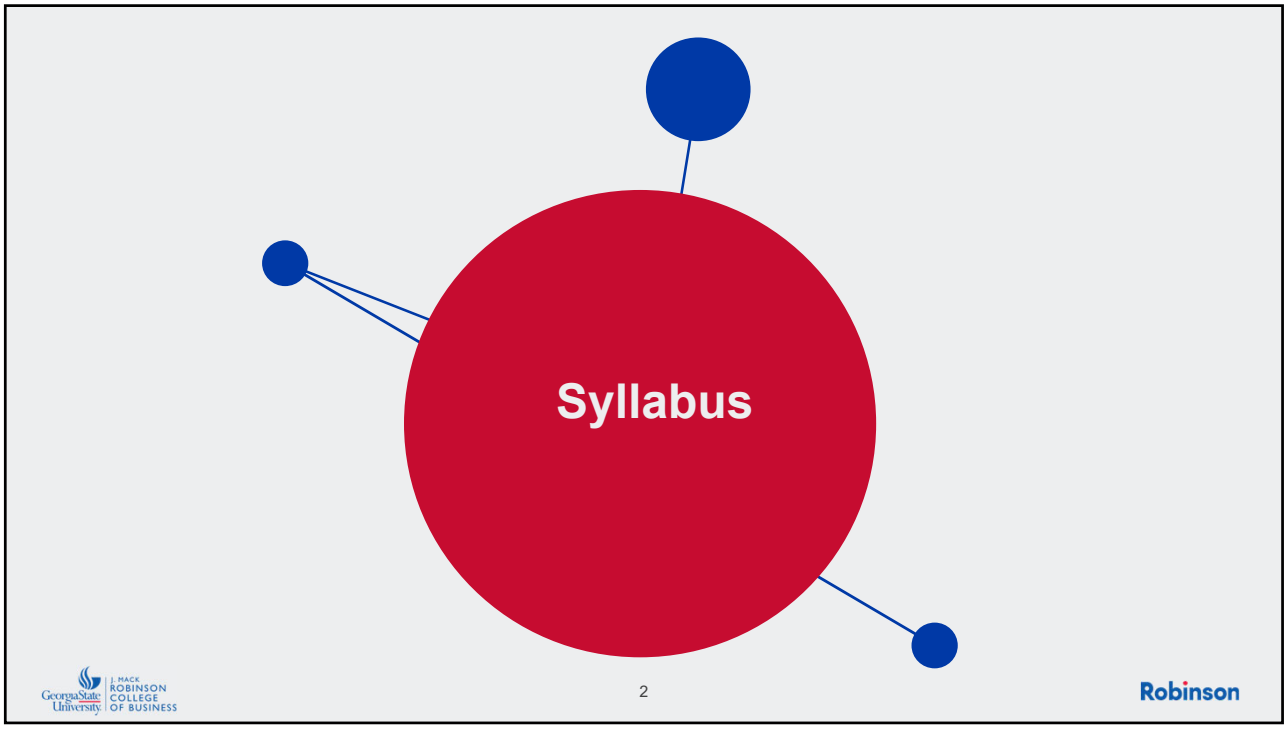
Image Analytics

MSA 8650

fall Semester, 2019

1



2

Paper Presentation

Week and Date	Presentation Topics
September 25	1
October 9	1
October 23	2
November 6	1
November 20	1

Presentation time is 25 minutes for each group

Group Projects

Type 1: Text related

Group Projects

Type 2: Image related

Pine tree log

Hardwood tree log

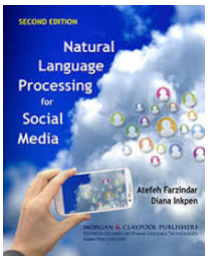
What do pictures in social media tell us?

Other business related problems?

Robinson

Group Projects

Type 3: Texts and Images Combined



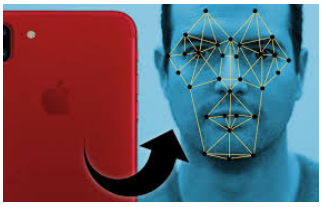
Robinson

Image Analytics Overview

1. Concepts of image formation, representation, operations
2. Spatial domain filtering: correlation and convolution
3. Feature extraction and image segmentation
4. Image matching and recognition

Smart cars

facial-recognition in iPhones



Robinson

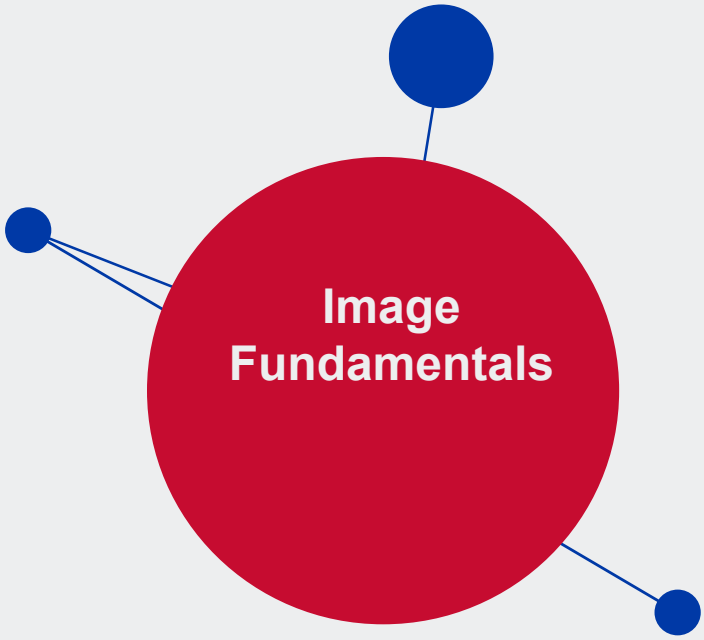
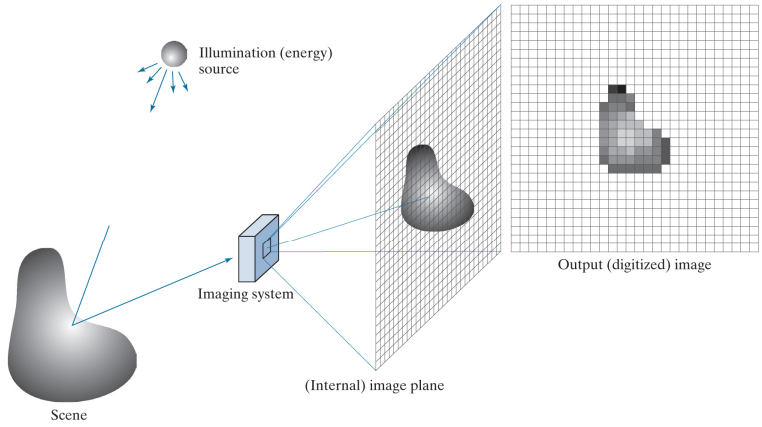


Image Formation

Image FormationType equation here.



a b c d e An example of digital image acquisition. (a) Illumination (energy) source. (b) A scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

Images are denoted by two-dimensional functions of the form $f(x, y)$. The value of *f* at spatial coordinates (x, y) is a scalar quantity whose physical meaning is determined by the source of the image, and whose *values are proportional to energy radiated by a physical source* (e.g., electromagnetic waves).

- f(x, y) must be nonnegative and finite;
- $f(x, y) = i(x, y) \cdot r(x, y)$ for $0 \leq i(x, y) \leq \infty$ is called ***illumination component*** that is **the amount of source illumination** incident on the scene being viewed; $0 \leq r(x, y) \leq 1$ is called ***reflection component*** which is the **amount of illumination reflected** by the objects in the scene.

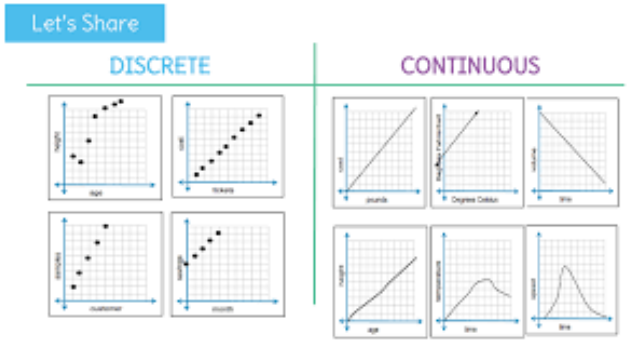
Robinson

Image Representation

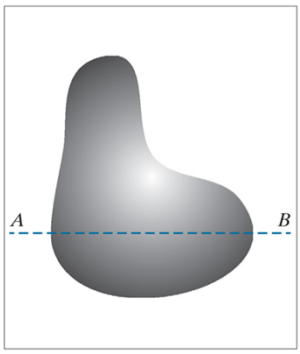
Representing Digital Images

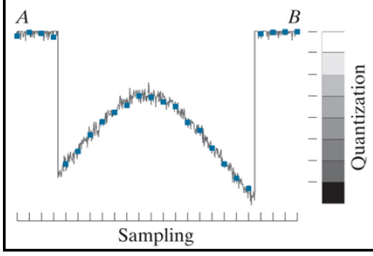
Let $f(s,t)$ represent a continuous image function of two continuous variables, s and t. The function can be converted into a digital image by sampling and quantization.

What is sampling and quantization?



Representing Digital Image





Figures show a continuous image f that we want to convert to digital form.

- An image may be continuous with respect to the x- and y-coordinates, and also in amplitude.
- To digitize it, we have to sample the function in both coordinates and also in amplitude.
- Digitizing the coordinate values is called *sampling*.
- Digitizing the amplitude values is called *quantization*.

- Along the line of AB, a lot of random variations that are due to image noise.
- To sample the function, we take equally spaced samples along line AB. The samples are shown as small dark squares superimposed on the function, and their discrete spatial locations are indicated by corresponding tick marks in the bottom of the figure.
- To quantize the intensity values, we divide the intensity scale into eight (2-256) intervals ranging from black to white. The continuous intensity levels are quantized by assigning one of the eight values to each sample, depending on the proximity of a sample to a vertical tick mark.

Robinson

Representing Digital Image

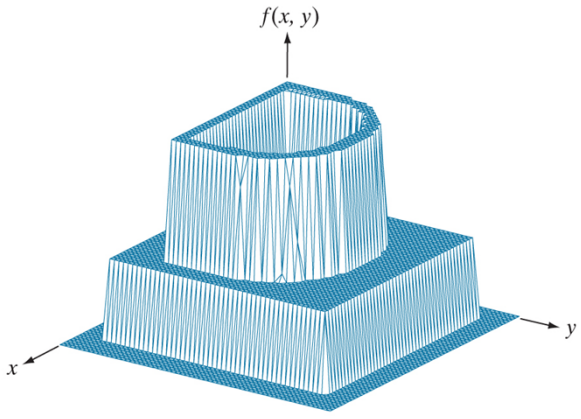


Image plotted as a surface

One way to represent image is *surface* in a 3-D. Two axes determine the spatial location and the third axis is the values of f as a function of x and y. This representation is useful when working with grayscale sets whose elements are expressed as triplets of the form (x, y, z), where x and y are spatial coordinates and z is the value of f at coordinates (x, y).

