



Image processing operators (2)

Bing Gong
(巩冰)

gongbing@shnu.edu.cn



Objectives

- Know the standard image processing operators methods that map pixel values from one image to another.
- Understand the point and linear operators for image processing

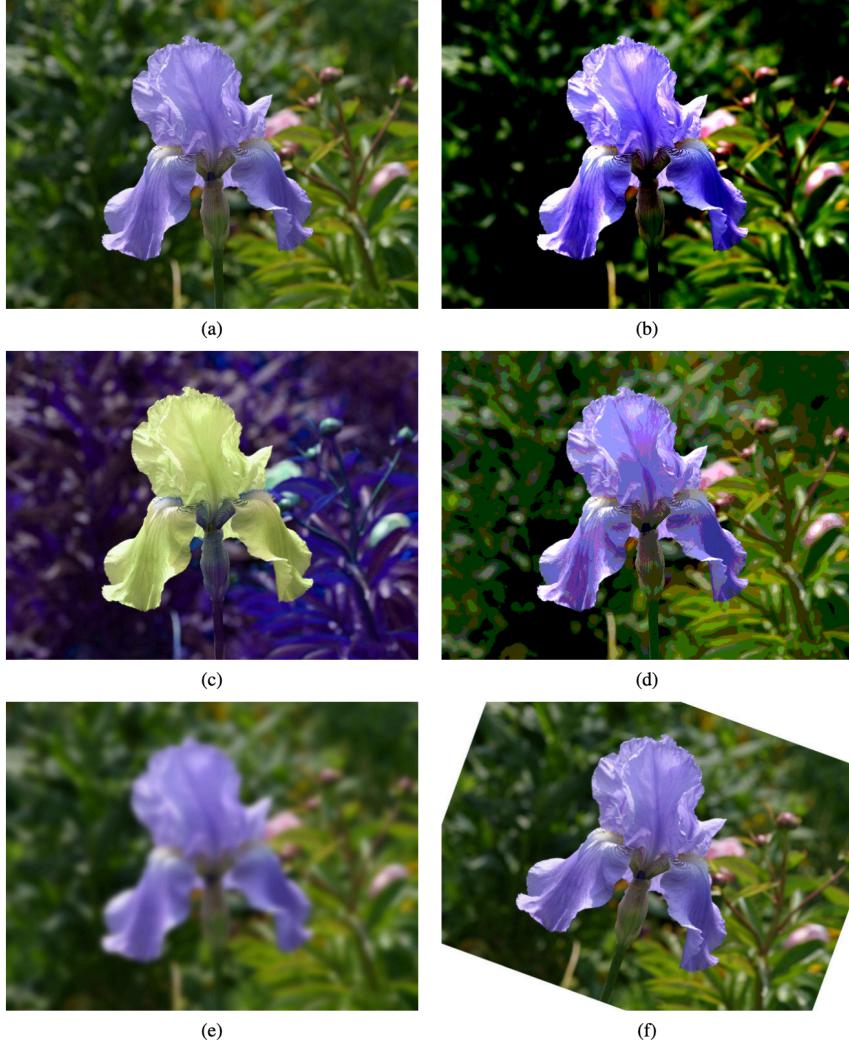


Figure 3.1 Some common image processing operations: (a) original image; (b) increased contrast; (c) change in hue; (d) “posterized” (quantized colors); (e) blurred; (f) rotated.

Why?

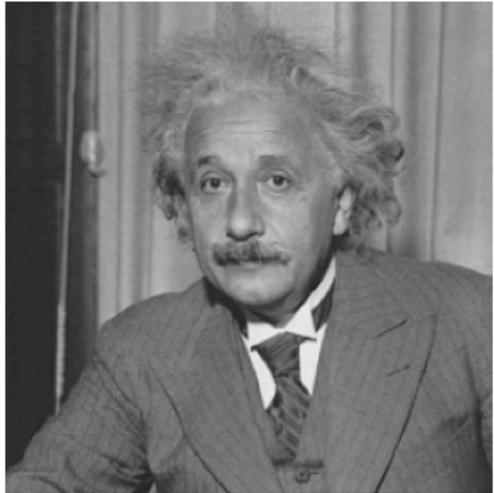


Why?



是啊 就是啊

Why?



(a)



(b)



(c)

Methods of image transform



- *point operators* or *point processes*: the simplest kind of image transforms, namely those that manipulate each pixel independently of its neighbors.
- *neighborhood (area-based) operators*: each new pixel's value depends on a small number of neighboring input values.
- *global operators*: *geometric transformations*, such as rotations, shears, and perspective deformations.
- *global optimization*: involve the minimization of an energy functional or, equivalently, optimal estimation using Bayesian *Markov random field* models.

Methods of image transform



- ***point operators or point processes:*** the simplest kind of image transforms, namely those that manipulate each pixel independently of its neighbors.
- ***neighborhood (area-based) operators:*** each new pixel's value depends on a small number of neighboring input values.
- ***global operators:*** *geometric transformations*, such as rotations, shears, and perspective deformations.
- ***global optimization:*** involve the minimization of an energy functional or, equivalently, optimal estimation using Bayesian *Markov random field* models.

Point operators

- Brightness and contrast adjustments as well as color correction and transformations

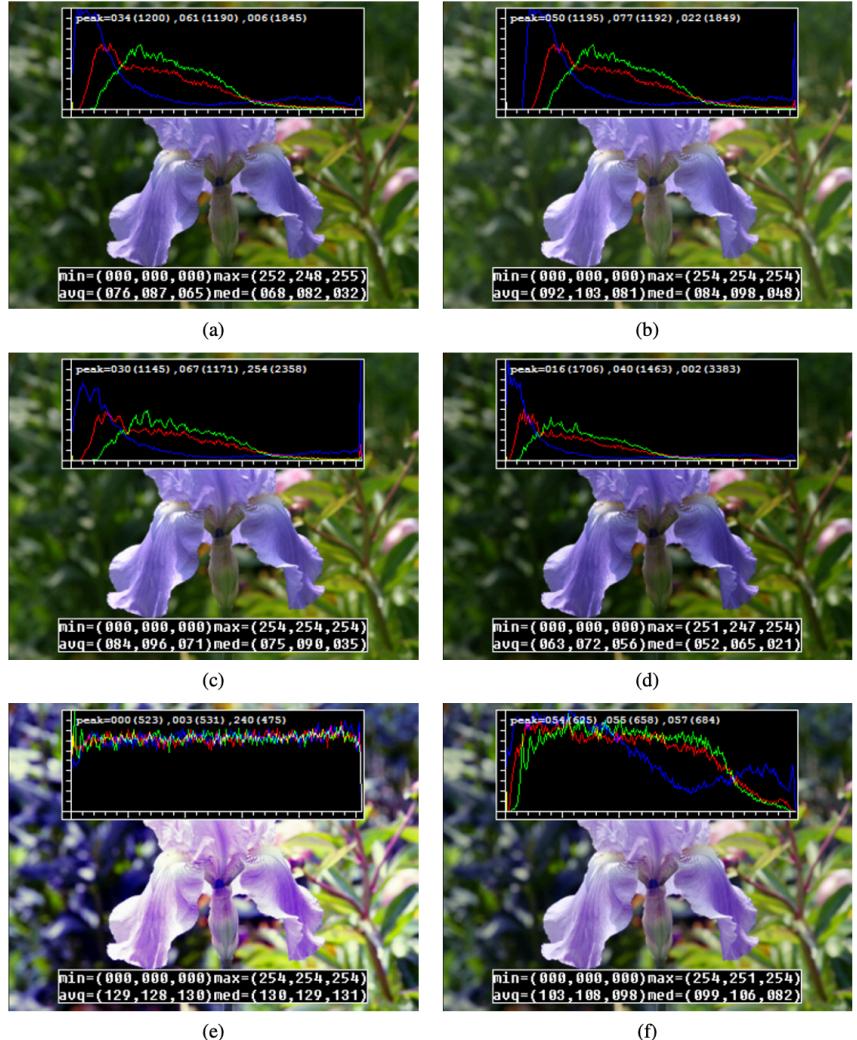


Figure 3.2 Some local image processing operations: (a) original image along with its three color (per-channel) histograms; (b) brightness increased (additive offset, $b = 16$); (c) contrast increased (multiplicative gain, $a = 1.1$); (d) gamma (partially) linearized ($\gamma = 1.2$); (e) full histogram equalization; (f) partial histogram equalization.

