Digital Image Processing

Lecture 3: Histogram Processing and Fundamentals of Spatial Filtering

Dr. Behnam Kiani
Department of Computer Science

Spring 2025

Readings: Chapter 3 (sections 3.3 and 3.4)
How to read: at least twice, once laid back, once with full focus.

Table of Content

Histogram Processing

- Let $\mathbf{r_k}$, for **k=0,1,…,L-1**, represent the intensities of an **L**-level image **f(x,y)**.
- The ***unnormalized histogram*** of **f** is defined as

$$\boxed{h(r_k) = n_k, \quad \text{for } k = 0, 1, 2, \ldots, L-1}$$

where $\mathbf{n_k}$ is the number of pixels of **f** with intensity $\mathbf{r_k}$.

- The subdivisions of the intensity scale are called **histogram bins**.
- The ***normalized histogram*** of **f** is defined as

$$\boxed{p(r_k) = \frac{h(r_k)}{MN} = \frac{n_k}{MN}}$$

where as usual, **M** and **N** are the number of image rows and columns.

Histogram Processing

- The sum of **p(r$_k$)** for all values of **k** is always 1:

$$\sum_{k=0}^{L-1} p(r_k) = 1$$

- The components of **p(r$_k$)** are the probabilities of intensity levels in the image.
- Histograms are very good indicators of the image contrast:
  - Low-contrast: histogram with narrow intensity range;
  - High-contrast: histogram with broad intensity range

Histogram Processing

- How to enhance the contrast in an image? Since a more even distribution in the histogram can be linked to higher contrast, it is natural to assume if a histogram is expanded to cover the full range of intensity values, the contrast is improved.
- In an ideal image, every intensity level corresponds to the same number of pixels. Histogram equalization aims to do this.

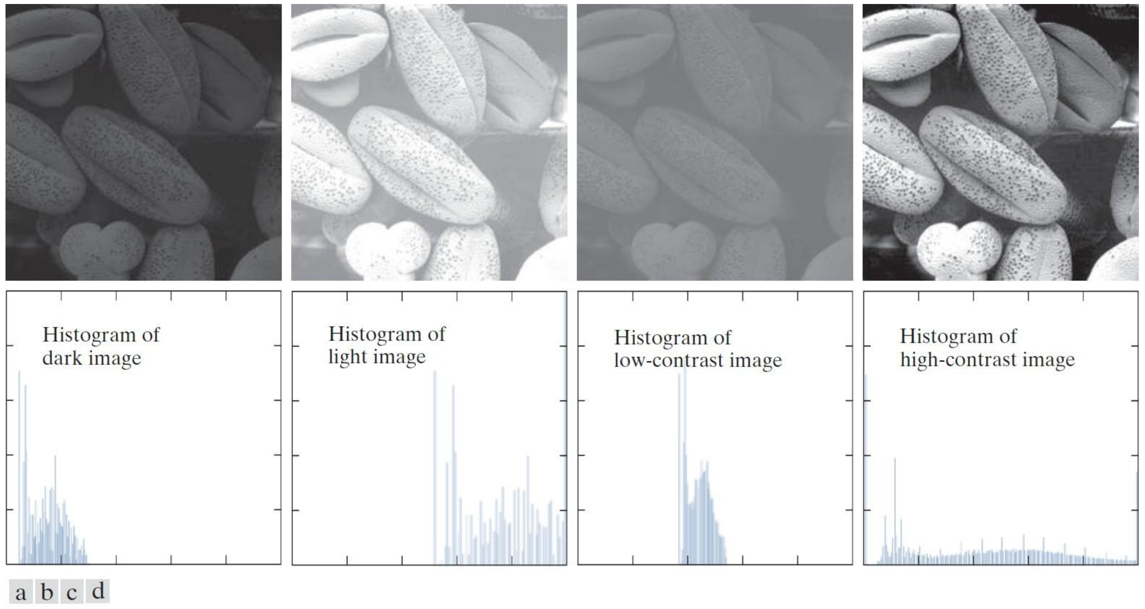


FIGURE 3.16 Four image types and their corresponding histograms. (a) dark; (b) light; (c) low contrast; (d) high contrast. The horizontal axis of the histograms are values of r_k and the vertical axis are values of $p(r_k)$.

Histogram Equalization: Continuous Case

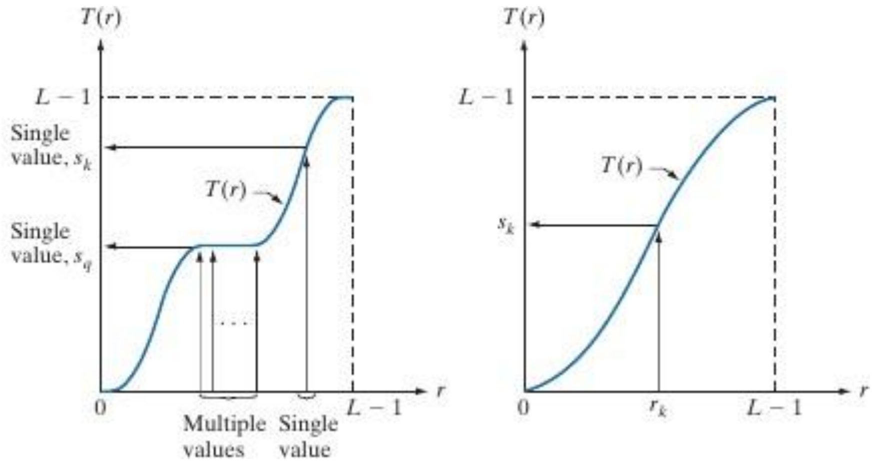
- Assume a continuous range of intensity values, **r** in the range **[0, L-1]**.
- Let's have a transformation (intensity mapping) function:

$$\boxed{s = T(r) \quad 0 \le r \le L-1}$$

- Function conditions:
  - **a) T(r)** is a monotonic increasing function in the interval **[0,L-1]**;
  - **b) 0 ≤ T(r) ≤ L-1 for 0 ≤ r ≤ L-1**.
- Function **T** is one-to-one and reversible if it is strictly monotonic (**a'**).

a b

FIGURE 3.17 (a) Monotonic increasing function, showing how multiple values can map to a single value. (b) Strictly monotonic increasing function. This is a one-to-one mapping, both ways.



Histogram Equalization: Continuous Case

- Condition (**a**) guarantees that output intensity values will never be less than the corresponding input values, preventing artifacts and intensity reversals.
- Condition (**b**) guarantees that the range of the output intensities is the same as the input image.
- Condition (**a'**) guarantees that the mapping function from **r** to **s** is one-to-one and reversible.

Histogram Equalization: Continuous Case

- Let $\mathbf{p_r(r)}$ and $\mathbf{p_s(s)}$ be the *Probability Distribution Functions* (PDFs) on intensity values in the two images.
- From probability theory, if $\mathbf{p_r(r)}$ and **T(r)** are known, and if **T(r)** is continuous and differentiable over the range of the values, then the PDF of the transformed variable **s** is:

$$\boxed{p_s(s) = p_r(r) \left| \frac{dr}{ds} \right|}$$

- This is too general. Let's assume a more specific function in DIP:

$$\boxed{s = T(r) = (L-1) \int_0^r p_r(w)\, dw}$$

where **w** is the dummy integration variable.