Georgia State University J. Mack Robinson College of Business

Robinson

Representing Digital Image

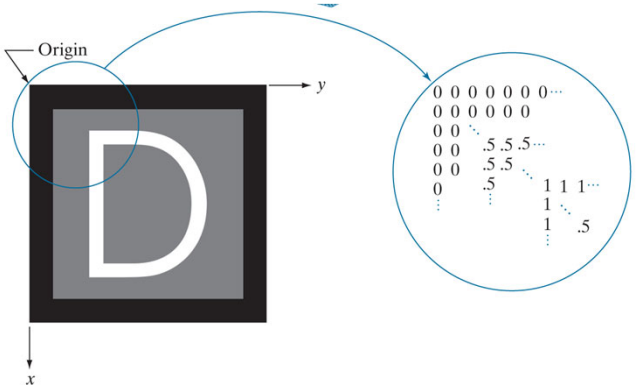


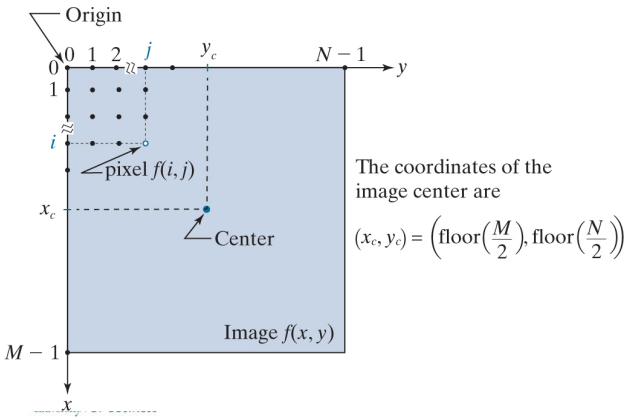
Image displayed as a visual intensity array.

The second representation is more common, and it shows $f(x, y)$ as it *would appear on a computer display or photograph*. Here, the intensity of each point in the display is proportional to the value of f at that point. In this figure, there are only three equally spaced intensity values. If the intensity is normalized to the interval [0, 1], then each point in the image has the value 0, 0.5, or 1.

Representing Digital Image

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

The third representation is an array (matrix) composed of the numerical values of $f(x, y)$. This is the representation used for computer processing. Each element of this array is called an image element, picture element or pixel.



The coordinates of the image center are

$$(x_c, y_c) = \left(\text{floor}\left(\frac{M}{2}\right), \text{floor}\left(\frac{N}{2}\right)\right)$$

- We generally use $f(i, j)$ when referring to a pixel with coordinates (i, j).
- Positive x-axis extends downward and the positive y-axis extends to the right.
- Image digitization requires that decisions be made regarding the values for M, N, and for the number, L, of discrete intensity levels. There are no restrictions placed on M and N, other than they have to be positive integers. However, digital storage and quantizing hardware considerations usually lead to the number of intensity levels, L, being an integer power of two; that is $L = 2^k$ for k being an integer.

Image Representation

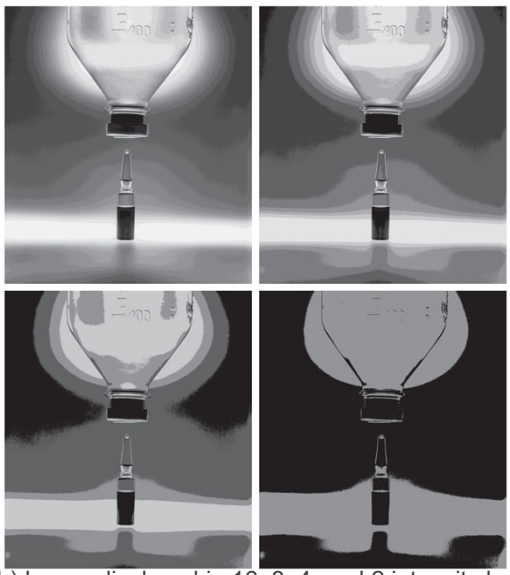
0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

Georgia State University J. MACK ROBINSON COLLEGE OF BUSINESS

Robinson

Spatial and Intensity Resolution

e f
g h



(e)-(h) Image displayed in 16, 8, 4, and 2 intensity levels.

The term **gray (or intensity) level** refers to a scalar measure of intensity that ranges from black, to grays, and finally to white.

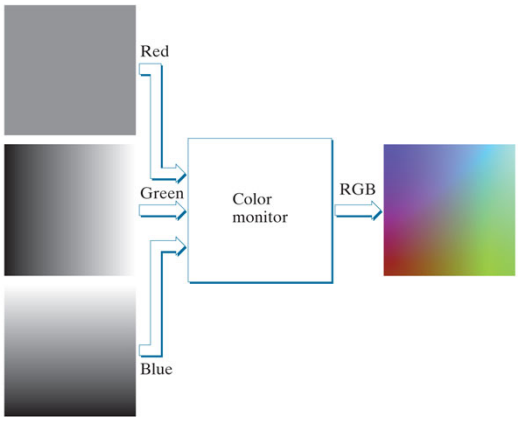
Based on hardware considerations, the number of intensity levels usually is an integer power of two. The most common number is 8 bits.

it is common practice to refer to the number of bits used to quantize intensity as the "intensity resolution." For example, it is common to say that an image whose intensity is quantized into 256 levels has 8 bits of intensity resolution

Georgia State University J. MACK ROBINSON COLLEGE OF BUSINESS

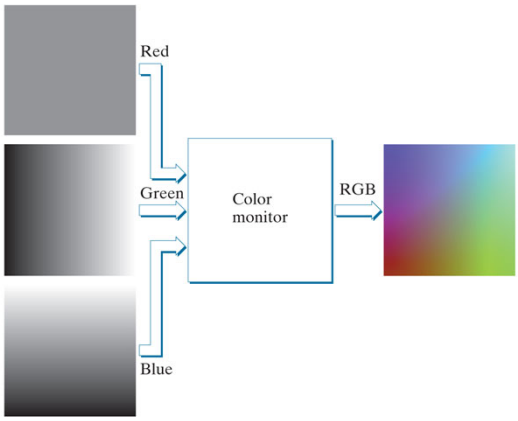
Robinson

RGB Color Model



- Images represented in the RGB color model consist of three component images, one for each primary color.
- When fed into an RGB monitor, these three images combine on the screen to produce a composite color image.
- Consider an RGB image in which each of the red, green, and blue images is an 8-bit image. Then each RGB color pixel [that is, a triplet of values (R, G, B)] has a depth of 24 bits (3 image planes times the number of bits per plane).

RGB Color Model



- Observe that each component image into the monitor is a grayscale image.
- In the component images, 0 represents black and 255 represents white.
- The monitor does the job of combining the intensities of these images to generate an RGB image;
- **With three filters that are sensitive to red, green, and blue respectively, three monochrome images can be produced**.

Convert RGB Color Image to Grayscale Image

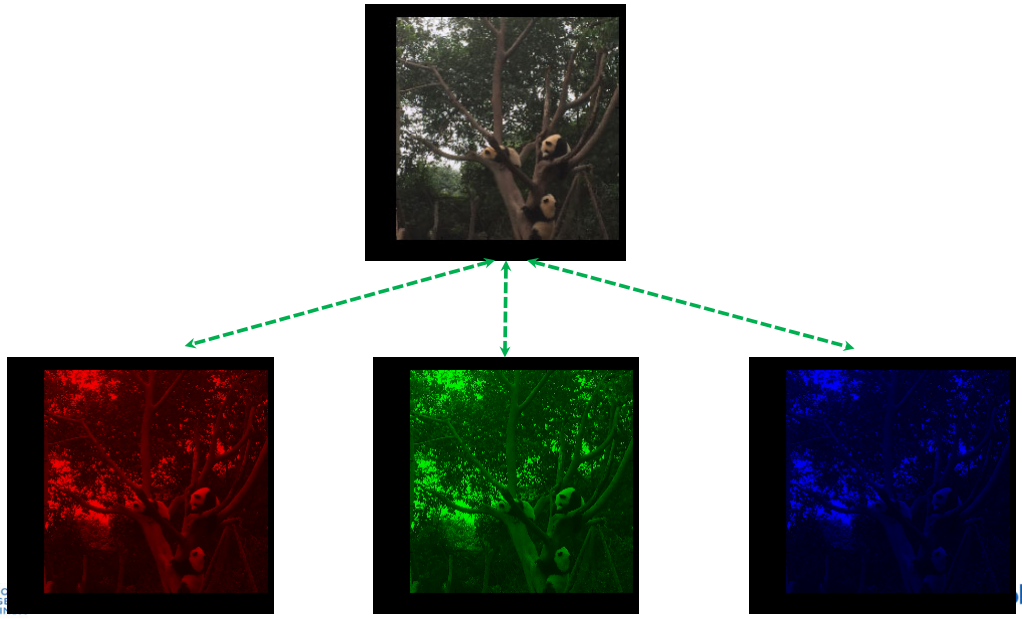
Convert RGB Color Image to Grayscale Image

Lightness method averages the most prominent and least prominent colors: (max(R, G, B) + min(R, G, B)) / 2.

Average method simply averages the values: (R + G + B) / 3.

Luminosity method is a more sophisticated version of the average method. It also averages the values, but it forms a weighted average to account for human perception. We're more sensitive to green than other colors, so green is weighted most heavily. The formula for luminosity is 0.21 R + 0.72 G + 0.07 B.

Color Image



Some Basic Relationships Between Pixels

Neighbors of a Pixel

A pixel p at coordinates (x, y) has two horizontal and two vertical neighbors with coordinates

$$(x+1, y),\ (x-1, y),\ (x, y+1),\ (x, y-1)$$

This set of pixels, called the **4-neighbors** of p, is denoted $N_4(p)$.

	0	1	2	3
0	0	1	1	1
1	1	0	1	0
2	1	1	0	1
3	1	0	1	0

The four diagonal neighbors of p have coordinates

$$(x+1, y+1),\ (x+1, y-1),\ (x-1, y+1),\ (x-1, y-1)$$

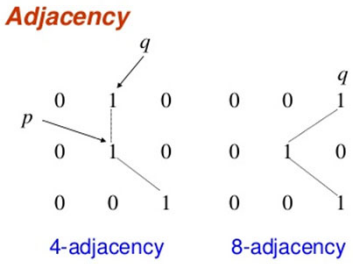
and are denoted $N_D(p)$. These neighbors, together with the 4-neighbors, are called the 8-neighbors of p, denoted by $N_8(p)$.

Some Basic Relationships Between Pixels

Adjacency

Let V be the set of intensity values used to define adjacency.

- **4-adjacency:** Two pixels p and q with values from V are 4-adjacent if q is in the set $N_4(p)$.
- **8-adjacency:** Two pixels p and q with values from V are 8-adjacent if q is in the set $N_8(p)$.