Point operators



- **Pixel transforms:**

$$g(x) = h(f(x)) \text{ or } g(x) = h(f_0(x), \dots, f_n(x))$$

Where x is in the D-dimensional *domain* of the functions (usually D = 2 for images) and the functions f and g operate over some *range*, which can either be scalar or vector-valued, e.g., for color images or 2D motion.



Point operators

- Pixel transforms:

$$g(x) = af(x) + b$$

Two commonly used point processes are multiplication and addition with a constant, The parameters $a > 0$ and b are often called the ***gain*** and ***bias*** parameters; sometimes these parameters are said to control ***contrast*** and ***brightness***, respectively.

Point operators



- **Pixel transforms:**

The bias and gain parameters can also be spatially varying

$$g(x) = a(x)f(x) + b(x)$$

Point operators



- **Pixel transforms:**

dyadic (two-input) operator is the linear blend operator:

$$g(x) = (1 - \alpha)f_0(x) + \alpha f_1(x).$$

This operator can be used to perform a temporal *cross-dissolve* between two images or videos, as seen in slide shows and film production, or as a component of image *morphing* algorithms

Point operators

- Pixel transforms:

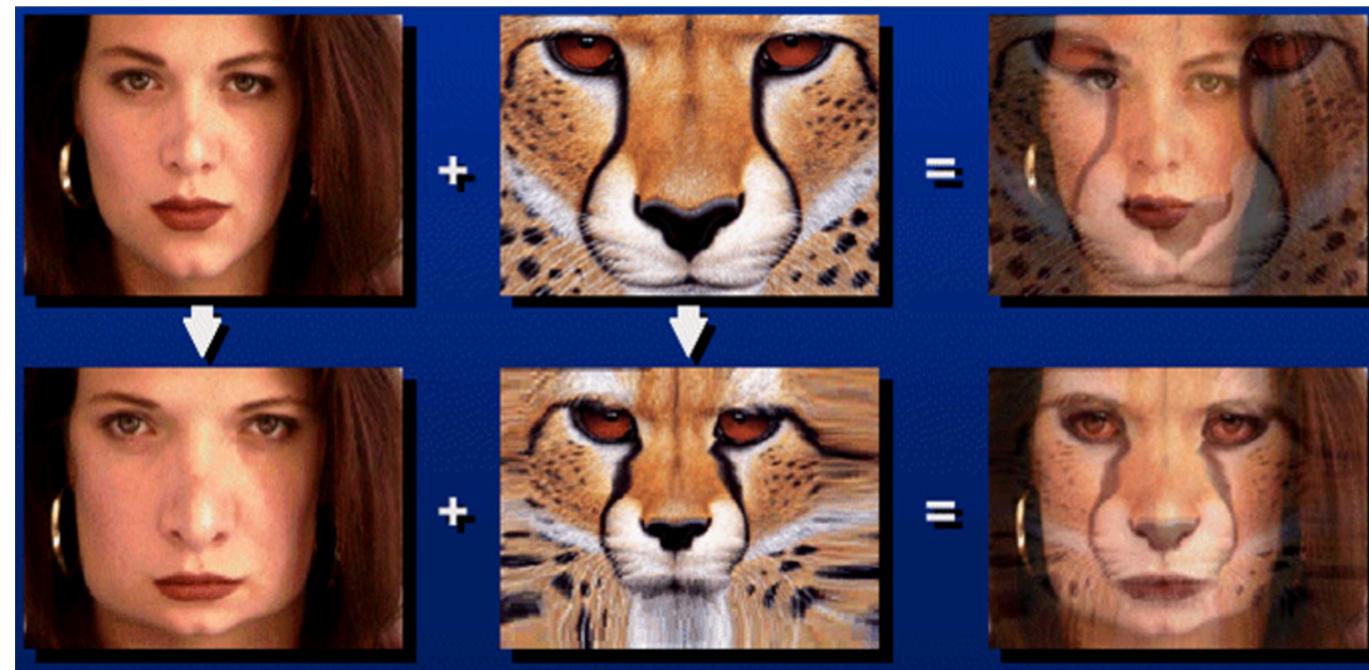
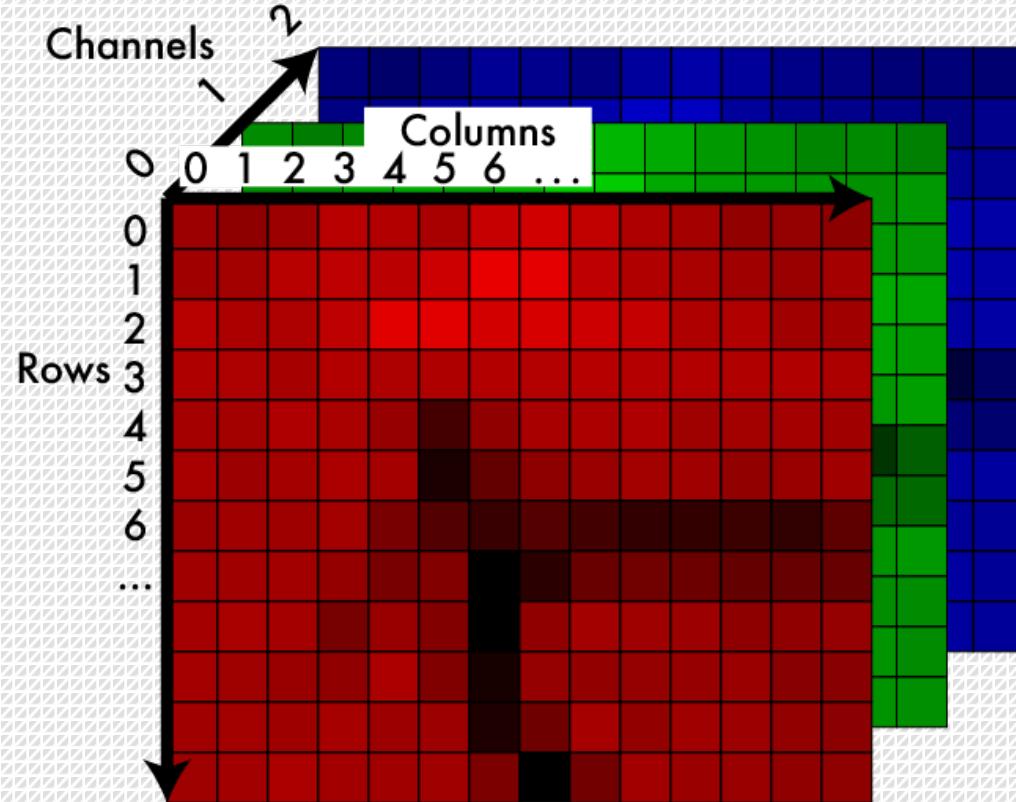
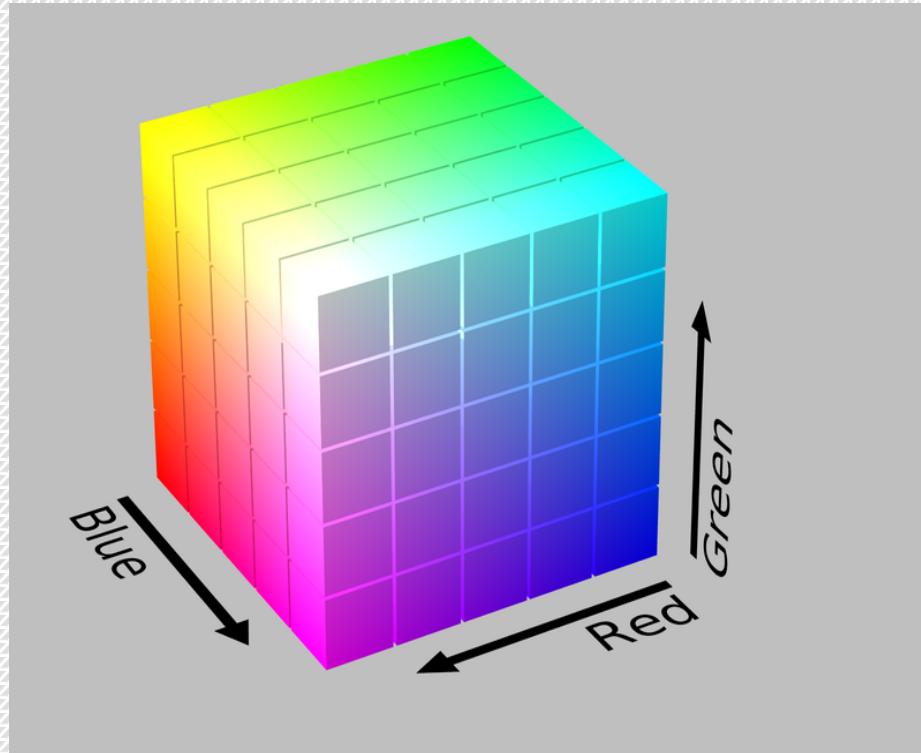


Figure 3.53 Image morphing (Gomes, Darsa, Costa *et al.* 1999) © 1999 Morgan Kaufmann. Top row: if the two images are just blended, visible ghosting results. Bottom row: both images are first warped to the same intermediate location (e.g., halfway towards the other image) and the resulting warped images are then blended resulting in a seamless morph.