- The integral on the right is the *Cumulative Distribution Function* (CDF).

Histogram Equalization: Continuous Case

- Let’s assume a more specific function in DIP:

$$s = T(r) = (L-1)\int_{0}^{r} p_r(w)\,dw$$

  where **w** is the dummy integration variable.

- Since PDFs are always positive, and the integral is the area under the function, the transformation function is monotonic and increasing, satisfying condition (**a**); Not necessarily condition (**a’**)
- Also, for the lower limit **r=0**, we have **s=0** and for the upper limit **r=L-1**, the integral adds up to 1, which makes **s=L-1**. This satisfies condition (**b**) which states the range of the output intensities should be the same as input intensities.
- Now let’s drive the PDF of **s**.

Histogram Equalization: Continuous Case

$$\frac{ds}{dr} = \frac{dT(r)}{dr} = (L-1)\frac{d}{dr}\left[\int_0^r p_r(w)dw\right] = (L-1)p_r(r)$$

- Therefore, **p$_s$(s)** can be derived as:

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| = p_r(r)\left|\frac{1}{(L-1)p_r(r)}\right| = \frac{1}{L-1} \quad \text{for } 0 \le s \le L-1$$

which is a ***uniform*** distribution!

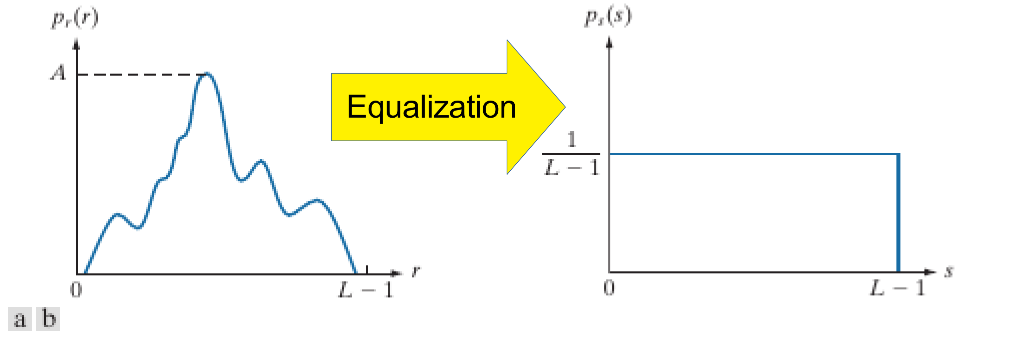


FIGURE 3.18 (a) An arbitrary PDF. (b) Result of applying Eq. (3-11) to the input PDF. The resulting PDF is always uniform, independently of the shape of the input.

Histogram Equalization: Continuous Case

- An example:

$$p_r(r) = \begin{cases} \dfrac{2r}{(L-1)^2} & 0 \le r \le L-1 \\ 0 & else \end{cases}$$

- Then we have:

$$s = T(r) = (L-1)\int_0^r P_r(w)\,dw = (L-1)\int_0^r \frac{2w}{(L-1)^2}\,dw = \frac{2}{L-1}\int_0^r w\,dw = \frac{r^2}{L-1}$$

- Therefore:

$$\frac{ds}{dr} = T'(r) = \frac{2r}{L-1} \Longrightarrow \frac{dr}{ds} = \frac{L-1}{2r}$$

- And finally:

$$\boxed{p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| = \frac{2r}{(L-1)^2}\left|\frac{L-1}{2r}\right| = \frac{1}{L-1}}$$

Histogram Equalization: Discrete Case

- For discrete values, we have probabilities and summations, instead of probability densities and integrals.
- We defined the normalized histogram for an image of size **M**-by-**N** and for $\mathbf{n_k}$ as the number of pixels that have intensity $\mathbf{r_k}$ as:

$$\boxed{p_r(r_k) = \frac{n_k}{MN}}$$

- The discrete transformation can be defined as:

$$\boxed{s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{L-1}{MN}\sum_{j=0}^{k} n_j, \ \text{for } k=0,1,2,\dots,L-1}$$

  where **L** is the number of intensity levels.
- The procedure is called a ***histogram equalization*** or ***histogram linearization*** transform.

Histogram Equalization: Discrete Case

- Example: a **64**-by-**64**, 3-bit (**L=8**) image, with the provided histogram.
- Let's define **s_k** as:

$$s_k = (L-1)\sum_{j=0}^{k} p_r(r_j), \text{ for } k = 0, 1, \ldots, L-1$$

- We will have:

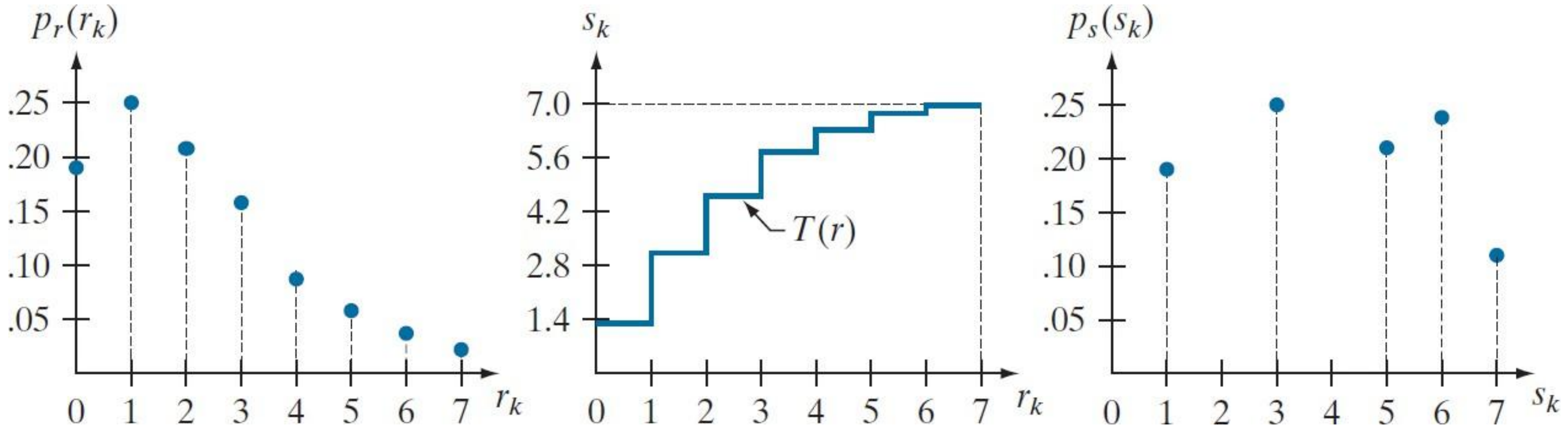
$$\begin{aligned} s_0 &= 7\, p_r(r_0) = 1.33 \\ s_1 &= 7\,(p_r(r_0) + p_r(r_1)) = 3.08 \\ s_2 &= 7\,(p_r(r_0) + p_r(r_1) + p_r(r_2)) = 4.55 \\ s_3 &= \ldots \end{aligned}$$

r_k	n_k	$p_r(r_k) = n_k / MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

Histogram Equalization: Discrete Case

- Fractional **s** values need to be rounded to the closest integer value in the range **[0,7]**.