Some Basic Relationships Between Pixels

Connectivity and Region

- A digital ***path (or curve)*** from pixel p with coordinates (x_0, y_0) to pixel q with coordinates (x_n, y_n) is a sequence of distinct pixels with coordinates $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ where points (x_i, y_i) and (x_{i-1}, y_{i-1}) are adjacent.
- Let S represent a subset of pixels in an image. Two pixels p and q are said to be ***connected*** in S if there exists a path between them consisting entirely of pixels in S. For any pixel p in S, the set of pixels that are connected to it in S is called a connected component of S.
- Let R represent a subset of pixels in an image. We call R a ***region*** of the image if ***R is a connected set***.

Robinson

Image Interpolation

- Interpolation is used in tasks such as zooming, shrinking, rotating, and geometrically correcting digital images
- Interpolation is the process of using known data to estimate values at unknown locations.
- **Question: for an image of size 500×500, how to enlarge it to be size of 750×750?**

Image Interpolation

- Question: for an image of size 500×500, how to enlarge it to be size of 750×750?
- **Answer**:
  - Method 1: Nearest neighbor interpolation --- assigns each location the intensity of its nearest neighbor in the original image

    can produce undesirable distortions..

Image Interpolation

- Method 2: Bilinear interpolation --- use four nearest neighbors to estimate the intensity at a given location. In particular, let (x, y) denote the coordinates of the location to which we want to assign an intensity value, and let $f(x, y)$ denote that intensity value. For bilinear interpolation, the assigned value is obtained using the equation

$$f(x, y) = ax + by + cxy + d$$

where the four coefficients are determined from the four equations in four unknowns.

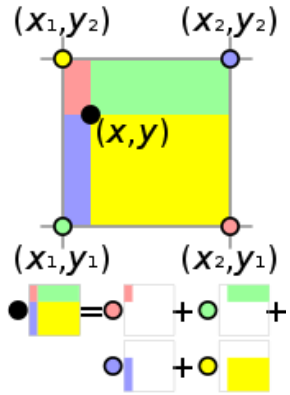


Image Interpolation

- Method 3: Bicubic interpolation --- use sixteen nearest neighbors of a point. The intensity value assigned to point (x, y) is obtained by

$$f(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j$$

- Generally, bicubic interpolation does a better job of preserving fine detail than its bilinear counterpart. Bicubic interpolation is the standard used in commercial image editing applications.

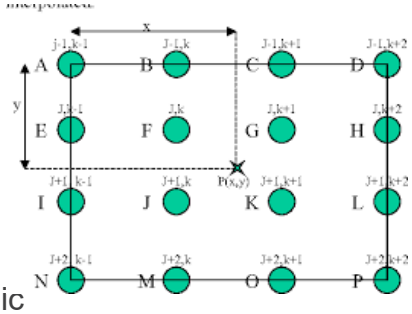


Image Interpolation

a b c

(a) nearest neighbor interpolation. (b) bilinear interpolation. (c) bicubic interpolation.

Elementwise Versus Matrix Operations

Given two matrices/images, $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ and $\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$. The elementwise product of two matrices is also called the *hadamard product* of the matrices.

The elementwise product (often denoted using the symbol $\odot$ or $\otimes$) of these two images is

$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \odot \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{bmatrix}$, which is different from the regular matrix product in linear algebra.

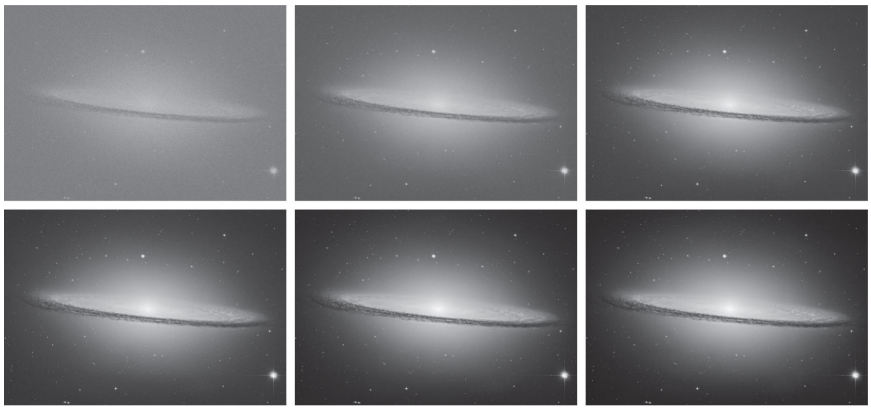
Arithmetic Operations for Images

Arithmetic operations between two images $f(x,y)$ and $g(x,y)$ are denoted as

$$s(x,y) = f(x,y) + g(x,y)$$
$$d(x,y) = f(x,y) - g(x,y)$$
$$p(x,y) = f(x,y) \times g(x,y)$$
$$v(x,y) = f(x,y) \div g(x,y)$$

These are *elementwise* operations which, as noted earlier in this section, means that they are performed between corresponding pixel pairs in f and g for $x = 0,1,2,\ldots,M-1$ and $y = 0,1,2,\ldots,N-1$. As usual, M and N are the row and column sizes of the images.

Image Summations



a b c
d e f

(a) Sample noisy image of the Sombrero Galaxy. (b)-(f) Result of averaging 10, 50, 100, 500, and 1,000 noisy images, respectively. All images are of size 1548×2238 pixels, and all were scaled so that their intensities would span the full [0, 255] intensity scale.

An example of image addition(averaging) is for noise reduction.

Arithmetic Operations for Images

Why does image addition (averaging) reduce image noise?.

Suppose that $g(x,y)$ is a corrupted image formed by the addition of noise (zero average value), η(x,y) to a noiseless image $f(x,y)$; that is,

$$g(x,y) = f(x,y) + \eta(x,y)$$

Let

$$\bar{g}(x,y) = \frac{1}{K}\sum_{i=1}^{K} g_i(x,y)$$

Then

$$E\{\bar{g}(x,y)\} = f(x,y)$$

and

$$\sigma^2_{\bar{g}(x,y)} = \frac{1}{K}\sigma^2_{\eta}(x,y) \ , \qquad \sigma_{\bar{g}(x,y)} = \frac{1}{\sqrt{K}}\sigma_{\eta}(x,y)$$

As K increases, the variability (as measured by the variance or the standard deviation) of the pixel values at each location (x, y) decreases.

This procedure of adding a set of noisy input images $\{g_i(x,y)\}$ can reduce the noise content of the output image is a technique used frequently for **image enhancement**.

Arithmetic Operations for Images

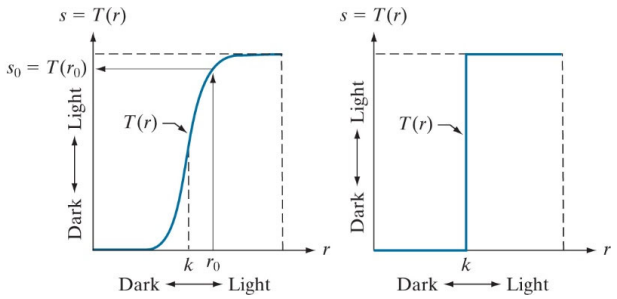
- Most images are displayed using 8 bits leading to image values being in the range from 0 to 255.
- When image values exceed the allowed range, clipping or scaling becomes necessary. For example, the values in the difference of two 8-bit images can range from a minimum of −255 to a maximum of 255, and the values of the sum of two such images can range from 0 to 510. Many software applications simply set all negative values to 0 and set to 255 all values that exceed this limit.
- **Is this a good approach?**
- Better approach:
  - First, perform the operation $g_m = g - \min(g)$ using elementwise subtraction, which creates an image whose minimum value is 0.
  - Second, perform the operation $g_s = K[g_m/\max(g_m)]$, which creates a scaled image, g_s, whose values are in the range $[0, K]$. When working with 8-bit images, setting $K = 255$.

Image Intensity Transforms

Intensity transformation: $g(x,y) = T[f(x,y)]$ where $f(x,y)$ is an input image, $g(x,y)$ is the output image, and *T is an operator on f defined over a neighborhood of point (x,y)*. The operator can be applied to the pixels of a single image or to the pixels of a set of images, such as performing the elementwise sum of a sequence of images for noise reduction.

a b

produce an image of higher contrast than the original, by darkening the intensity levels below k and brightening the levels above k. In this technique, sometimes called **contrast stretching**



$T(r)$ produces a two-level (binary) image. A mapping of this form is called a **thresholding function**

Image Histogram

- What is a histogram?
- The histogram of an image is a discrete function that is formed by *counting the number of pixels* in the image that have a *certain gray value*.
- When this function is normalized to sum up to 1 for all the gray values, it can be treated as *a probability density function* that expresses how probable it is for a certain gray value to be found in the image.

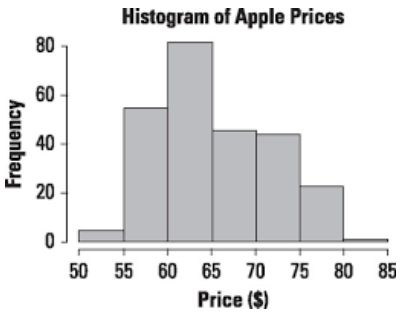


Image Histogram

let r_k, for $k = 0,1,2,\ldots,L-1$, denote the intensities of an $L-level$ digital image, $f(x,y)$. The un-normalized histogram of f is defined as

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

where n_k is the number of pixels in *f* with intensity r_k, and the subdivisions of the intensity scale are called histogram bins.

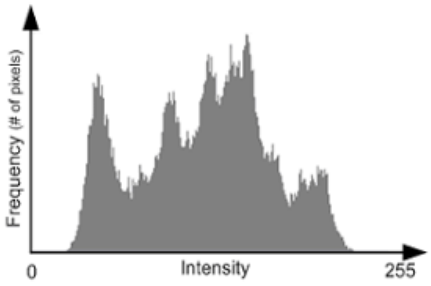


Image Histogram

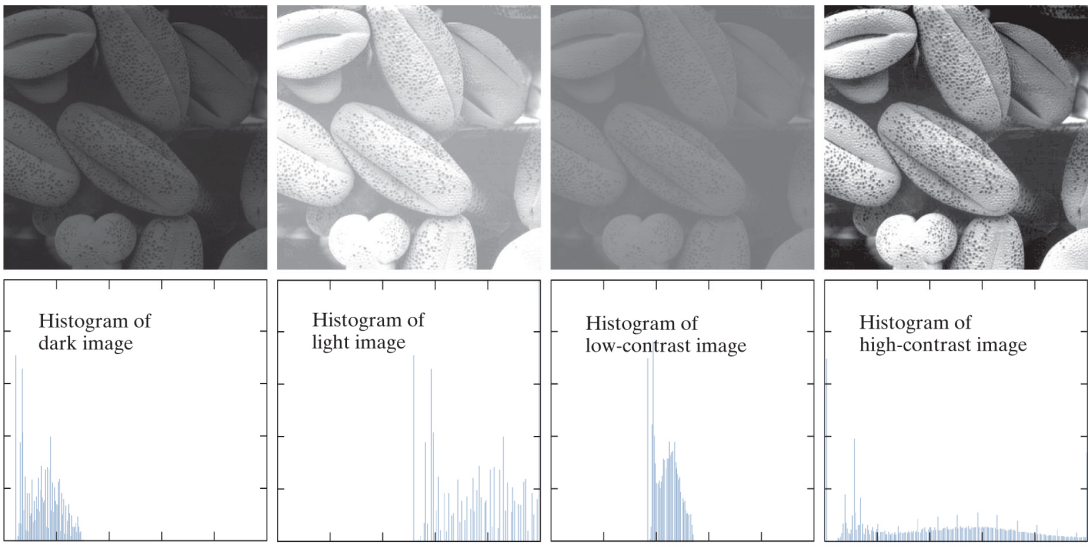
- The normalized histogram of f is defined as

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

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

- Mostly, we work with normalized histograms, which we refer to simply as histograms or image histograms. The sum of $p(r_k)$ for all values of *k* is always 1. In fact, the components of $p(r_k)$ are estimates of the probabilities of intensity levels occurring in an image.