Point operators

- Color transforms: RGB is a cube...

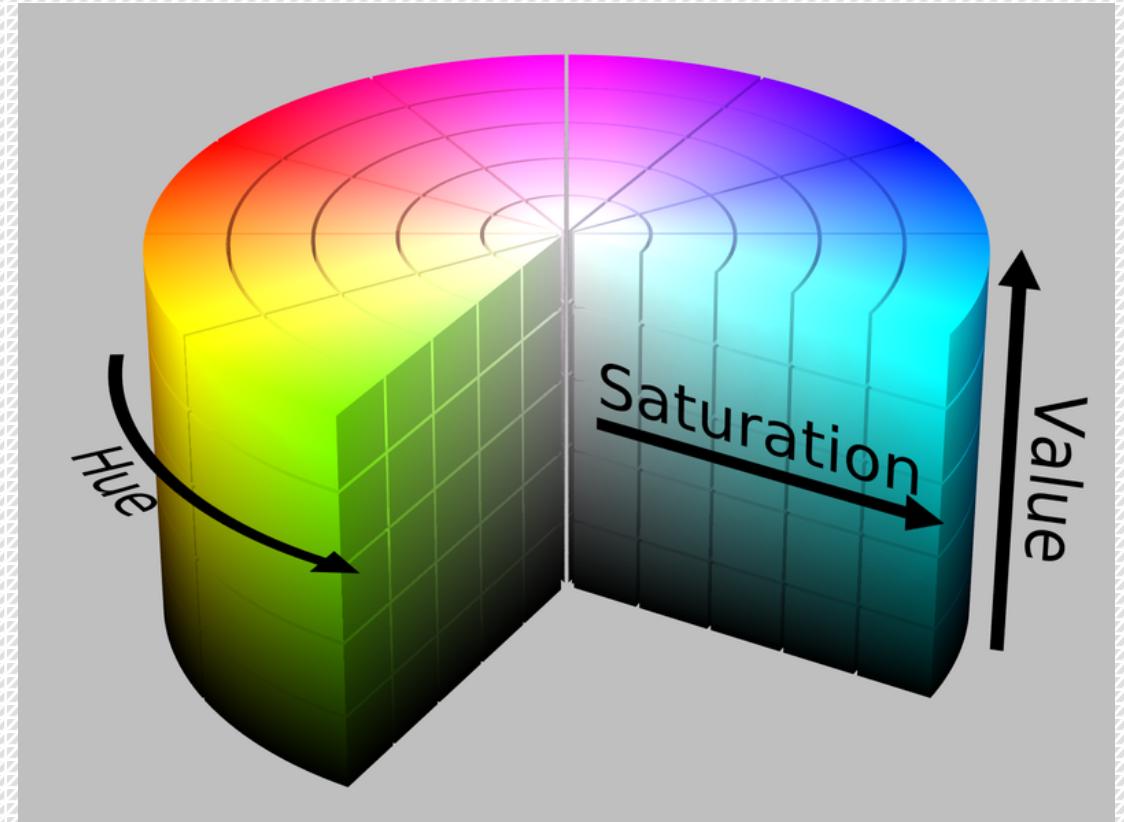


Point operators



- **Color transforms:** Hue, Saturation, Value: cylinder!

- Different model based on perception of light
- **Hue:** what color
- **Saturation:** how much color
- **Value:** how bright
- Allows easy image transforms
 - Shift the hue
 - Increase saturation



Point operators

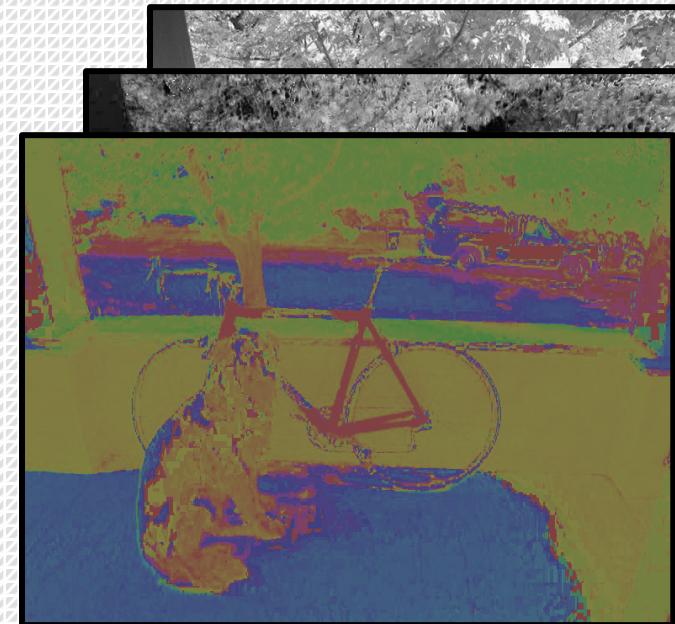
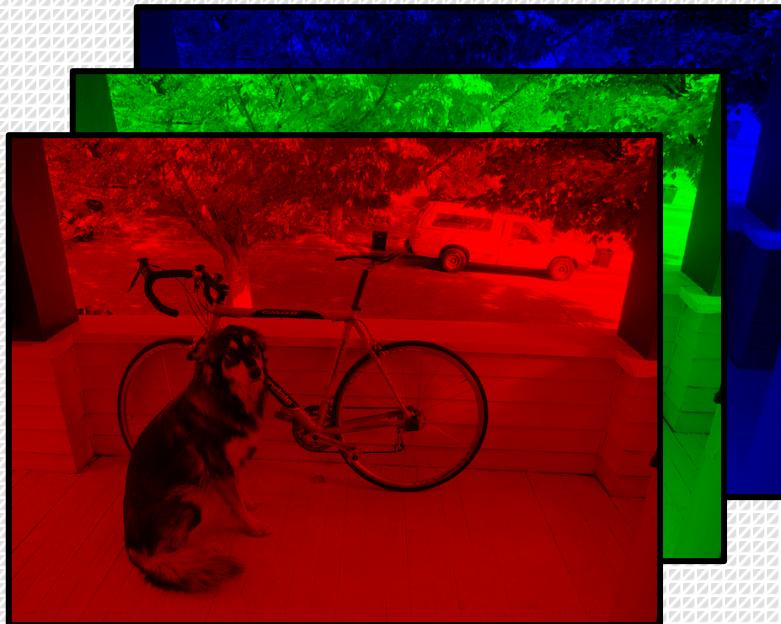


- **Color transforms:** Hue, Saturation, Value: cylinder!