$$\boxed{\begin{array}{ll} s_0 = 1.33 \rightarrow 1 & s_4 = 6.23 \rightarrow 6 \\ s_1 = 3.08 \rightarrow 3 & s_5 = 6.65 \rightarrow 7 \\ s_2 = 4.55 \rightarrow 5 & s_6 = 6.86 \rightarrow 7 \\ s_3 = 5.67 \rightarrow 6 & s_7 = 7.00 \rightarrow 7 \end{array}}$$



Histogram Equalization: Discrete Case

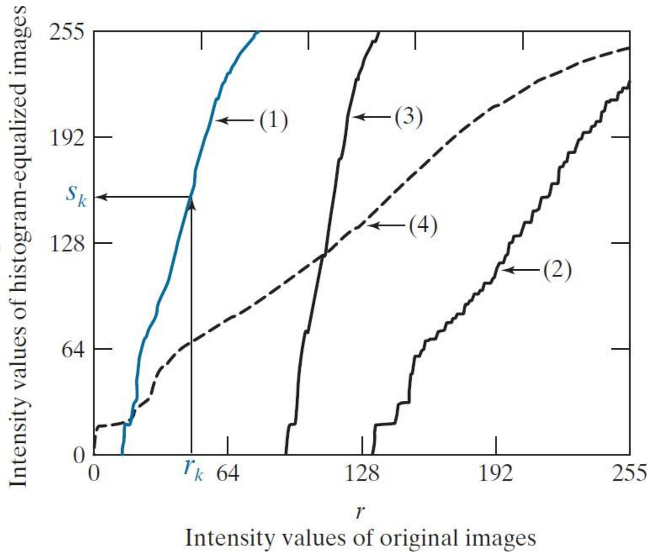
- Because a histogram is an approximate PDF, and no new intensity values are created in the process, perfectly flat histograms are rare in practical applications.
- Unlike the continuous case, it cannot be proved than the discrete histogram equalization results in a uniform histogram.
- However, histogram equalization is fully automatic, without the need for any parameter specification.
- The inverse transformation from **s** back to **r** is:

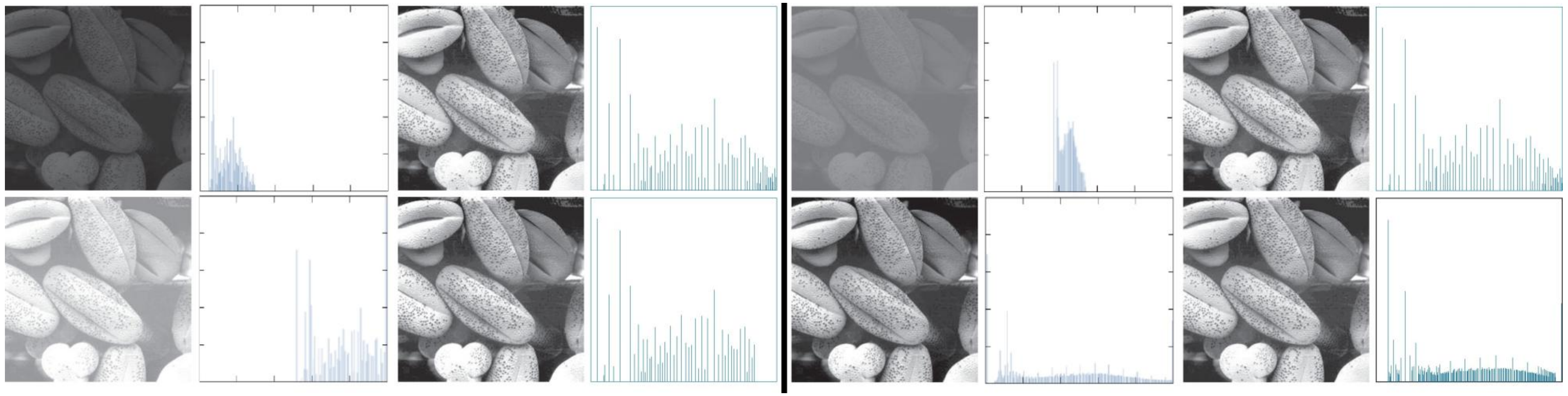
$$\boxed{r_k = T^{-1}(s_k)}$$

- This transformation satisfies two initial conditions, if all intensity levels are present in the input image. Or in another word, none of the histogram bins of the image are empty.

Histogram Equalization: Discrete Case

- Advantage: no parameters needed.
- Conceptually optimal





Histogram Matching (Specification)

- Histogram equalization produces a transformation function to generate an output image with a uniform histogram.
- This is not always ideal. What if we want the specify the shape ourselves?
- The method to achieve this is called ***histogram matching*** or ***histogram specification***.
- Let's assume the continuous case again.
- Assume an image with PDF $\mathbf{p_r(r)}$ that we want to transform to an image with PDF $\mathbf{p_z(z)}$.
- Here $\mathbf{p_z(z)}$ is not uniform.

Histogram Matching: Continuous Case

- Let **s** be a random variable defined as before:

$$\boxed{s = T(r) = (L-1)\int_0^r p_r(w)\,dw}$$

  where **w** is the dummy variable of integration.

- Let's do the same based on variable **z**:

$$\boxed{G(z) = (L-1)\int_0^z p_z(v)\,dv = s}$$

  where **v** is the dummy variable of integration.

- Combining the two equations, we have:

$$\boxed{G(z) = s = T(r) \longrightarrow z = G^{-1}(s) = G^{-1}[T(r)]}$$

Histogram Matching: Continuous Case

- The procedure can be summarized as:

  1) Obtain $\mathbf{p_r(r)}$ from the input image;

  2) Use the specified PDF, $\mathbf{p_z(z)}$, to obtain function $\mathbf{G(z)}$;

  3) Compute the inverse transformation $\mathbf{z=G^{-1}(s)}$, which is a mapping from $\mathbf{s}$ to $\mathbf{z}$.

  4) Obtain the output image by first equalizing the input ($\mathbf{r}$ to $\mathbf{s}$). Then, for each pixel $\mathbf{s}$ in the equalized image perform the inverse mapping $\mathbf{z=G^{-1}(s)}$ to obtain the corresponding pixel in the output image.

- In general, finding the analytical expression for $\mathbf{G^{-1}}$ is not a trivial task. In the discrete case this is not a problem.

Histogram Matching: Discrete Case

- In the discrete case, first histogram equalization transformation from **r** to **s** is performed:

$$\boxed{s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) \text{ for } k = 0,1,2,\ldots,L-1}$$

- Similarly given a specific value of $\mathbf{s_k}$, for a value of **q** we find $\mathbf{G(z_q)}$ such that:

$$\boxed{G(z_q) = (L-1)\sum_{i=0}^{q} p_z(z_i) = s_k}$$

  where $\mathbf{p_z(z_i)}$ is the **i**'th value of the specified histogram.