- Histograms are simple to compute and are also suitable for fast hardware implementations, thus making histogram-based techniques a popular tool for real-time image processing.

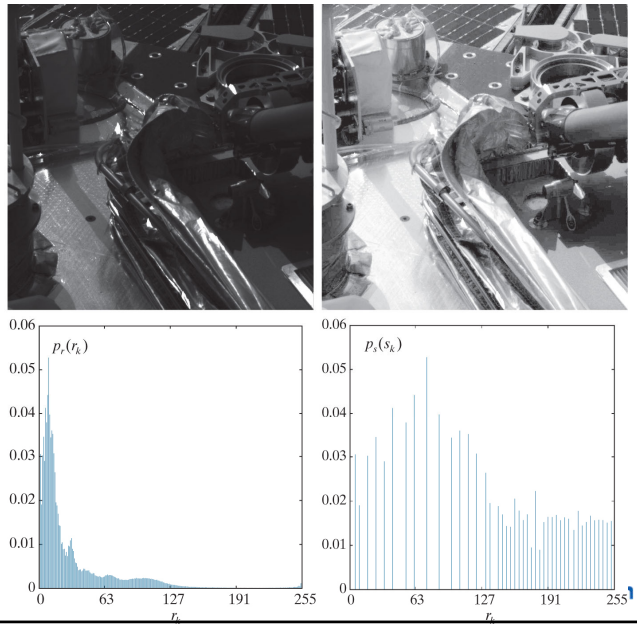


a b c d

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 rk and the vertical axis are values of p(rk).

Histogram Manipulation

a b
c d

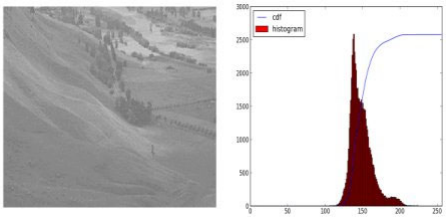


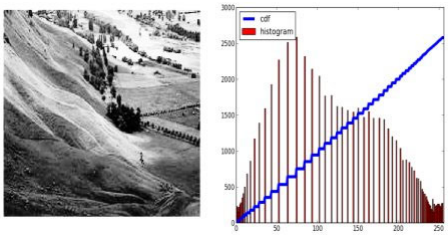
Histogram equalization

(a) Image from Phoenix Lander. (b) Result of histogram equalization. (c) Histogram of image (a). (d) Histogram of image (b).

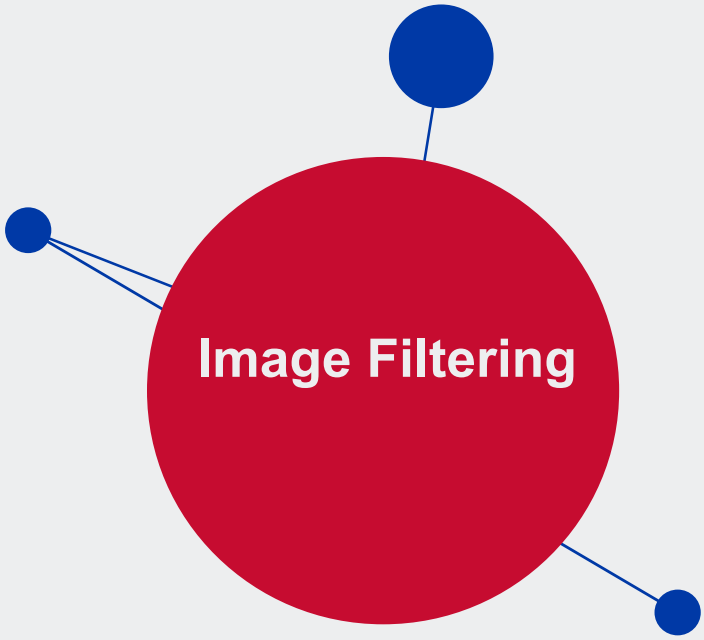
Histogram of an Image

Why do we care about histogram?





https://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html



Intensity Transformations

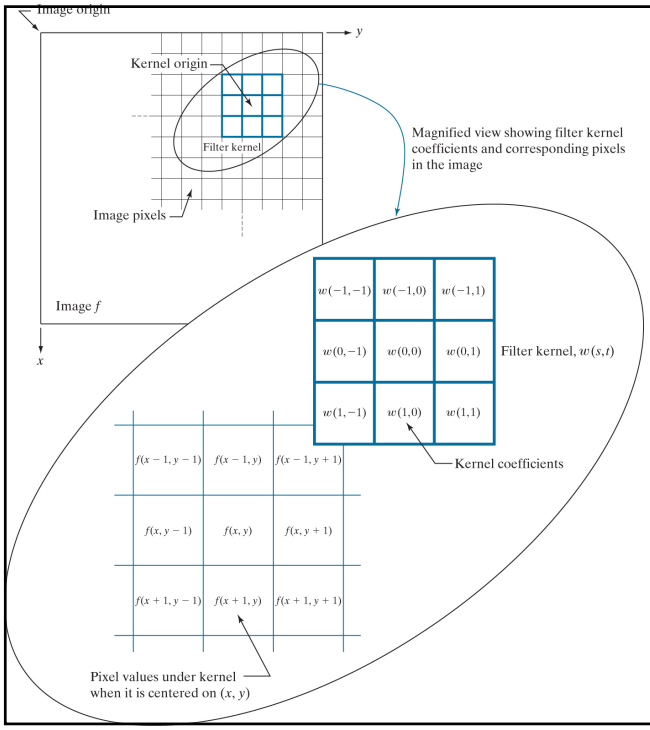
Fundamentals of Spatial Filtering

- Spatial filtering modifies an image by *replacing the value of each pixel* by a function of **the values of the pixel and its neighbors**.

- A linear spatial filter performs a *sum-of-products operation between an image f and a filter kernel, w*. The kernel is an array whose size defines the neighborhood of operation, and whose coefficients determine the nature of the filter. Other terms used to refer to a spatial filter kernel are *mask, template, and window.*

Georgia State University J. MACK ROBINSON COLLEGE OF BUSINESS

Robinson

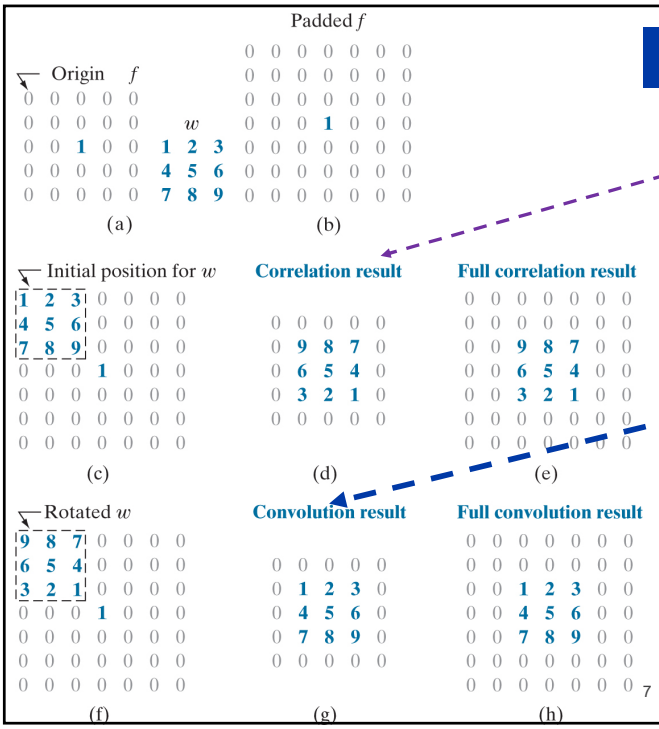


The mechanics of linear spatial filtering using a 3×3 kernel

At any point (x, y) in the image, 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:

$$
\begin{aligned}
&g(x,y) \\
&= w(-1,-1)f(x-1,y-1) \\
&+ w(-1,0)f(x-1,y) + \cdots \\
&+ w(0,0)f(x,y) + \cdots + w(1,1)f(x \\
&+ 1, y+1)
\end{aligned}
$$

Robinson



Correlation and Convolution

The ***correlation*** of a kernel w of size $m \times n$ with an image $f(x,y)$ is

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

where $a = (m-1)/2, b = (n-1)/2,$

The ***convolution*** of a kernel w of size $m \times n$ with an image $f(x,y)$ is

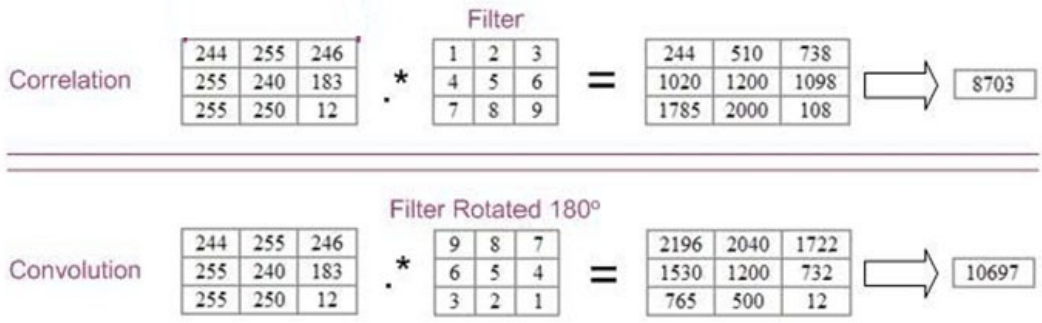
$$(w \star f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

where the minus signs align the coordinates of f and w when one of the functions is rotated by 180°.

Robinson

7

Correlation and Convolution



When will correlation and convolution be equal?

- Correlation and convolution yield the same result if the kernel values are symmetric about the center.
- If $w(-s,-t) = w(s,t)$, then correlation = convolution.

Georgia State University J. Mack Robinson College of Business

48

Robinson

Correlation and Convolution

When will correlation and convolution be equal?

