\ 8 & 16 & 32 & 64 & 32 & 16 & 8 \\ 4 & 8 & 16 & 32 & 16 & 8 & 4 \\ 2 & 4 & 8 & 16 & 8 & 4 & 2 \\ 1 & 2 & 4 & 8 & 4 & 2 & 1 \end{bmatrix}$$

(a)* Are any of these kernels Gaussian? Explain.

(b) The kernels are separable. Give a general expression for a vector $\mathbf{v}$, such that $\mathbf{v}\mathbf{v}^T$ generates all three kernels and, in general, kernels of size $n \times n$, with integer coefficients that are powers of two.

3.44 In the original image used to generate the three blurred images shown, the vertical bars are 5 pixels wide, 100 pixels high, and their separation is 20 pixels. The image was blurred using square box kernels of sizes 23, 25, and 45 elements on the side, respectively. The vertical bars on the left, lower part of (a) and (c) are blurred, but a clear separation exists between them.



However, the bars have merged in image (b), despite the fact that the kernel used to generate this image is much smaller than the kernel that produced image (c). Explain the reason for this.

3.47 Do the following:

(a)* Develop a procedure for computing the median of an $n \times n$ neighborhood.

(b) Propose a technique for updating the median as the center of the neighborhood is moved from pixel to pixel.

3.48 In a given application, a smoothing kernel is applied to input images to reduce noise, then a Laplacian kernel is applied to enhance fine details. Would the result be the same if the order of these operations is reversed?

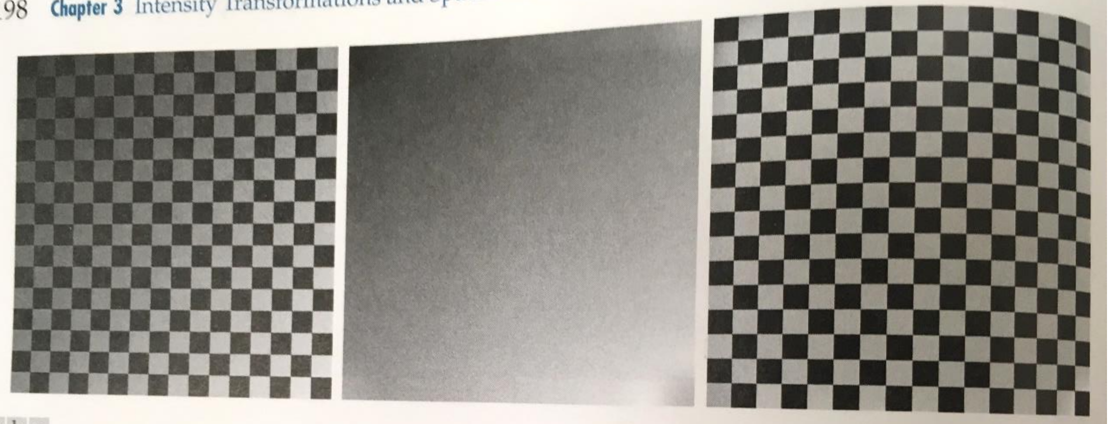
3.49* Show that the Laplacian defined in Eq. (3-59) is isotropic (invariant to rotation). Assume continuous quantities. From Table 2.3, coordinate rotation by an angle θ is given by

$$x' = x\cos\theta - y\sin\theta \quad \text{and} \quad y' = x\sin\theta + y\cos\theta$$

Figures & formulas

EXAMPLE 3.18: Shading correction using lowpass filtering.

One of the principal causes of image shading is nonuniform illumination. *Shading correction* (also called *flat-field correction*) is important because shading is a common cause of erroneous measurements, degraded performance of automated image analysis algorithms, and difficulty of image interpretation by humans. We introduced shading correction in Example 2.7, where we corrected a shaded image by dividing it by the shading pattern. In that example, the shading pattern was given. Often, that is not the



a b c

FIGURE 3.48 (a) Image shaded by a shading pattern oriented in the −45° direction. (b) Estimate of the shading patterns obtained using lowpass filtering. (c) Result of dividing (a) by (b). (See Section 9.8 for a morphological approach to shading correction).

case in practice, and we are faced with having to estimate the pattern directly from available samples of shaded images. Lowpass filtering is a rugged, simple method for estimating shading patterns.

Consider the 2048×2048 checkerboard image in Fig. 3.48(a), whose inner squares are of size 128×128 pixels. Figure 3.48(b) is the result of lowpass filtering the image with a 512×512 Gaussian kernel (four times the size of the squares), $K = 1$, and $\sigma = 128$ (equal to the size of the squares). This kernel is just large enough to blur-out the squares (a kernel three times the size of the squares is too small to blur them out sufficiently). This result is a good approximation to the shading pattern visible in Fig. 3.48(a). Finally, Fig. 3.48(c) is the result of dividing (a) by (b). Although the result is not perfectly flat, it definitely is an improvement over the shaded image.

In the discussion of separable kernels in Section 3.4, we pointed out that the computational advantage of separable kernels can be significant for large kernels. It follows from Eq. (3-53) that the computational advantage of the kernel used in this example (which of course is separable) is 262 to 1. Thinking of computation time, if it took 30 sec to process a set of images similar to Fig. 3.48(b) using the two 1-D separable components of the Gaussian kernel, it would have taken 2.2 hrs to achieve the same result using a nonseparable lowpass kernel, or if we had used the 2-D Gaussian kernel directly, without decomposing it into its separable parts.

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

a b c d

FIGURE 3.51 (a) Laplacian kernel used to implement Eq. (3-62). (b) Kernel used to implement an extension of this equation that includes the diagonal terms. (c) and (d) Two other Laplacian kernels.

a
b c
d e

FIGURE 3.56 (a) A 3×3 region of an image, where the *z*s are intensity values. (b)–(c) Roberts cross-gradient operators. (d)–(e) Sobel operators. All the kernel coefficients sum to zero, as expected of a derivative operator.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0
0	1

0	−1
1	0

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

$$G(r) = Ke^{-\frac{r^2}{2\sigma^2}} \qquad (3\text{-}55)$$

$$M(x,y) = \left[g_x^2 + g_y^2\right]^{\frac{1}{2}} = \Big[\left[(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)\right]^2 + \left[(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)\right]^2\Big]^{\frac{1}{2}} \qquad (3\text{-}74)$$

TABLE 3.6 Mean and standard deviation of the product ($\times$) and convolution ($\star$) of two 1-D Gaussian functions, f and g. These results generalize directly to the product and convolution of more than two 1-D Gaussian functions (see Problem 3.33).

	f	g	$f \times g$	$f \star g$
Mean	m_f	m_g	$m_{f\times g} = \dfrac{m_f\sigma_g^2 + m_g\sigma_f^2}{\sigma_f^2 + \sigma_g^2}$	$m_{f\star g} = m_f + m_g$
Standard deviation	σ_f	σ_g	$\sigma_{f\times g} = \sqrt{\dfrac{\sigma_f^2\sigma_g^2}{\sigma_f^2 + \sigma_g^2}}$	$\sigma_{f\star g} = \sqrt{\sigma_f^2 + \sigma_g^2}$