- The desired value of $\mathbf{z_q}$ is computed from the inverse transform:

$$\boxed{z_q = G^{-1}(s_k)}$$

- In practice, no need to compute the inverse of **G.** Since the intensity levels are integers, it is simple to compute all the possible values of **G** using the second equation for **q=0,1,2,….,L-1**. These values are rounded to their nearest integer value and stored in a lookup table. Given a particular value of $\mathbf{s_k}$, we look for the closest match in the table.

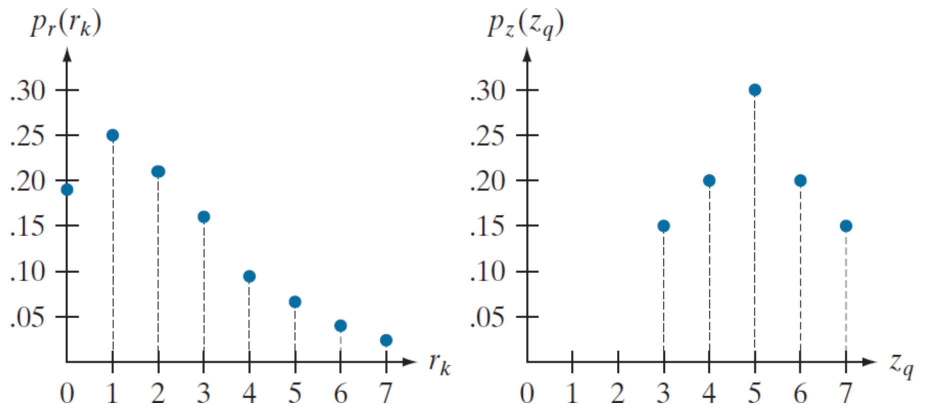
Histogram Matching: Discrete Case

- To summarize:
  1) Compute histogram $\mathbf{p_r(r)}$ of the input image, and then histogram-equalize it to the uniform histogram. Round the resulting values, $\mathbf{s_k}$, to the integer range **[0,L-1]**;
  2) Compute all values of function $\mathbf{G(z_q)}$ for **q=0,1,2,…,L-1**, round the values to integers in range **[L-1]**, and store them in a lookup table.
  3) For every value of $\mathbf{s_k}$, **k=0,1,2,…,L-1**, use the stored value of **G** from Step 2 to find the corresponding value of $\mathbf{z_q}$ so that $\mathbf{G(z_q)}$ is closest to $\mathbf{s_k}$. *When more than one value of* $\mathbf{z_q}$ *gives the same match, choose the smallest value by convention.*
  4) Form the histogram-specified image by mapping every equalized pixel with value $\mathbf{s_k}$ to the corresponding pixel with value $\mathbf{z_q}$ in the histogram-specified image, using the mappings found in Step 3.

Histogram Matching (Specification)

- Example: same **64**-by-**64**, 3-bit (**L=8**) image as the previous example.

r_k	n_k	$p_r(r_k) = n_k/MN$	z_q	Specified $p_z(z_q)$
$r_0 = 0$	790	0.19	$z_0 = 0$	0.00
$r_1 = 1$	1023	0.25	$z_1 = 1$	0.00
$r_2 = 2$	850	0.21	$z_2 = 2$	0.00
$r_3 = 3$	656	0.16	$z_3 = 3$	0.15
$r_4 = 4$	329	0.08	$z_4 = 4$	0.20
$r_5 = 5$	245	0.06	$z_5 = 5$	0.30
$r_6 = 6$	122	0.03	$z_6 = 6$	0.20
$r_7 = 7$	81	0.02	$z_7 = 7$	0.15



Histogram Matching (Specification)

- The first step is to compute the rounded histogram-equalized values, as we did in the previous example:

$$\begin{aligned} s_0 &= 1.33 \rightarrow 1 & s_4 &= 6.23 \rightarrow 6 \\ s_1 &= 3.08 \rightarrow 3 & s_5 &= 6.65 \rightarrow 7 \\ s_2 &= 4.55 \rightarrow 5 & s_6 &= 6.86 \rightarrow 7 \\ s_3 &= 5.67 \rightarrow 6 & s_7 &= 7.00 \rightarrow 7 \end{aligned}$$

- Then, we compute the values of **G(z$_q$)** using the values of **p$_z$(z$_q$)**:

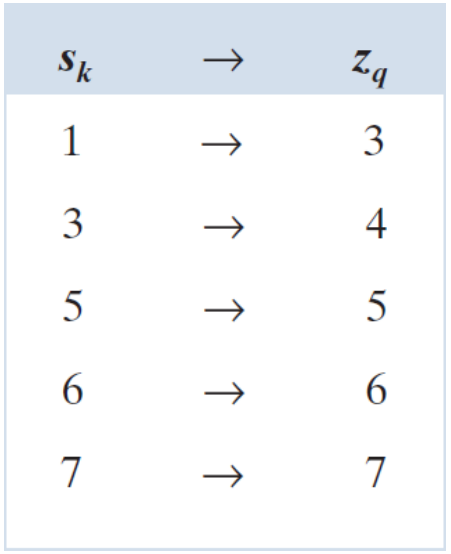
$$\begin{aligned} G(z_0) &= 0.00 & G(z_2) &= 0.00 & G(z_4) &= 2.45 & G(z_6) &= 5.95 \\ G(z_1) &= 0.00 & G(z_3) &= 1.05 & G(z_5) &= 4.55 & G(z_7) &= 7.00 \end{aligned}$$

- These values are then rounded to their nearest integer values:

$$\begin{aligned} G(z_0) &= 0.00 \rightarrow 0 & G(z_4) &= 2.45 \rightarrow 2 \\ G(z_1) &= 0.00 \rightarrow 0 & G(z_5) &= 4.55 \rightarrow 5 \\ G(z_2) &= 0.00 \rightarrow 0 & G(z_6) &= 5.95 \rightarrow 6 \\ G(z_3) &= 1.05 \rightarrow 1 & G(z_7) &= 7.00 \rightarrow 7 \end{aligned}$$

Histogram Matching (Specification)

- The first three values of **G** are equal, so it is not strictly monotonic, therefore condition (**a’**) is violated.
- Given this, we follow step three of the histogram matching approach which states we need to find the smallest value of $\mathbf{z_q}$ so that the value $\mathbf{G(z_q)}$ is the closest to $\mathbf{s_k}$. This is done for every value of $\mathbf{s_k}$ to create the required mapping from **s** to **z**.
- For example, for $\mathbf{s_0=1}$, we have $\mathbf{G(z_3)=1}$ which is a perfect match, which means that every pixel in the histogram-equalized image with value of **1**, is mapped to a pixel with value **3** in the histogram-specified image.