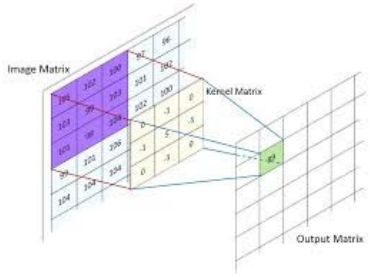
a b

$$\frac{1}{9}\times\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}\qquad \frac{1}{4.8976}\times\begin{bmatrix}0.3679&0.6065&0.3679\\0.6065&1.0000&0.6065\\0.3679&0.6065&0.3679\end{bmatrix}$$

Examples of smoothing kernels: (a) is a box kernel; (b) is a Gaussian kernel.

Spatial Filter Kernels

In general, a filter that computes the *average* of pixels in a neighborhood *blurs an image*. Computing an average is analogous to integration. Conversely, a filter that computes *the local derivative* of an image *sharpens* the image,



Smoothing (Lowpass) Spatial Filter

- ***Smoothing (also called averaging)*** spatial filters are used to reduce sharp transitions in intensity. Because random noise typically consists of sharp transitions in intensity, an obvious application of smoothing is noise reduction.
- Smoothing is used to reduce irrelevant detail in an image, where "irrelevant" refers to pixel regions that are small with respect to the size of the filter kernel.
- Another application is for smoothing the false contours that result from using an insufficient number of intensity levels in an image.

Smoothing (Lowpass) Spatial Filter

Convolving a smoothing kernel with an image blurs the image, with the degree of blurring being determined by *the size of the kernel* and the values of its coefficients.

Two important smoothing filters are:

- Box kernels
- Gaussian kernels

Smoothing (Lowpass) Spatial Filter

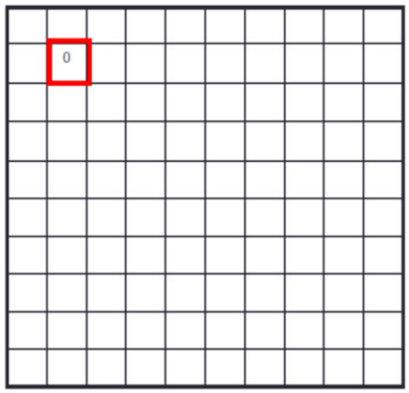
- The simplest, separable lowpass filter kernel is the ***box kernel***, whose coefficients have the same value (typically 1).
- The name "box kernel" comes from a constant kernel resembling a box when viewed in 3-D.
- An $m \times n$ box filter is an $m \times n$ array of 1's, with a normalizing constant in front, whose value is 1 divided by the sum of the values of the coefficients (i.e., $1/mn$ when all the coefficients are 1's).

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Smoothing (Lowpass) Spatial Filter

This is an example

$F[x, y]$ $G[x, y]$

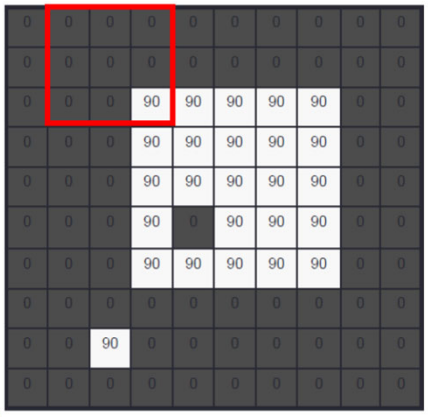


$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

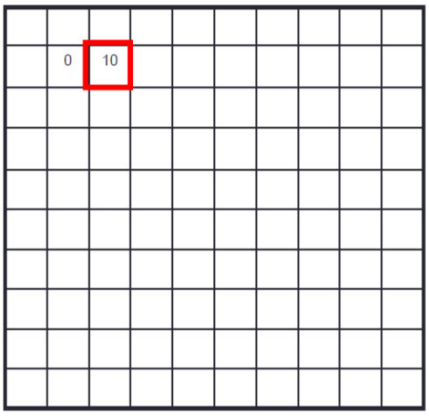
Smoothing (Lowpass) Spatial Filter

This is an example

$F[x, y]$



$G[x, y]$



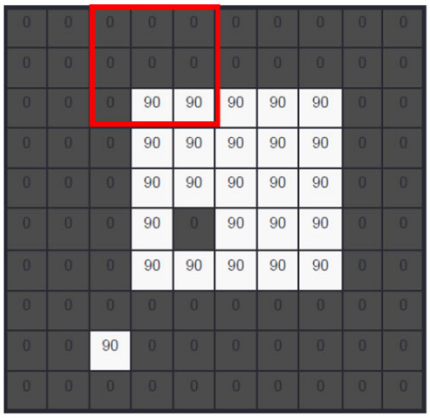
$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

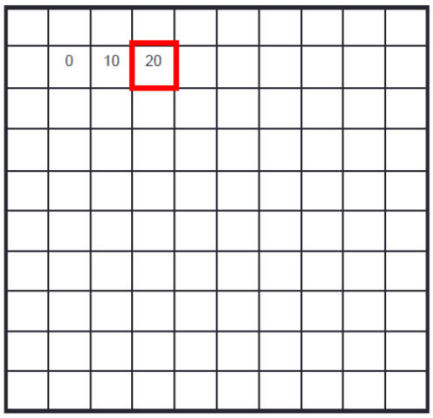
Smoothing (Lowpass) Spatial Filter

This is an example

$F[x, y]$



$G[x, y]$



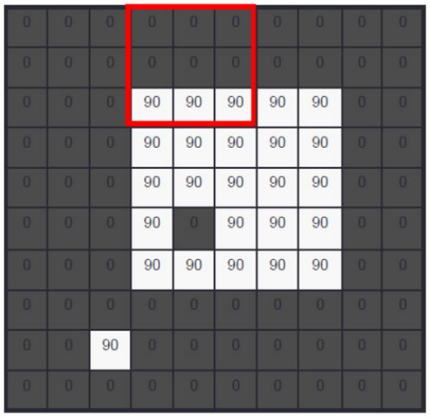
$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

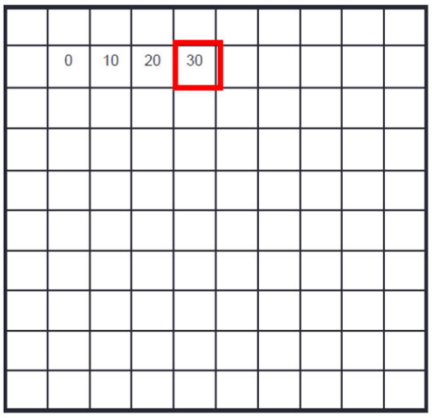
Smoothing (Lowpass) Spatial Filter

This is an example

$F[x, y]$



$G[x, y]$



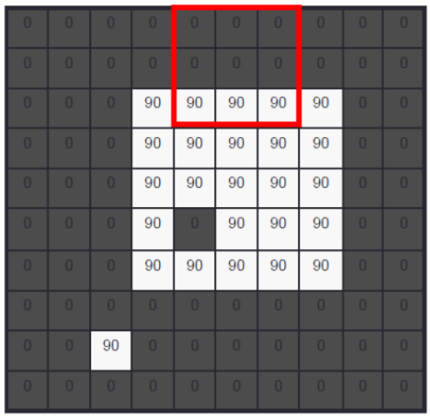
$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

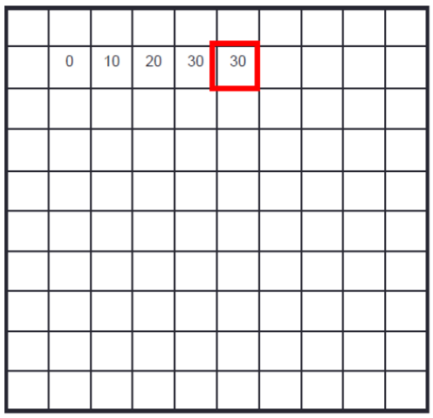
Smoothing (Lowpass) Spatial Filter

This is an example

$F[x, y]$



$G[x, y]$



$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

Smoothing (Lowpass) Spatial Filter

This is an example

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

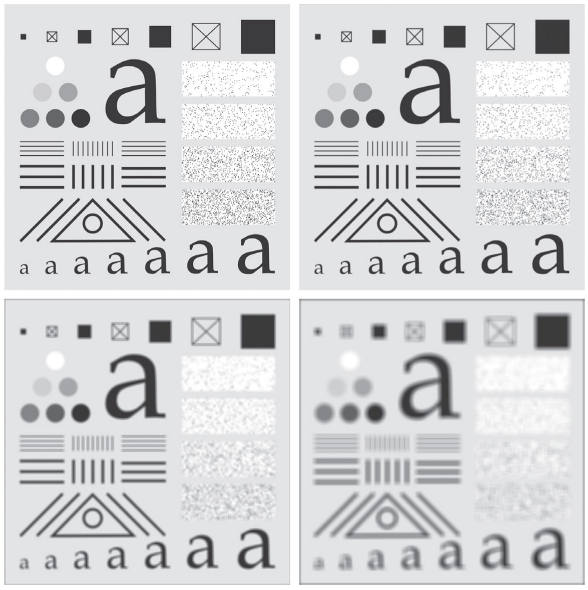
Smoothing (Lowpass) Spatial Filter

box kernel

a b
c d

(a) Test pattern of size 1024×1024 pixels. (b)-(d) Results of lowpass filtering with box kernels of sizes 3×3, 11×11, and 21×21, respectively.

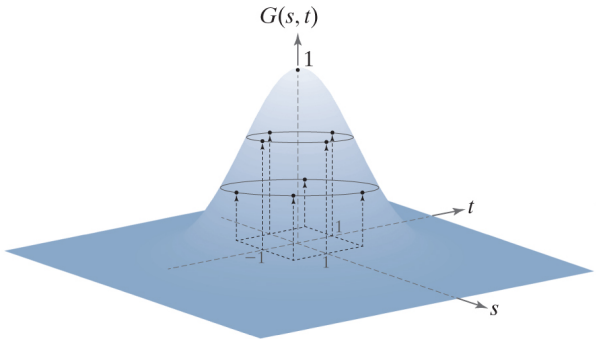
The larger the size, the more the blur.



Gaussian Filter Kernels

Gaussian kernels have the form $w(s,t) = G(s,t) = Ke^{-\frac{s^2+t^2}{2\sigma^2}}$

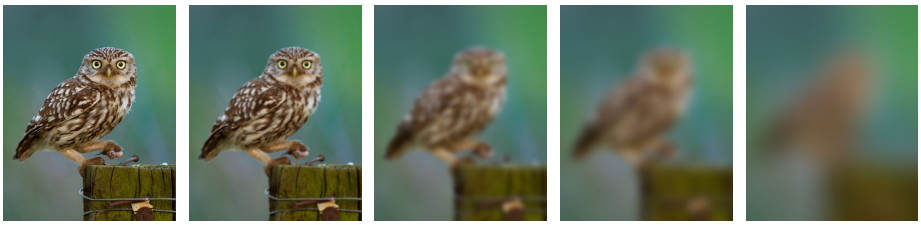
a b



$$\frac{1}{4.8976} \times \begin{bmatrix} 0.3679 & 0.6065 & 0.3679 \\ 0.6065 & 1.0000 & 0.6065 \\ 0.3679 & 0.6065 & 0.3679 \end{bmatrix}$$

(a) Sampling a Gaussian function to obtain a discrete Gaussian kernel. The values shown are for $K = 1$ and $\sigma = 1$. (b) Resulting 3×3 kernel

Gaussian filters



σ = 1 pixel σ = 5 pixels σ = 10 pixels σ = 30 pixels

The larger the standard deviation, the more blur.