- Different model based on perception of light
- **Hue:** what color
- **Saturation:** how much color
- **Value:** how bright
- Allows easy image transforms
 - Shift the hue
 - Increase saturation

Point operators

- **Color transforms:** Still 3d tensor, different info



Point operators

- Color transforms:

Hue



Saturation



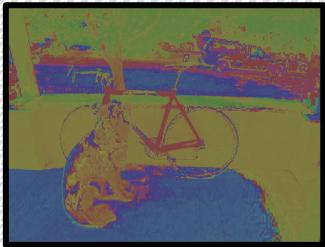
Value



Point operators

- **Color transforms:** More saturation = intense colors

H Channel



S Channel



V Channel



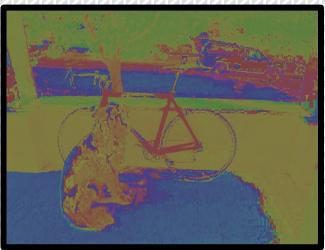
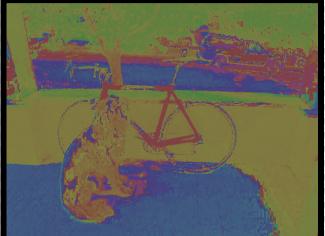
2x



Point operators

- **Color transforms:** More value = lighter image

H Channel



S Channel



V Channel



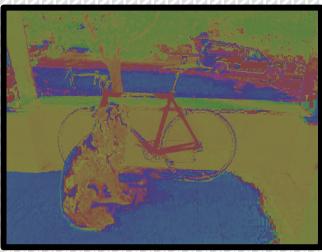
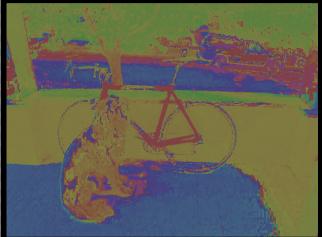
2x



Point operators

- **Color transforms:** Shift hue = shift colors

H Channel



S Channel



V Channel



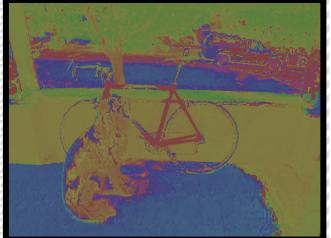
2x



Point operators

- Color transforms: Shift hue = shift colors

H Channel



- .2

S Channel



V Channel



Point operators



- More Details on Color Spaces
 - RGB standard for cameras
 - HSI/HSV allows us to separate
 - CIE L*a*b intensity plus 2 color channels
 - YIQ color TVs, Y is intensity
 - Opponent used in Swain & Ballard work



Point operators

- **Compositing and matting :**

In many photo editing and visual effects applications, it is often desirable to cut a *foreground* object out of one scene and put it on top of a different *background*. The process of extracting the object from the original image is often called *matting*, while the process of inserting it into another image (without visible artifacts) is called *compositing*.

Point operators

- Compositing and matting :



Figure 3.4 Image matting and compositing (Chuang, Curless, Salesin *et al.* 2001) © 2001 IEEE: (a) source image; (b) extracted foreground object F ; (c) alpha matte α shown in grayscale; (d) new composite C .

$$\begin{array}{ccccc}
 \begin{matrix} \text{(a)} \\ \text{B} \end{matrix} & \times & \begin{matrix} \text{(b)} \\ \text{\alpha} \end{matrix} & + & \begin{matrix} \text{(c)} \\ \alpha F \end{matrix} = \begin{matrix} \text{(d)} \\ \text{C} \end{matrix}
 \end{array}$$

Figure 3.5 Compositing equation $C = (1 - \alpha)B + \alpha F$. The images are taken from a close-up of the region of the hair in the upper right part of the lion in Figure 3.4.

Point operators



- Compositing and matting :

- Over operator:

$$C = (1 - \alpha)B + \alpha F$$

This operator *attenuates* the influence of the background image B by a factor $(1 - \alpha)$ and then adds in the color (and opacity) values corresponding to the foreground layer F

Point operators



- **Histogram equalization:**

While the brightness and gain controls described above can improve the appearance of an image, how can we automatically determine their best values?

Point operators



- What is histogram equalization?

Perform *histogram equalization* to find an intensity mapping function $f(I)$ such that the resulting histogram is flat.

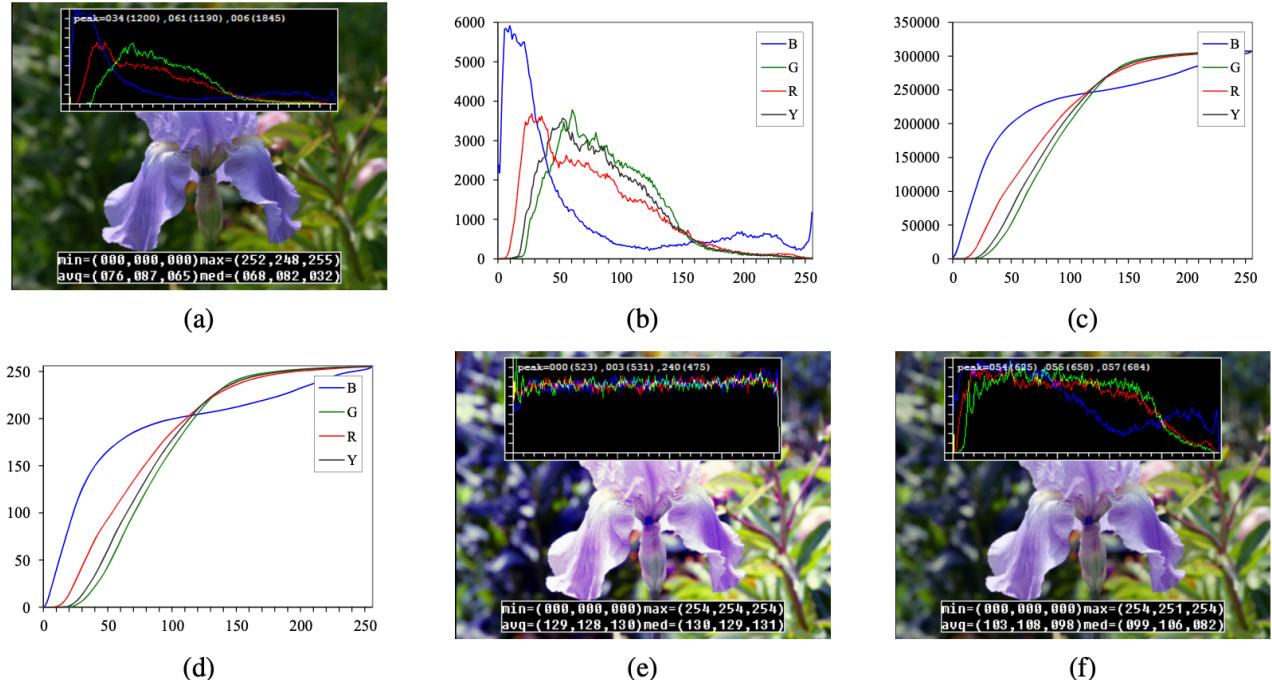
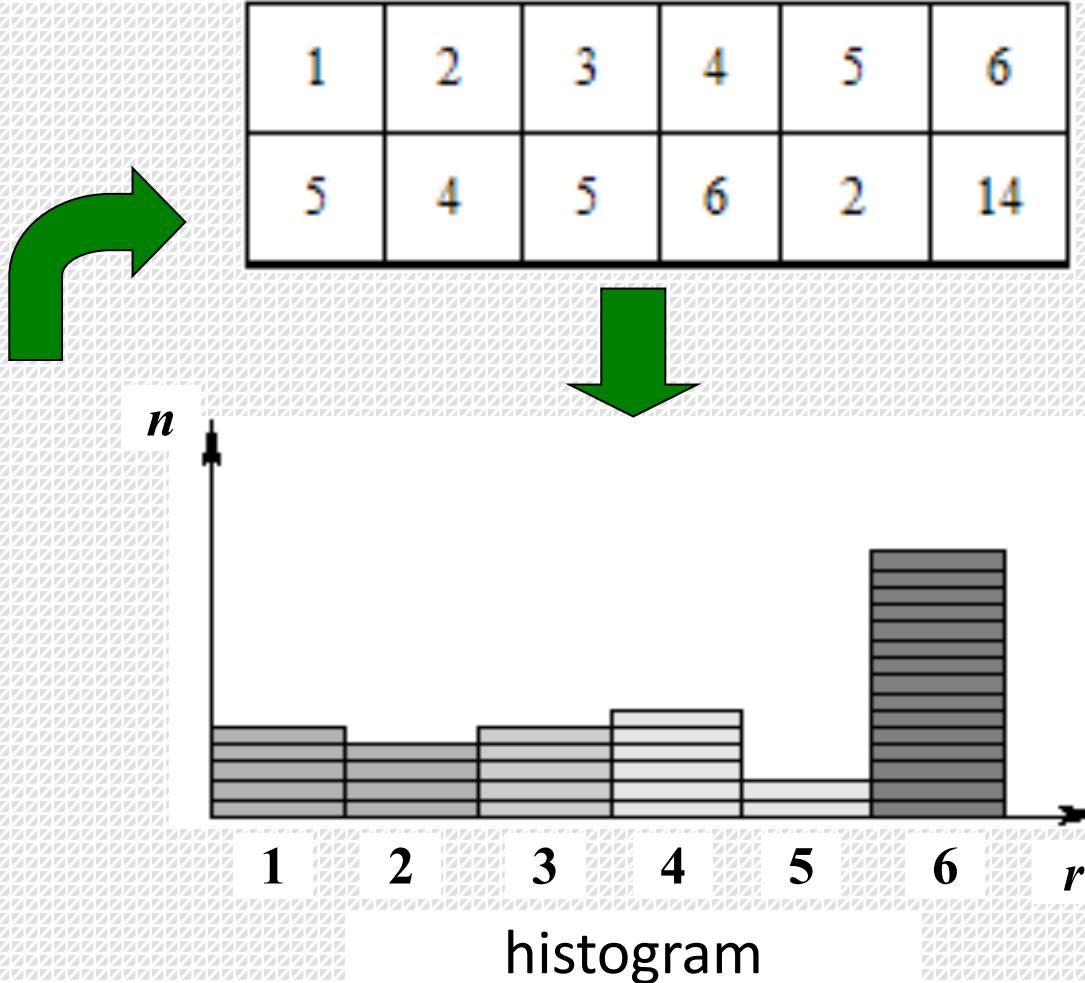