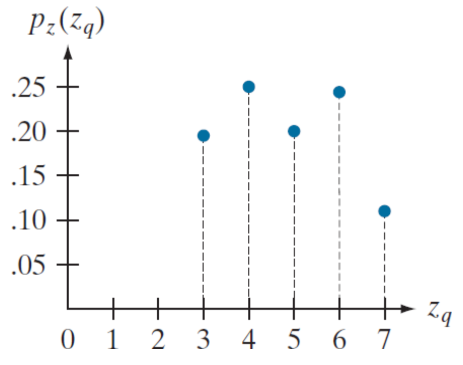


s_k	$\rightarrow$	z_q
1	$\rightarrow$	3
3	$\rightarrow$	4
5	$\rightarrow$	5
6	$\rightarrow$	6
7	$\rightarrow$	7

Histogram Matching (Specification)

- The final step is to use the computed mappings to map every pixel in the histogram-equalized image into the corresponding pixel in the newly created histogram-specified image.
- The final result does not match the specified histogram exactly, however the general trend was achieved.
- Obtaining the histogram-equalized image is not necessary. We can list the mappings from **r** to **s**, and from **s** to **z**, and then use these mappings to map the original pixels directly into the pixels of the histogram-specified image.

z_q	Specified $p_z(z_q)$	Actual $p_z(z_q)$
$z_0 = 0$	0.00	0.00
$z_1 = 1$	0.00	0.00
$z_2 = 2$	0.00	0.00
$z_3 = 3$	0.15	0.19
$z_4 = 4$	0.20	0.25
$z_5 = 5$	0.30	0.21
$z_6 = 6$	0.20	0.24
$z_7 = 7$	0.15	0.11

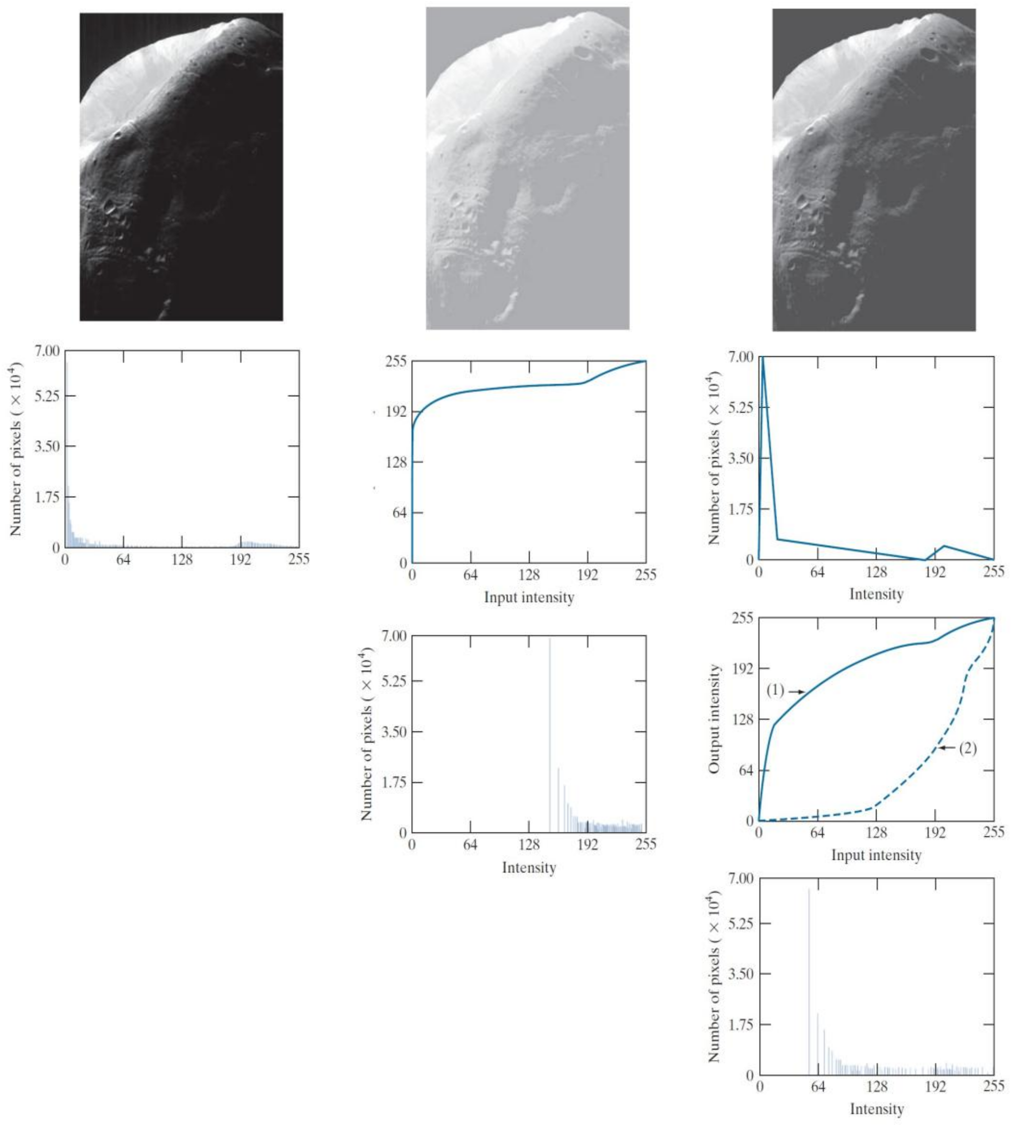


Histogram Matching vs. Histogram Equalization

- Left:
  1) Input image
  2) Initial Histogram

- Middle
  1) Histogram equalization result
  2) Histogram equalization transformation
  3) Histogram of histogram equalized image

- Right
  1) Histogram specification result
  2) Specified histogram
  3) Transformation **G(z$_q$)**, labeled (1), and **G^{-1}(s$_k$)** labeled (2)
  4) Histogram of the histogram-specified image