Sharpening (Highpass) Spatial Filters

- Sharpening highlights transitions in intensity.
- Sharpening is accomplished by **spatial differentiation** as image differentiation enhances edges and other discontinuities (such as noise) and de-emphasizes areas with slowly varying intensities.

Sharpening (Highpass) Spatial Filters

Foundation:

- Derivatives of a digital function are defined in terms of differences. A basic definition of the first-order derivative of a one-dimensional function $f(x)$ is the difference: $\frac{\partial f}{\partial x} = f(x+1) - f(x)$. Here we use partial derivative notation for consistency later on.
- The second-order derivative of $f(x)$ is :

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x).$$

Using First-Order Derivatives for Image Sharpening—the Gradient

First derivatives in image processing are implemented using the magnitude of the gradient. The gradient of an image *f* at coordinates (*x*, *y*) is defined as the two-dimensional column vector

$$\nabla f = grad(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

This vector has the important geometrical property that *it points in the direction of the greatest rate of change of f at location (x, y)*.

The magntitude (length) of vector ∇f, denoted as $M(x,y)$ or $\|\nabla f\| = \sqrt{g_x^2 + g_y^2}$ is the value at (x,y) of the rate of change in the direction of the gradient vector.

Using First-Order Derivatives for Image Sharpening—the Gradient

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

The simplest approximations to a first-order derivative are

$g_x = (z_8 - z_5) = f(x+1, y) - f(x, y)$ and $g_y = (z_6 - z_5) = f(x, y+1) - f(x, y)$

Another definition, proposed by Roberts, uses cross differences:

$g_x = (z_9 - z_5) = (fx + 1, y + 1) - f(x, y)$ and $g_y = (z_8 - z_6) = f(x+1, y) - f(x, y+1)$

The two kernels are called ***Roberts*** *cross-gradient operators*.

Using First-Order Derivatives for Image Sharpening—the Gradient

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

We prefer to use kernels of odd sizes because they have a unique, (integer) center of spatial symmetry. The smallest kernels in which we are interested are of size 3×3. Approximations to g_x and g_y using a 3×3 neighborhood centered on z_5 are as follows:

$$g_x = \partial f/\partial x = (z_7 + 2z_8 + z_9) - (z_1 + 2\,z_2 + z_3)$$
$$g_y = \partial f/\partial y = (z_3 + 2\,z_6 + z_9) - (z_1 + 2\,z_4 + z_7)$$

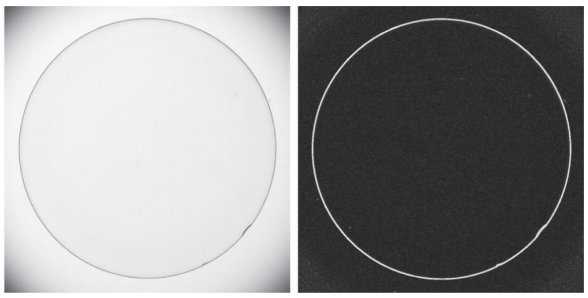
The left two kernels are called **Sobel operators**. The idea behind using a weight value of 2 in the center coefficient is to achieve some smoothing by giving more importance to the center point. The partial derivatives at all points in an image are obtained by convolving the image with these kernels.

https://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm

Robinson

Using First-Order Derivatives for Image Sharpening—the Gradient

Using the gradient for edge enhancement.



(a) Image of a contact lens (b) Sobel gradient.

Robinson

Image Segmentation

Let R represent the image. Image segmentation is a process that partitions R into n subregions, $R_1, R_2, \cdots, R_n$, such that

a. $\bigcup_{i=1}^{n} R_i = R$ --- the segmentation must be complete and every pixel must be in a region

b. R_i is a connected set, for $i = 0,1,2,\ldots,n.$ --- points in a region be connected in some predefined sense (e.g., 8-connected)

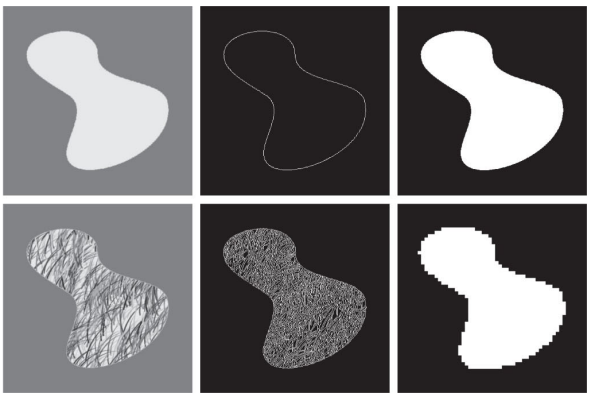
c. $R_i \cap R_j = \emptyset$ for all i and $j, i \neq j$.

d. $Q(R_i)$ =TRUE for $i = 0,1,2,\ldots,n.$ --- Q() is a logical predicate defined over the points in a set and the condition deals with the properties that must be satisfied by the pixels in a segmented region. For example, $Q(R_i)$ =TRUE if all pixels in R_i have the same intensity.

e. $Q(R_i \cup R_j)$ =FALSE for any adjacent regions R_i and R_j.

Image Segmentation

a b c
d e f



(a) Image of a constant intensity region. (b) Boundary based on intensity discontinuities. (c) Result of segmentation. (d) Image of a texture region. (e) Result of intensity discontinuity computations (note the large number of small edges). (f) Result of segmentation based on region properties.

Image Segmentation

- Most of the segmentation algorithms are based on properties of image intensity values:
  - Discontinuity: partition an image into regions based on abrupt changes in intensity, such as edges.
  - Similarity: partition an image into regions that are similar according to a set of predefined criteria. An example is thresholding
- Image segmentation based on active contours and deformable models includes snakes and level sets.
  - Snakes are active contours based on explicit (e.g., parametric) representation of segmentation curves;
  - Level sets are based on implicit representation of curves, which are techniques for representing active contours as ***the intersection of a 3-D surface with a plane.***

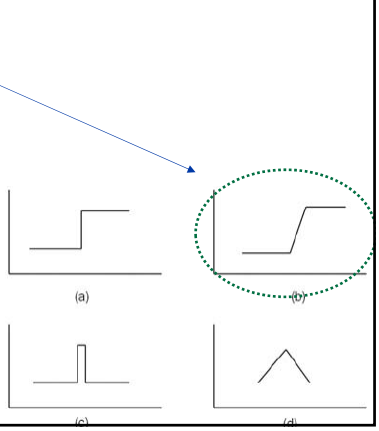
Image Segmentation
Point, Line and Edge Detection

- Detect sharp, local changes in intensity for three types:
  - **Edge pixels:** pixels at which the intensity of an image changes abruptly;
  - **A line**: the intensity of the background on either side of the line is either much higher or much lower than the intensity of the line pixels.
  - **An isolated point:** a foreground (background) pixel surrounded by background (foreground) pixels.

Point, Line and Edge Detection

- Derivatives of a digital function are defined in terms of finite differences;
- We approximate the first order derivative with the following requirements:

- Equations:
  - Forward difference: $\frac{\partial f(x)}{\partial x} = f(x+1) - f(x)$
  - Backward difference: $\frac{\partial f(x)}{\partial x} = f(x) - f(x-1)$
  - Central difference: $\frac{\partial f(x)}{\partial x} = \frac{f(x+1) - f(x-1)}{2}$



Point, Line and Edge Detection

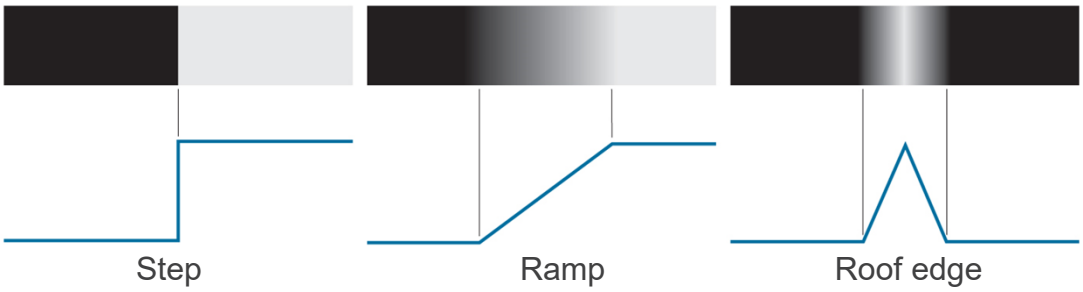
- Derivatives of a digital function are defined in terms of finite differences;
- We approximate the second order derivative with the following requirements:
  - must be zero in areas of constant intensity;
  - must be nonzero at the onset of an intensity step or ramp; and
  - must be zero along intensity ramps.

$$\frac{\partial^2 f(x)}{\partial x^2} = f(x+1) - 2f(x) + f(x-1)$$

$$\frac{\partial^2 f(x,y)}{\partial x^2} = f(x+1,y) - 2f(x,y) + f(x-1,y)$$

$$\frac{\partial^2 f(x,y)}{\partial y^2} = f(x,y+1) - 2f(x,y) + f(x,y-1)$$

Point, Line and Edge Detection

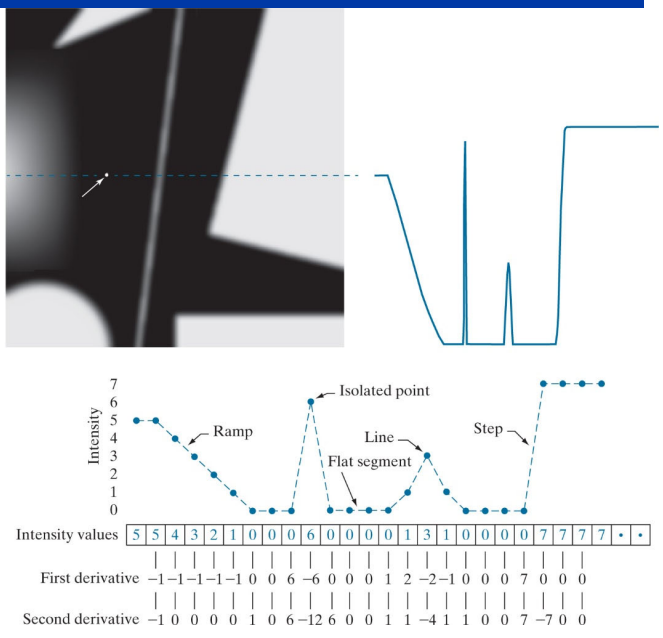


- A **step edge** is characterized by a transition between two intensity levels occurring ideally over the distance of *one pixel*.
- **Ramp**: edges that are blurred and noisy.
- **Roof edges** are models of lines through a region, with the base (width) of the edge being determined by the thickness and sharpness of the line.