Figure 3.7 Histogram analysis and equalization: (a) original image (b) color channel and intensity (luminance) histograms; (c) cumulative distribution functions; (d) equalization (transfer) functions; (e) full histogram equalization; (f) partial histogram equalization.

Point operators

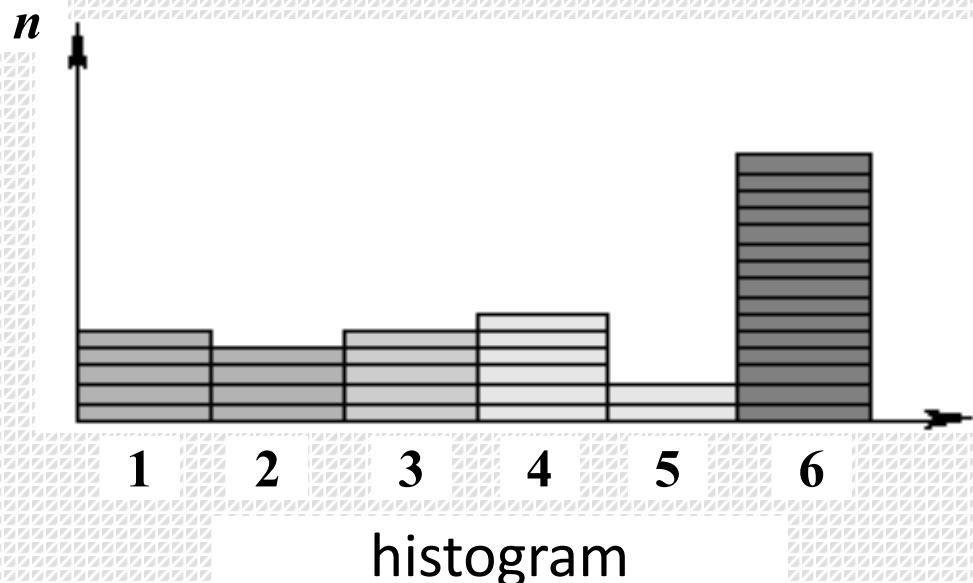
- What is histogram:

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |
| 6 | 4 | 3 | 2 | 2 | 1 |
| 1 | 6 | 6 | 4 | 6 | 6 |
| 3 | 4 | 5 | 6 | 6 | 6 |
| 1 | 4 | 6 | 6 | 2 | 3 |
| 1 | 3 | 6 | 4 | 6 | 6 |



Point operators

- What is histogram:



The histogram provides a visual representation of the distribution of the data, showing the number of observations that fall within each bin.

Point operators

- What is density:

$$P_r(r_k) = \frac{n_k}{n} \quad 0 \leq r_k \leq 1$$

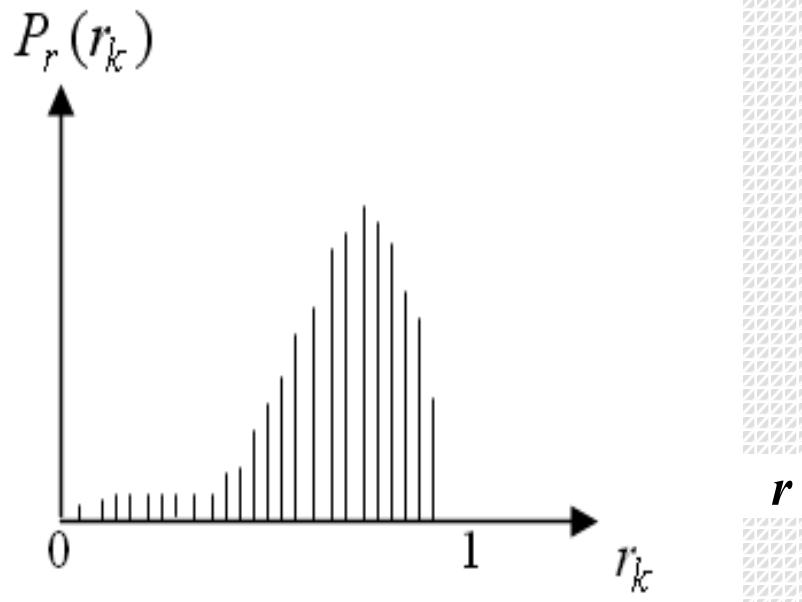
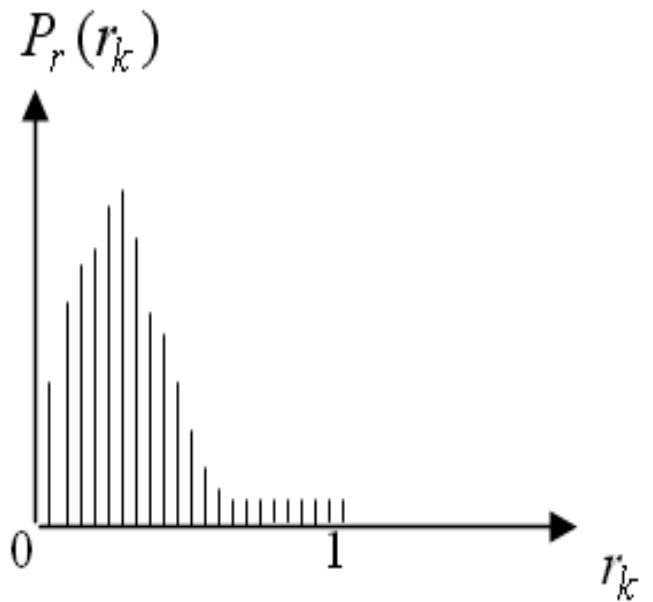
$$k = 0, 1, 2, \dots, l-1$$

- $r=0$ represents 黑 ,
- $r=1$ represents 白。



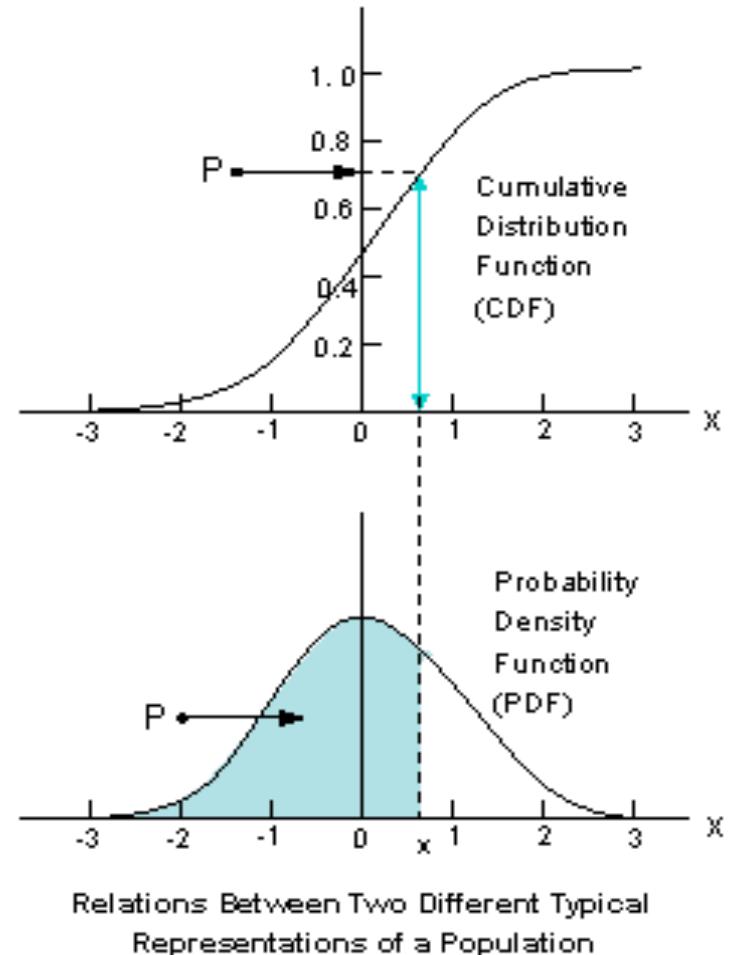
Point operators

- What is density:

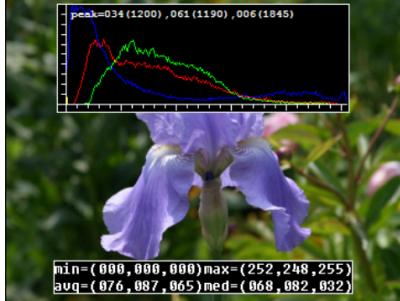


Point operators

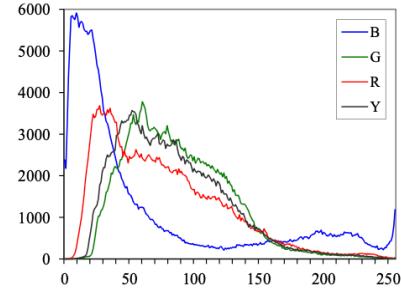
- What is cumulative distribution function :



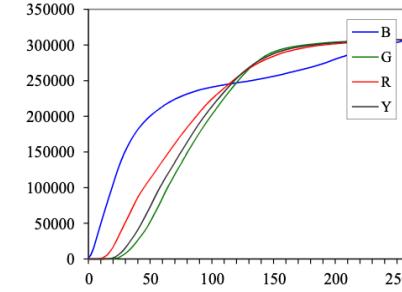
Point operators



(a)



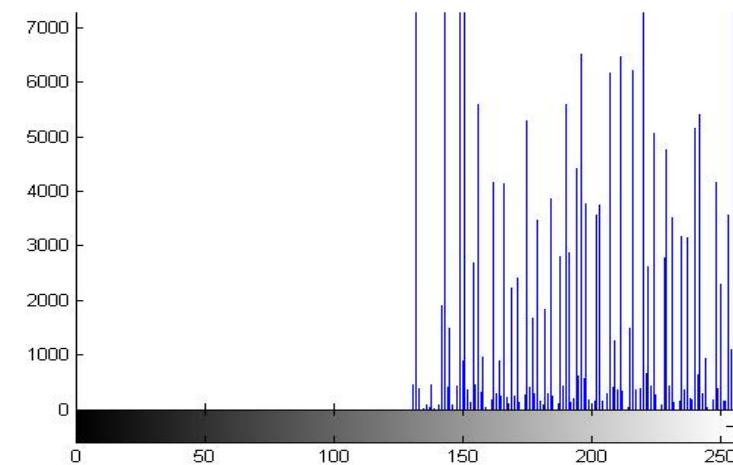
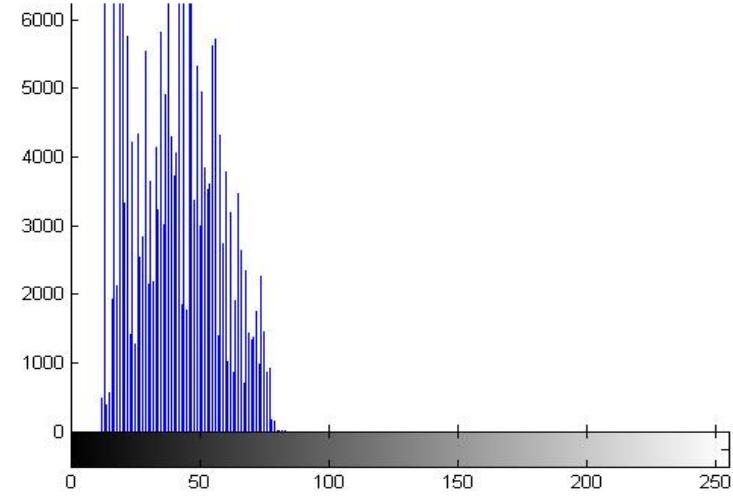
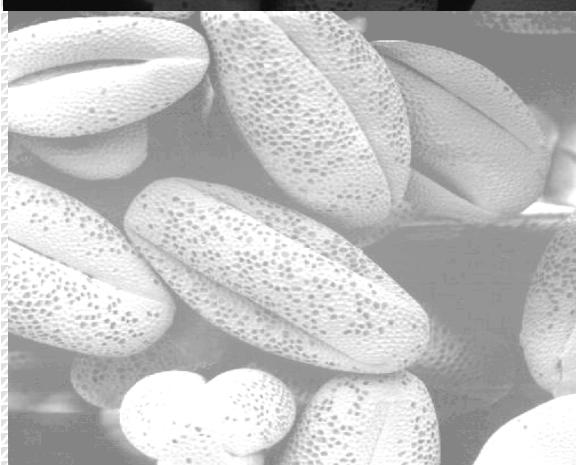
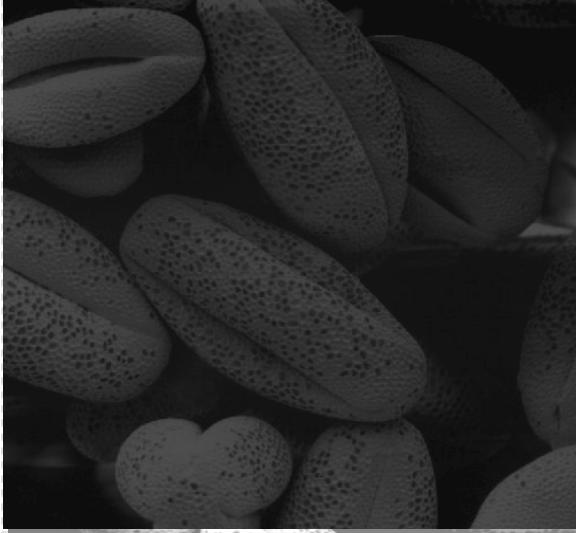
(b)



(c)

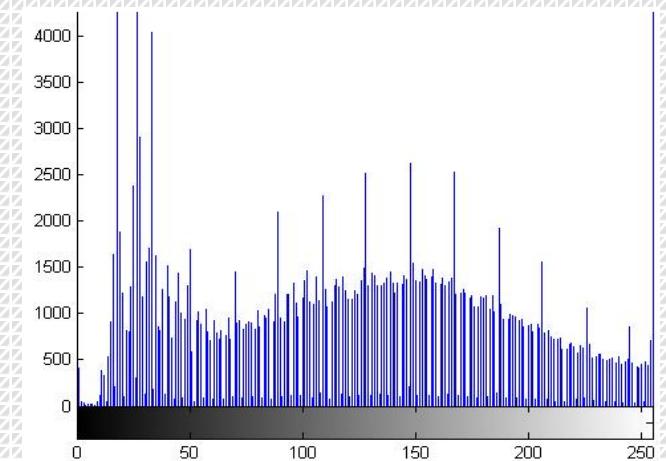
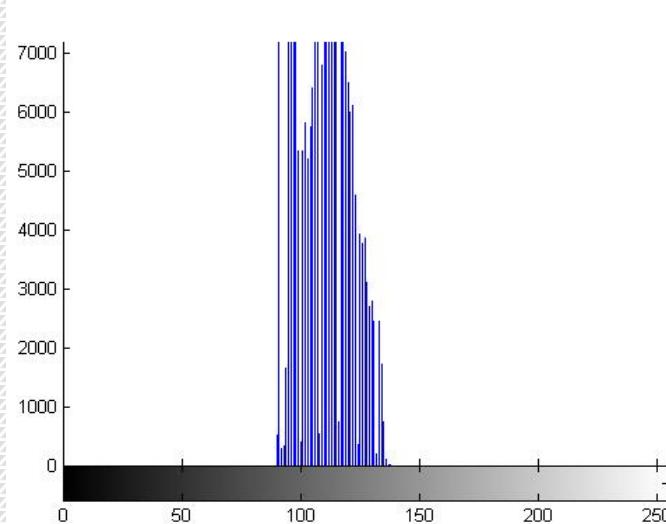
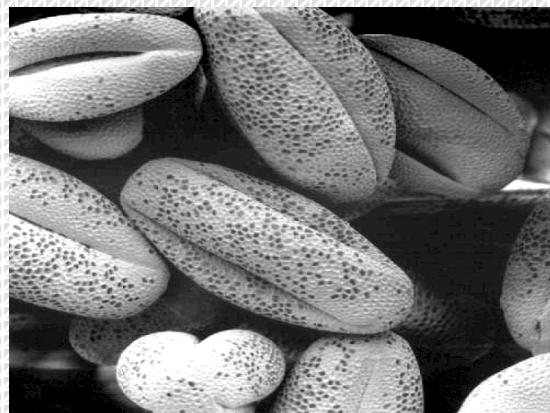
Point operators