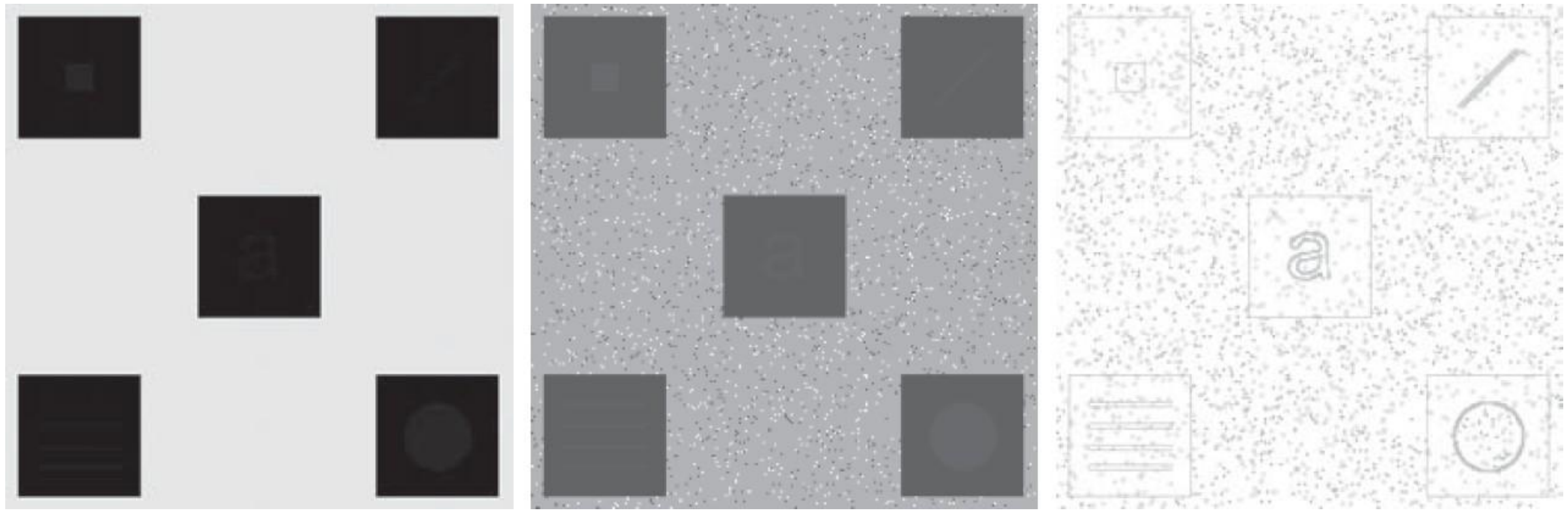


Global Histogram Processing

- Both histogram equalization and histogram matching are ***global*** histogram processing methods, since pixels are modified based on the intensity distribution of the entire image.
- If the aim is to enhance details over small areas of an image, they fail, since the number of pixels in small areas have less significant impact in the computation of global transformations.
- To remedy this, we can devise transformations based on the intensity distribution of pixel neighborhoods.
- For this, we define a neighborhood, move its center from pixel to pixel in a raster scanning scheme, and at each location, histogram equalization or matching is performed and a new mapped intensity is computed for the center pixel of the neighborhood.
- Since only one row or column of the neighborhood changes in a one-pixel translation, the histogram from the previous step can be updated so there is no need for full calculation of the histogram in each neighborhood.

Global vs. Local Histogram Processing

- Example of global vs. local (3-by-3 neighborhood) histogram equalization in a noisy image, with very faint shapes in the dark regions.



Fundamentals of Spatial Filtering

- Two components in any spatial filter definition:
  - A neighborhood
  - An operation defined in the neighborhood
- Spatial Filtering:
  - Linear spatial filters (e.g. mean)
  - Nonlinear spatial filters (e.g. median)

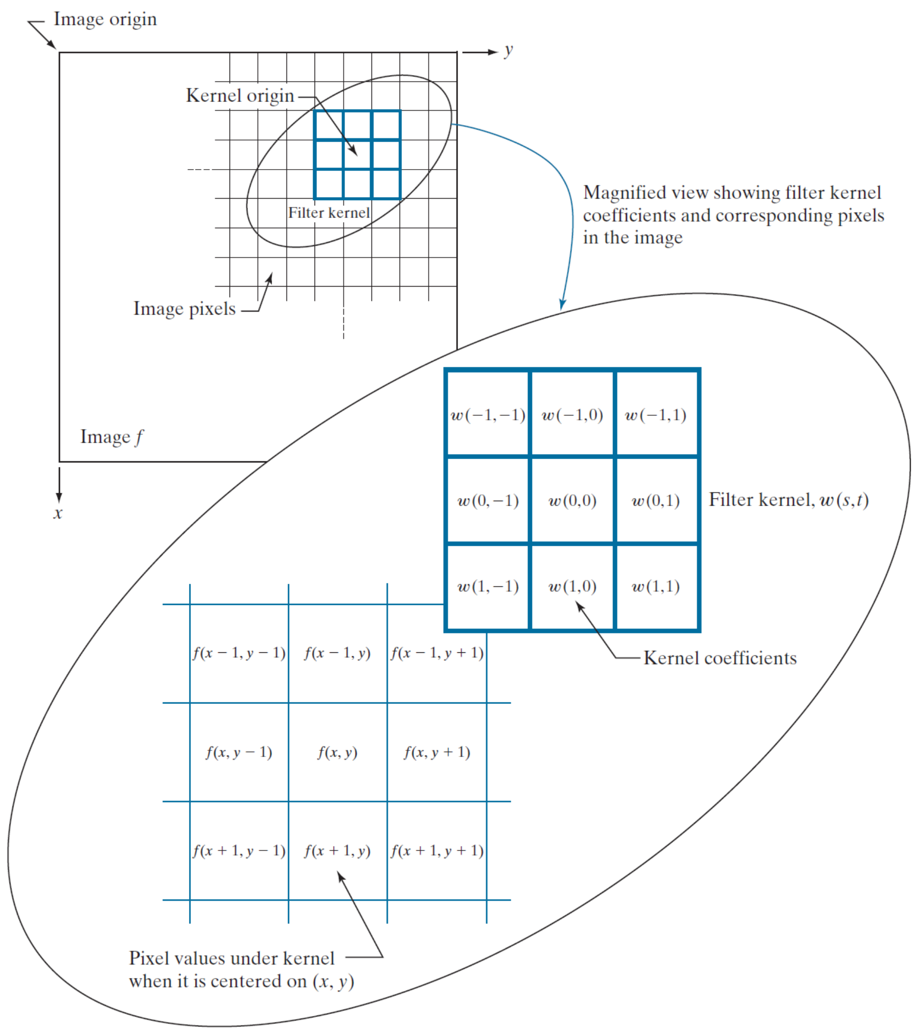


Image origin
y
Kernel origin
Filter kernel
Image pixels
Image f
x
Magnified view showing filter kernel coefficients and corresponding pixels in the image
$w(-1,-1)$ $w(-1,0)$ $w(-1,1)$
$w(0,-1)$ $w(0,0)$ $w(0,1)$
$w(1,-1)$ $w(1,0)$ $w(1,1)$
Filter kernel, $w(s,t)$
Kernel coefficients
$f(x-1,y-1)$ $f(x-1,y)$ $f(x-1,y+1)$
$f(x,y-1)$ $f(x,y)$ $f(x,y+1)$
$f(x+1,y-1)$ $f(x+1,y)$ $f(x+1,y+1)$
Pixel values under kernel when it is centered on (x, y)

Linear Spatial Filtering

- Sum-of-products operation between an image **f** and a filter kernel **w**.
- The kernel is an array which defines the neighborhood of operation, and its coefficients determine the nature of the filter.
- A kernel is also called *mask*, *template*, and *window*.
- Assume a **3**-by-**3** kernel like the figure.

$w(-1,-1)$	$w(-1,0)$	$w(-1,1)$
$w(0,-1)$	$w(0,0)$	$w(0,1)$
$w(1,-1)$	$w(1,0)$	$w(1,1)$

- At any point **(x,y)**, the response, **g(x,y)**, of the filter is the sum-of-products of the kernel coefficients and the image pixels encompassed by the kernel:

$$g(x,y)=w(-1,-1)f(x-1,y-1)+w(-1,0)f(x-1,y)+\ldots$$
$$+w(0,0)f(x,y)+\ldots+w(1,1)f(x+1,y+1)$$

- As the coordinates **(x,y)** change, the center of the kernel moves from pixel to pixel, generating the filtered image **g**.

Linear Spatial Filtering

- The center coefficient of the kernel, **w(0,0)**, aligns with the pixel at location **(x,y)**.
- A generalized kernel of size (**mxn**), with **m=2a+1** and **n=2b+1**, where **a** and **b** are non-negative integers, can be applied to the image to create the filtered image **g(x,y)** such as:

$$g(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x+s,y+t)$$

where **x** and **y** are changed so the center of the kernel goes through every pixel in image **f** once.