Point, Line and Edge Detection

How the first- and second-order derivatives behave as they encounter a point, a line, and the edges of objects.

a b c

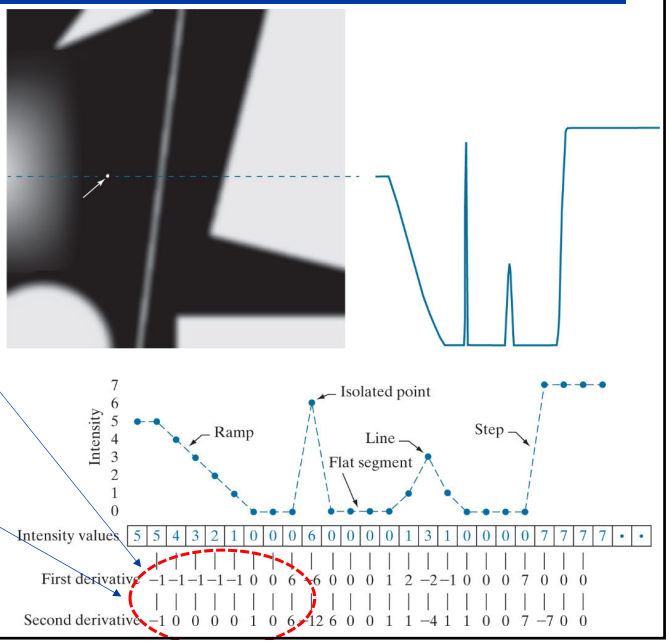


(a) Image.
(b) Horizontal intensity profile that includes the isolated point indicated by the arrow.
(c) Subsampled profile; The numbers in the boxes are the intensity values of the dots shown in the profile.

Point, Line and Edge Detection

- Initially, the first-order derivative is ***nonzero*** at the onset and along the entire intensity ramp, while the second-order derivative is nonzero only at the onset and end of the ramp.
- Because the edges of digital images resemble this type of transition, we conclude that first-order derivatives produce "thick" edges, and second-order derivatives much thinner ones.

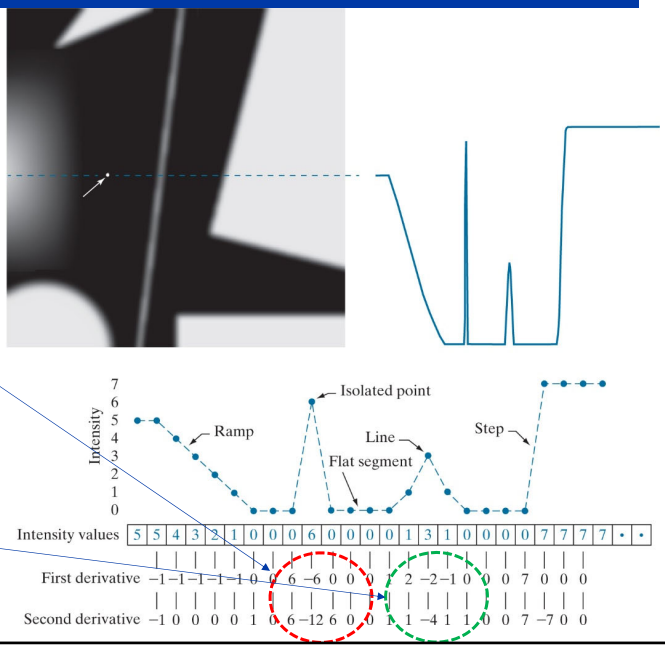
a b c



Point, Line and Edge Detection

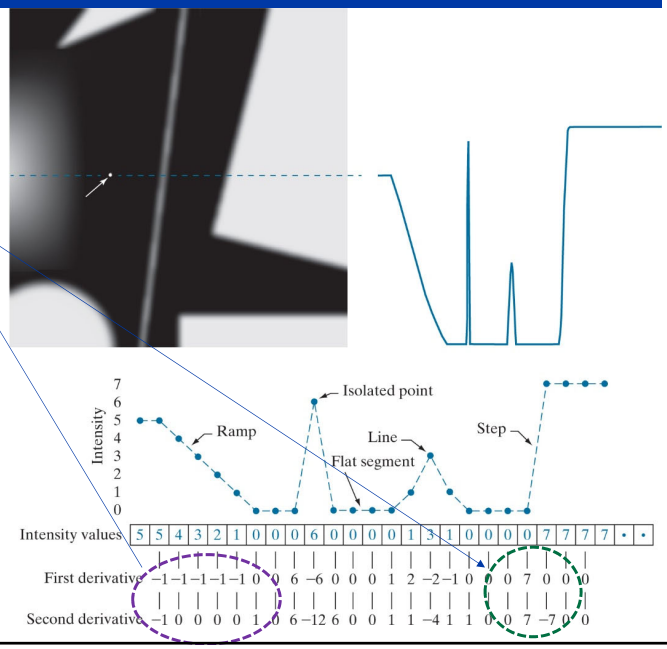
- Next we encounter the isolated noise point. Here, the magnitude of the response at the point is much stronger for the second- than for the first-order derivative.
- A second-order derivative is much more aggressive than a first-order derivative in enhancing sharp changes.
- Thus, we can expect second-order derivatives to enhance fine detail (including noise) much more than first-order derivatives.
- The line is rather thin, so it too is fine detail, and we see again that the second derivative has a larger magnitude.

a b c



Point, Line and Edge Detection

- Note in both the ramp and step edges that the second derivative has opposite signs (negative to positive or positive to negative) as it transitions into and out of an edge.
- **This "double-edge" effect is an important characteristic that is used to locate edges.**
- As we move into the edge, the sign of the second derivative is used also to determine whether an edge is a transition from light to dark, or from dark to light.



Point, Line and Edge Detection

The approach to compute first and second derivatives at every pixel location in an image is to use spatial convolution.

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

$$z = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{k=1}^{9} w_k z_k$$

z_k is the intensity of the pixel whose spatial location corresponds to the location of the kth kernel coefficient.

Detection of Isolated Points

A point is detected at a location (x, y) on which the kernel is centered if the absolute value of the response of the filter at that point *exceeds a specified threshold*. Such points are labeled 1 and all others are labeled 0 in the output image, thus producing a binary image. In other words, we use the expression:

$$g(x,y) = \begin{cases} 1 & if\ |Z(x,y)| > T \\ 0 & \text{otherwise} \end{cases}$$

where $g(x, y)$ is the output image, T is a nonnegative threshold, and Z is filtered value. This formulation simply measures the weighted differences between a pixel and its 8-neighbors.

$$z = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{k=1}^{9} w_k z_k$$

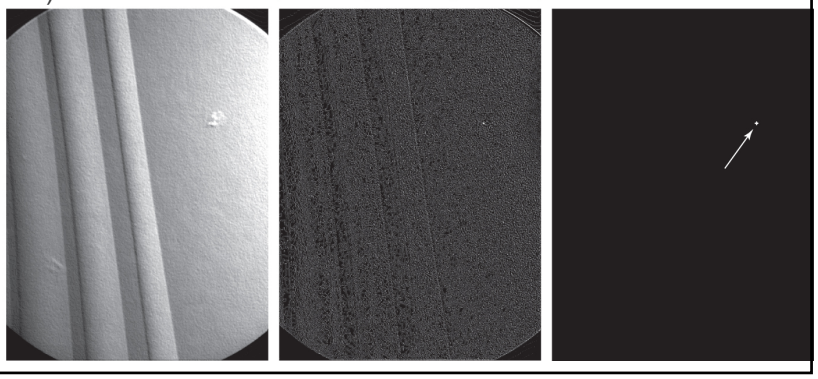
Detection of Isolated Points

a
b c d

1	1	1
1	−8	1
1	1	1

(a) Laplacian kernel used for point detection.
(b) Original image
(c) Result of convolving the kernel with the image.
(d) Result of using the equation was a single point (shown enlarged at the tip of the arrow).

$$g(x,y) = \begin{cases} 1 & if\ |Z(x,y)| > T \\ 0 & \text{otherwise} \end{cases}$$



Line Detection

Second derivatives to result in a stronger filter response, and to produce thinner lines than first derivatives. Thus, we can use the Laplacian kernel.

−1	−1	−1	2	−1	−1	−1	2	−1	−1	−1	2
2	2	2	−1	2	−1	−1	2	−1	−1	2	−1
−1	−1	−1	−1	−1	2	−1	2	−1	2	−1	−1
Horizontal			+45°			Vertical			−45°		

a b c d

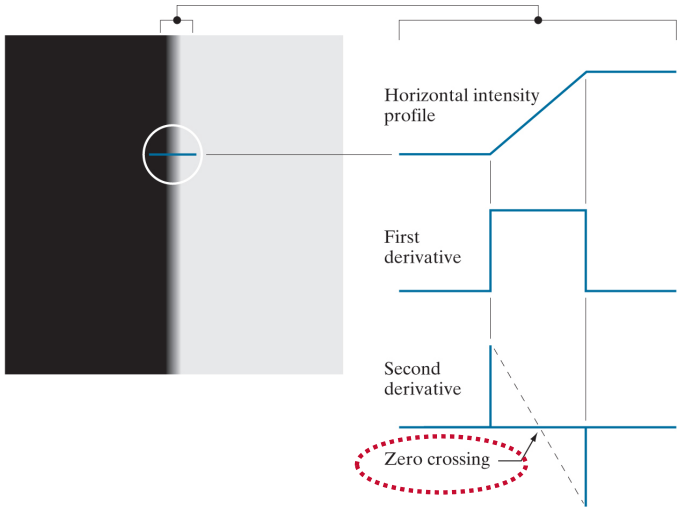
Line detection kernels. Detection angles are counterclockwise with respect to the (vertical) x-axis.

- If we are interested in detecting all the lines in an image in the direction defined by a given kernel, we simply run the kernel through the image and *threshold the absolute value of the result*.
- The nonzero points remaining after thresholding are the strongest responses which, for lines one pixel thick, correspond closest to the direction defined by the kernel.