- Histogram equalization:



Point operators

- Histogram equalization:



Point operators



- **Histogram equalization:**

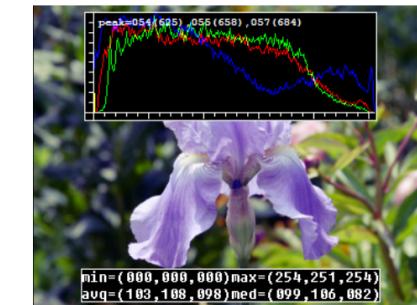
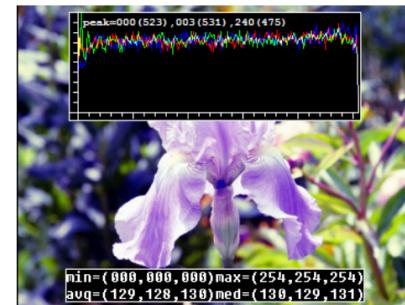
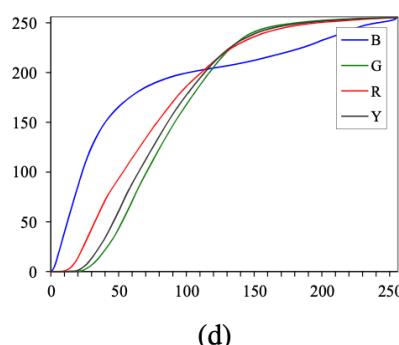
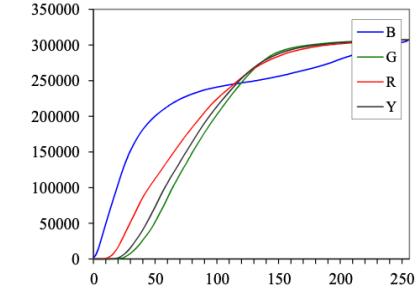
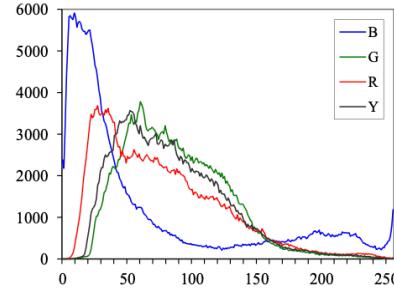
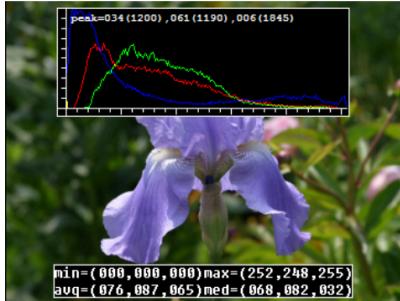


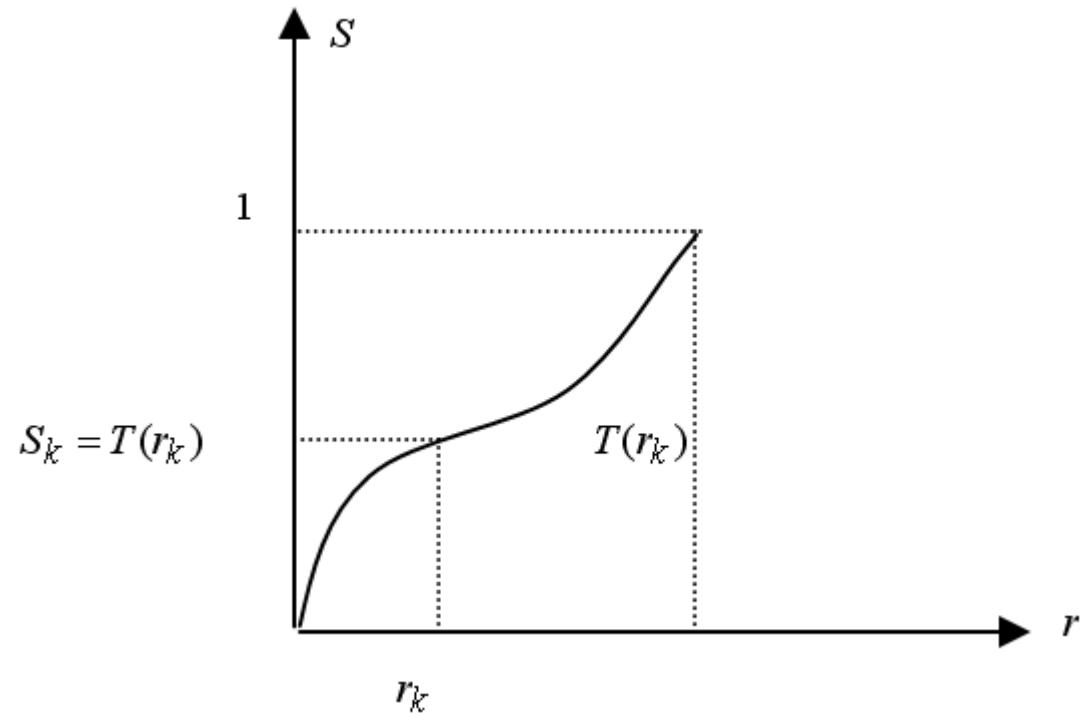
Figure 3.7 Histogram analysis and equalization: (a) original image (b) color channel and intensity (luminance) histograms; (c) cumulative distribution functions; (d) equalization (transfer) functions; (e) full histogram equalization; (f) partial histogram equalization.

Point operators



- How to perform histogram equalization?

$$S = T(r)$$



An example of histogram equalization function

Point operators

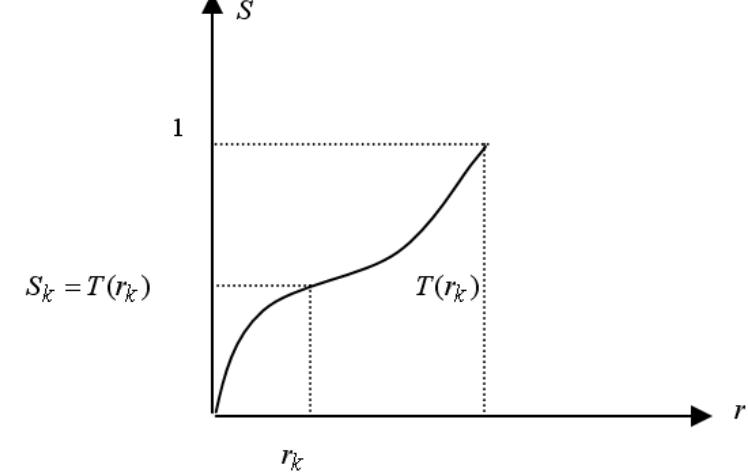


- How to perform histogram equalization?

The transform function $T(r)$ should follow the assumption :

(1) when $0 \leq r \leq 1$, $T(r)$ increases monotonically ;

(2) when $0 \leq r \leq 1$, there is $0 \leq T(r) \leq 1$



Point operators



- How to perform histogram equalization?