Correlation vs. Convolution: 1D Case

- Correlation: (also shown with ☆)

$$(w \oplus f)(x) = \sum_{s=-a}^{a} w(s) f(x+s)$$

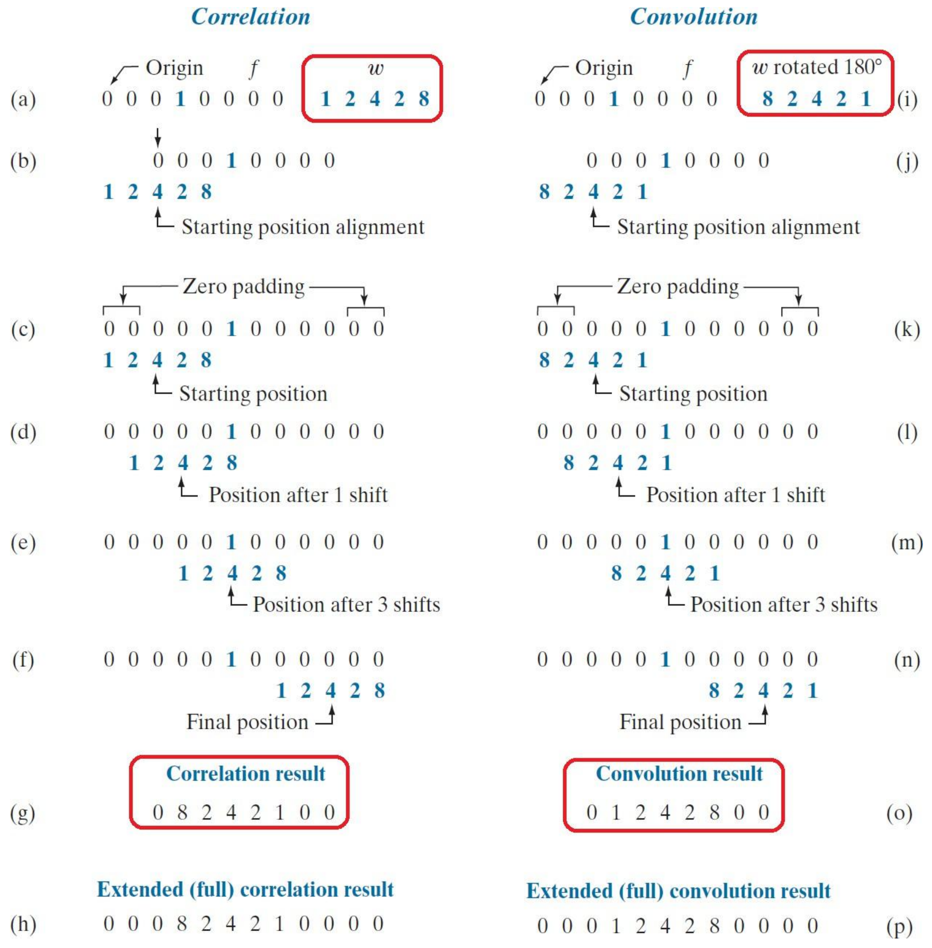
- Convolution: (also shown with ★)

$$(w \otimes f)(x) = \sum_{s=-a}^{a} w(s) f(x-s)$$

- Function **f** is a discrete unit impulse.

$$\delta(x - x_0) = \begin{cases} 1 & \text{if } x = x_0 \\ 0 & \text{otherwise} \end{cases}$$

- Pre-rotating the kernel results in an exact copy of the kernel.
- ***Linear spatial filtering and spatial convolution are synonymous.***



Correlation
Convolution
Origin
f
w
w rotated 180°
(a) 0 0 0 1 0 0 0 0 1 2 4 2 8
(i) 0 0 0 1 0 0 0 0 8 2 4 2 1
(b) 0 0 0 1 0 0 0 0 / 1 2 4 2 8
(j) 0 0 0 1 0 0 0 0 / 8 2 4 2 1
Starting position alignment
Zero padding
(c) 0 0 0 0 0 1 0 0 0 0 0 0 / 1 2 4 2 8
(k) 0 0 0 0 0 1 0 0 0 0 0 0 / 8 2 4 2 1
Starting position
(d) (l)
Position after 1 shift
(e) (m)
Position after 3 shifts
(f) (n)
Final position
(g) Correlation result 0 8 2 4 2 1 0 0
(o) Convolution result 0 1 2 4 2 8 0 0
(h) Extended (full) correlation result 0 0 0 8 2 4 2 1 0 0 0 0
(p) Extended (full) convolution result 0 0 0 1 2 4 2 8 0 0 0 0

Correlation vs. Convolution: 2D Case

- The same can be said about 2D case.
- Correlation:

$$(w \oplus f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

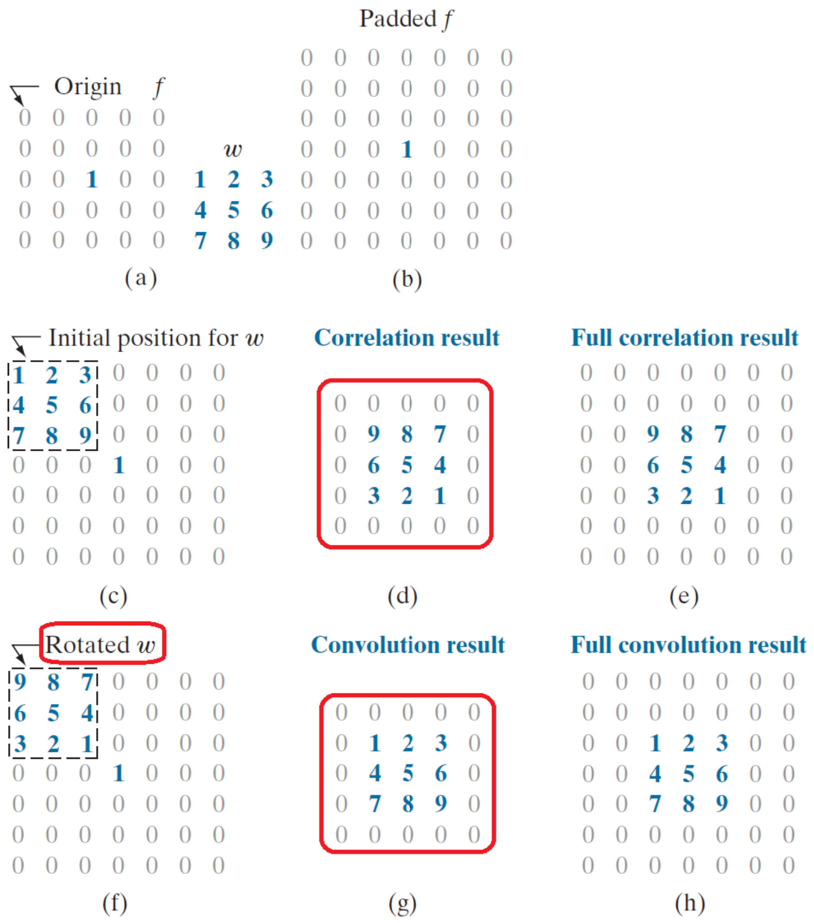
- Convolution:

$$(w \otimes f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

- Function **f** is a discrete unit impulse.