Edge Detection

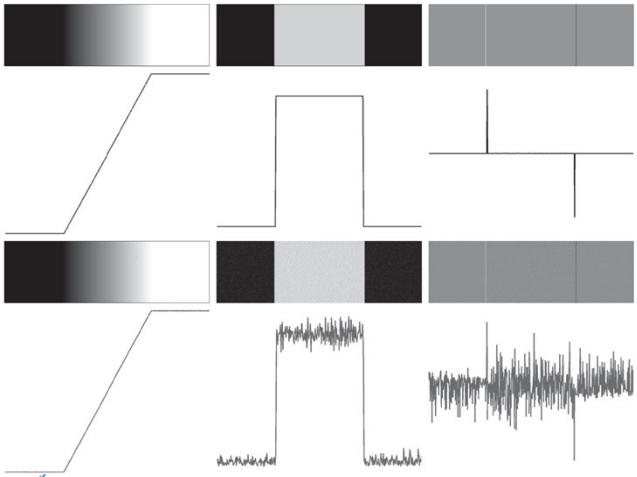
Edge detection is an approach used frequently for segmenting images based on abrupt (local) changes in intensity

a b



Edge Detection

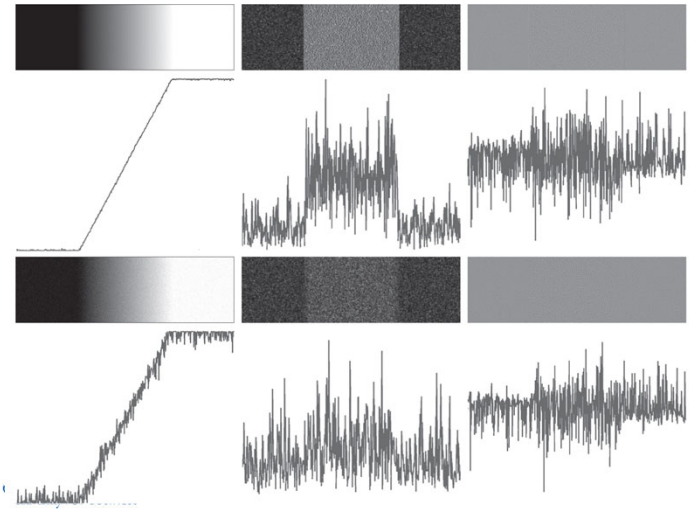
Behavior of the first and second derivatives in the region of a noisy edge.



- First column shows the ramp edges from black to white; The second column is first order derivative; the third column is the second order derivative;
- The top one does not have noise; The second one has additive Gaussian noise with zero mean and standard deviation of 0.1.

Edge Detection

Behavior of the first and second derivatives in the region of a noisy edge.



Additive Gaussian noise with zero mean and standard deviation of 1.0 and 10.0.

Observation:?

Key observation is that the noise is almost visually undetectable in the images on the left column, but the first and second derivatives are very sensitive to these noises, and noise has high impact.

Edge Detection

- Because noise has high impact on first and second derivatives, image smoothing should be considered.
- In general, there are three steps that are typical for edge detection:
  - Image smoothing for noise reduction.
  - Detection of edge points: this is a local operation that extracts from an image all points that are potential edge-point candidates.
  - Edge localization. The objective of this step is to select from the candidate points only the points that are members of the set of points comprising an edge. Linking pixels that have similar magnitude and similar direction.

The Robert Edge Detector

Algorithm Robert Edge Detector:

Input: Image I, and a threshold τ

1. Apply noise smoothing as appropriate (e.g., Gaussian smoothing), obtaining a new image I_s
2. Filter I_s with the masks

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

   Obtaining two images I_1 and I_2
3. Estimate the gradient magnitude at each pixel (i,j) as

$$G(i,j) = \sqrt{I_1^2(i,j) + I_2^2(i,j)}$$

   Obtaining an image of magnitude gradients, G;
4. Mark as edges all pixels (i,j) such that $G(i,j) > \tau$

Output: the location of edge points obtained in the last step

The Sobel Edge Detector

Algorithm Robert Edge Detector:

Input: Image I, and a threshold τ

1. Apply noise smoothing as appropriate (e.g., Gaussian smoothing), obtaining a new image I_s
2. Filter I_s with the masks

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

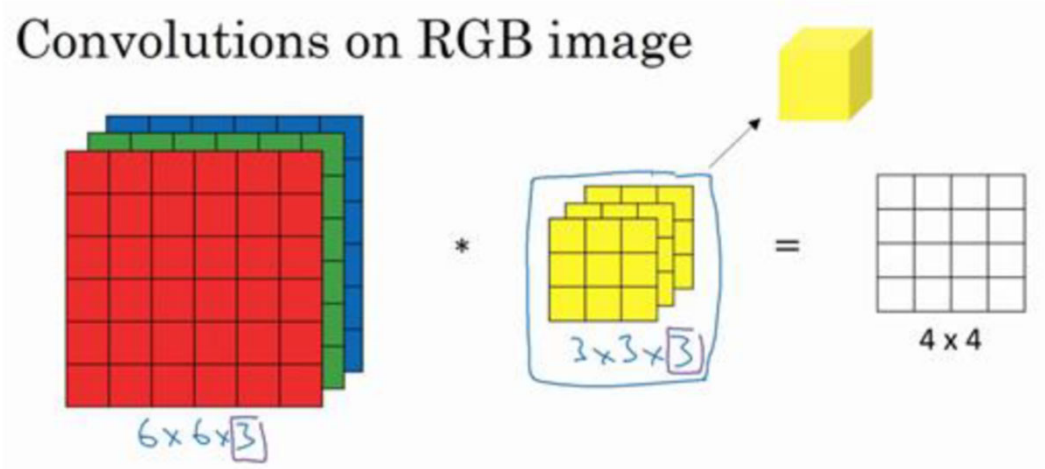
   Obtaining two images I_1 and I_2
3. Estimate the gradient magnitude at each pixel (i,j) as

$$G(i,j) = \sqrt{I_1^2(i,j) + I_2^2(i,j)}$$

   Obtaining an image of magnitude gradients, G;
4. Mark as edges all pixels (i,j) such that $G(i,j) > \tau$

Output: the location of edge points obtained in the last step

Convolution for Color Images



Question: why 4×4 size image?

Convolution for Color Images

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	
...	...	...	...	...	...	...

Input channel #1 (red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	
...	...	...	...	...	...	...

Input channel #1 (green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	
...	...	...	...	...	...	...

Input channel #1 (blue)

-1	-1	1
0	1	-1
0	1	1

Kernel #1

1	0	0
1	-1	-1
1	0	-1

Kernel #2

0	1	1
0	1	0
1	-1	1

Kernel #3

http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html

Convolution for Color Images

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	
...	...	...	...	...	...	...

Input channel #1 (red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	
...	...	...	...	...	...	...

Input channel #1 (green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	
...	...	...	...	...	...	...

Input channel #1 (blue)

-1	-1	1
0	1	-1
0	1	1

Kernel #1

308

1	0	0
1	-1	-1
1	0	-1

Kernel #2

-498

0	1	1
0	1	0
1	-1	1

Kernel #3

164

http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html

Convolution for Color Images

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	
...	...	...	...	...	...	...

Input channel #1 (red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	
...	...	...	...	...	...	...

Input channel #1 (green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	
...	...	...	...	...	...	...

Input channel #1 (blue)

-1	-1	1
0	1	-1
0	1	1

Kernel #1

1	0	0
1	-1	-1
1	0	-1

Kernel #2

0	1	1
0	1	0
1	-1	1

Kernel #3

310

Georgia State University J. Mack Robinson College of Business

93

http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html

Convolution for Color Images

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	
...	...	...	...	...	...	...

Input channel #1 (red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	
...	...	...	...	...	...	...

Input channel #1 (green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	
...	...	...	...	...	...	...

Input channel #1 (blue)

-1	-1	1
0	1	-1
0	1	1

Kernel #1

1	0	0
1	-1	-1
1	0	-1

Kernel #2

0	1	1
0	1	0
1	-1	1

Kernel #3

310

-170

325

Georgia State University J. Mack Robinson College of Business

94

Convolution for Color Images

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	
...	...	...	...	...	...	...

Input channel #1 (red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	
...	...	...	...	...	...	...

Input channel #1 (green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	
...	...	...	...	...	...	...

Input channel #1 (blue)

-1	-1	1
0	1	-1
0	1	1

Kernel #1

314

1	0	0
1	-1	-1
1	0	-1

Kernel #2

-175

0	1	1
0	1	0
1	-1	1

Kernel #3

326

http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html

Convolution for Color Images

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	
...	...	...	...	...	...	...

Input channel #1 (red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	
...	...	...	...	...	...	...

Input channel #1 (green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	
...	...	...	...	...	...	...

Input channel #1 (blue)

-1	-1	1
0	1	-1
0	1	1

Kernel #1

298

1	0	0
1	-1	-1
1	0	-1

Kernel #2

-491

0	1	1
0	1	0
1	-1	1

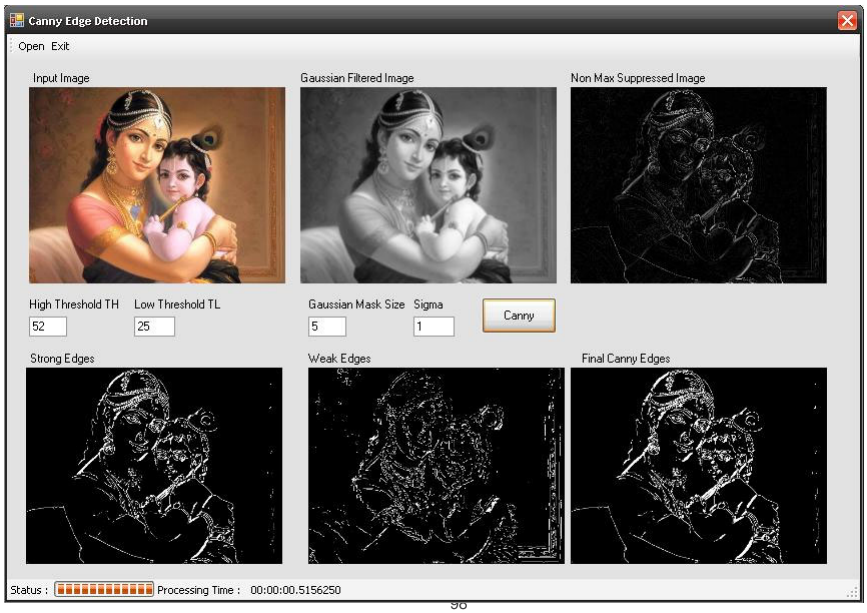
Kernel #3

487

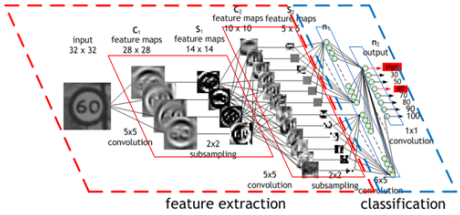
http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html

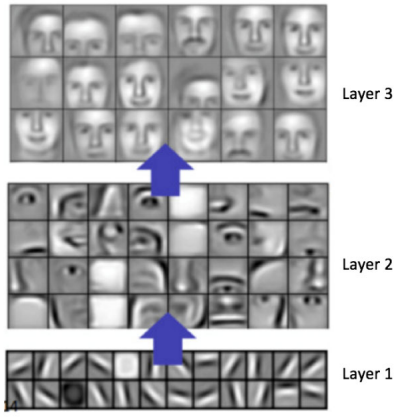
Derivative of the Image

Derivative of the Image



Feature Extraction in CNN





OpenCV

https://docs.opencv.org/4.1.1/