$$\delta(x-x_0, y-y_0) = \begin{cases} 1 & \text{if } x = x_0 \text{ and } y = y_0 \\ 0 & \text{otherwise} \end{cases}$$

- Pre-rotating the kernel results in an exact copy of the kernel.
- ***Linear spatial filtering and spatial convolution are synonymous.***



Correlation vs. Convolution: Properties

Properties	Convolution	Correlation
Commutative	$f \otimes g = g \otimes f$	-
Associative	$f \otimes (g \otimes h) = (f \otimes g) \otimes h$	-
Distributive	$f \otimes (g+h) = (f \otimes g) + (f \otimes h)$	$f \oplus (g+h) = (f \oplus g) + (f \oplus h)$

- Because of the commutative property of convolution, it is not important whether the kernel or the image is pre-rotated.
- Also, performing multistage filtering is possible, if the kernels are convolved first, and the resulting kernel is applied to the image.

$$w = w_1 \otimes w_2 \otimes \ldots \otimes w_N$$

Separable Filter Kernels

- A 2D function **G(x,y)** is said to be separable if it can be written as the product of two 1D functions, **G_1(x)** and **G_2(y)**:

$$\boxed{G(x,y)=G_1(x)*G_2(y)}$$

- For example, the following kernel *w* is separable, since:

$$w=\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \longrightarrow \boldsymbol{c}=\begin{bmatrix}1\\1\end{bmatrix} \text{ and } \boldsymbol{r}=\begin{bmatrix}1\\1\\1\end{bmatrix} \longrightarrow w=\boldsymbol{c}\ \boldsymbol{r}^{\boldsymbol{T}}$$

- A separable kernel of size **mxn** can be expressed as the outer product of two vectors **v** and **w**:

$$\boxed{w=\boldsymbol{v}\ \boldsymbol{w}^{\boldsymbol{T}}}$$

  where **v** and **w** are vectors of size **mx1** and **nx1** respectively.
- For a square kernel of size **mxm** we have:

$$\boxed{w=\boldsymbol{v}\ \boldsymbol{v}^{\boldsymbol{T}}}$$

- These are equivalent to the 2D convolution of a column vector and a row vector.

Separable Filter Kernels

- Why is it important? Recall the commutative and associative properties of the colvolution:

$$f \otimes g = g \otimes f \text{ and } f \otimes (g \otimes h) = (f \otimes g) \otimes h$$

- Now with a separable kernel *w*:

$$w \otimes f = (w_1 \otimes w_2) \otimes f = (w_2 \otimes w_1) \otimes f = w_2 \otimes (w_1 \otimes f) = (w_1 \otimes f) \otimes w_2$$

- Assume an image of size **MxN** and a kernel of size **mxn**.
- The computational advantage of a separable in comparison to a non-separable kernel in terms of the number multiplications and summations can be derived as (how? DIY!):

$$\boxed{C = \frac{MNmn}{MN(m+n)} = \frac{mn}{m+n}}$$

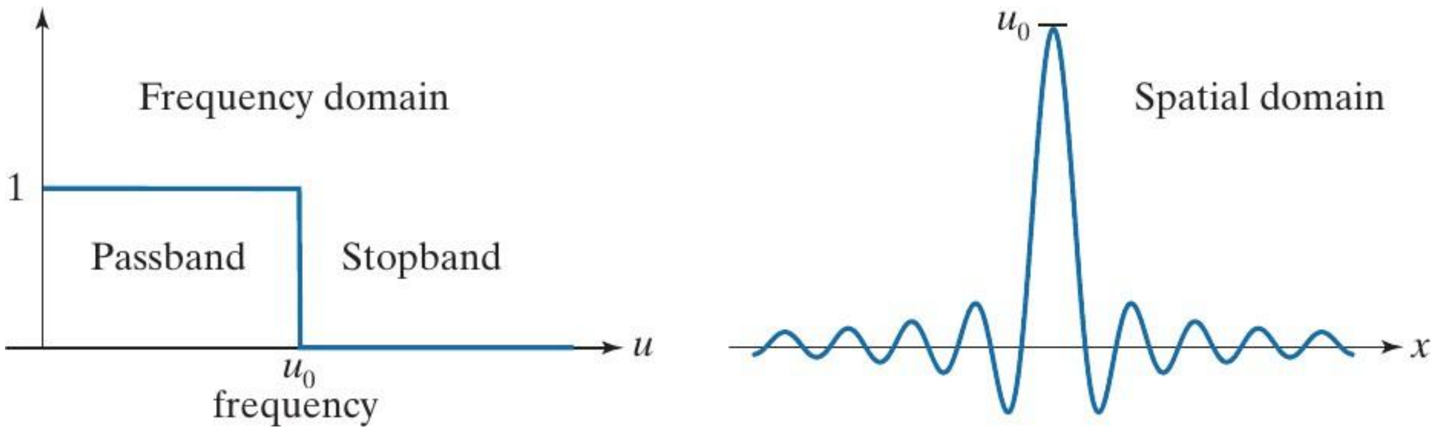
- For a kernel of 11x11, this is 5.5. What is it for a kernel of 101x101?

Separable Filter Kernels

- Matrix formed by product of a column vector and a row vector always has rank of 1.
- In another word, rows and columns of the matrix are linearly dependent, meaning the rows differ only by a constant multiplier. The same is true for the columns.
- When a matrix has a rank of 1 (***rank*** function in MATLAB), to get the separable kernels:
  1) Find any non-zero element in the matrix, denote its value as *E*.
  2) Form vectors **c** and **r** equal to the column and row in the kernel containing the element in Step 1, respectively;
  3) Given these, $\mathbf{v}=\mathbf{c}$ and $\mathbf{w}^T=\mathbf{r}/E$.

Filtering in Spatial and Frequency Domains

- Fourier Transform (FT) is used to go from the spatial to frequency domain.
- FT decomposes the signal (image) into its consisting frequency components.
- Two fundamental properties:
  1) Convolution in the spatial domain, is equivalent to multiplication in the frequency domain, and vice versa.
  2) An impulse function of amplitude **A** in the spatial domain, is a constant function of value **A** in the frequency domain, and vice versa.



How to Build Spatial Filter Kernels?

- There are three main approaches for building spatial kernel filters:
  1) Formulating based on mathematical properties; for example using integration (averaging) for blurring an image, or computing local derivatives to sharpen an image.
  2) Sampling a 2D spatial function whose shape has a desired property; for example creating a weighted-average filter by sampling a 2D Gaussian function, or sampling the inverse FT of a 2D filter that is specified in the frequency domain.
  3) Designing a spatial filter with a specified frequency response; for example designing a 1D spatial filter and then forming a separable or circularly symmetric 2D version.

What is next?

- Smoothing (lowpass) spatial filters
- Sharpening (highpass) spatial filters
- Highpass, bandreject, and bandpass filters from lowpass filters
- Combining spatial enhancement methods