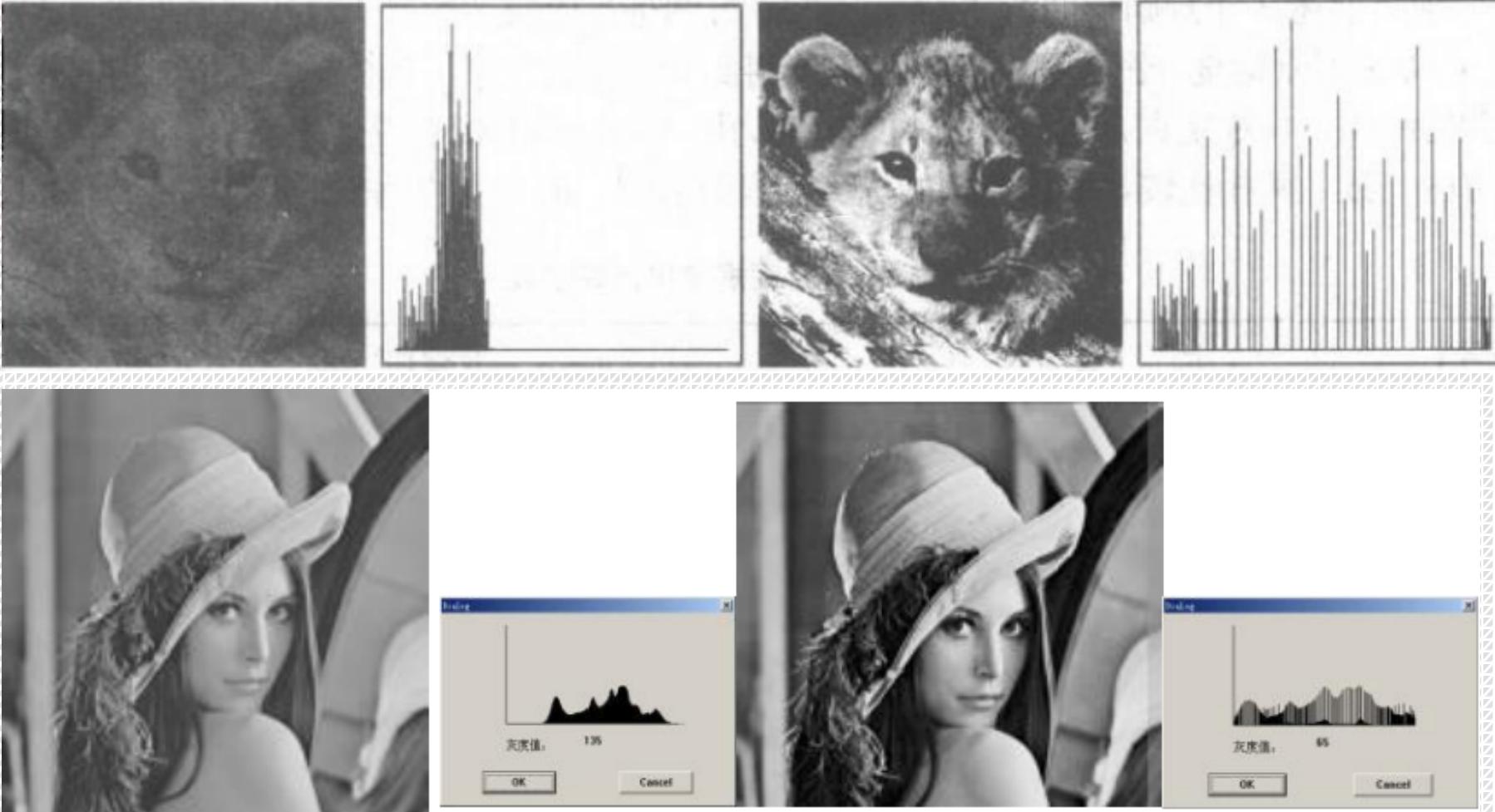
| | | | | |
|---|---|---|---|---|
| 1 | 3 | 9 | 9 | 8 |
| 2 | 1 | 3 | 7 | 3 |
| 3 | 6 | 0 | 6 | 4 |
| 6 | 8 | 2 | 0 | 5 |
| 2 | 9 | 2 | 6 | 0 |

| Pix | Ni | Pi=Ni/i mage | sumPi | sumPi * 256-1 | Approx imate |
|-----|----|-----------------|-------|------------------|-----------------|
| 0 | 3 | 0.12 | 0.12 | 29.72 | 30 |
| 1 | 2 | 0.08 | 0.2 | 50.2 | 50 |
| 2 | 4 | 0.16 | 0.36 | 91.16 | 91 |
| 3 | 4 | 0.16 | 0.52 | 132.12 | 132 |
| 4 | 1 | 0.04 | 0.56 | 142.36 | 142 |
| 5 | 1 | 0.04 | 0.6 | 152.6 | 153 |
| 6 | 4 | 0.16 | 0.76 | 193.56 | 194 |
| 7 | 1 | 0.04 | 0.8 | 203.8 | 204 |
| 8 | 2 | 0.08 | 0.88 | 224.28 | 224 |
| 9 | 3 | 0.12 | 1 | 255 | 255 |

| | | | | |
|-----|-----|-----|-----|-----|
| 30 | 132 | 225 | 225 | 224 |
| 91 | 50 | 132 | 204 | 132 |
| 132 | 194 | 30 | 194 | 142 |
| 194 | 224 | 91 | 30 | 153 |
| 91 | 255 | 91 | 194 | 30 |

Point operators

- Histogram equalization:



Methods of image transform



- *point operators* or *point processes*: the simplest kind of image transforms, namely those that manipulate each pixel independently of its neighbors.
- *neighborhood (area-based) operators*: each new pixel's value depends on a small number of neighboring input values.
- *global operators*: *geometric transformations*, such as rotations, shears, and perspective deformations.
- *global optimization*: involve the minimization of an energy functional or, equivalently, optimal estimation using Bayesian *Markov random field* models.

Neighborhood operators

Neighborhood operators can be used to *filter* images in order to add soft blur, sharpen details, accentuate edges, or remove noise



(a)



(b)



(c)



(d)

Neighborhood operators

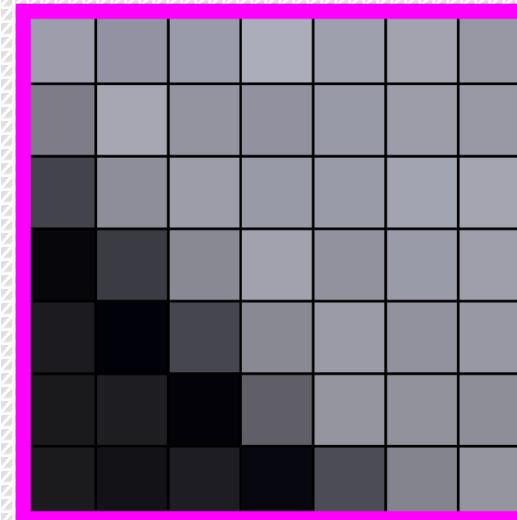
- What is the convolution?

Call this operation “*convolution*”

Filter or kernel

$\frac{1}{49}$

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 1x |
| 1x |
| 1x |
| 1x |
| 1x |
| 1x |
| 1x |
| 1x |



Source:

<https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>

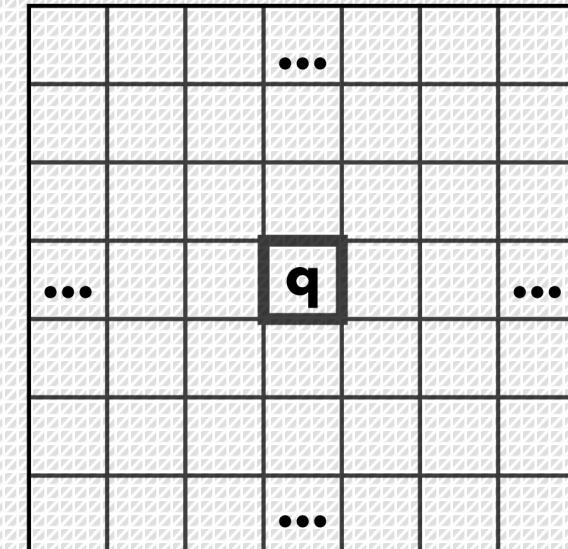
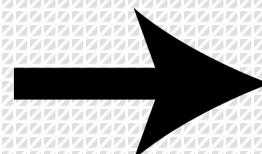
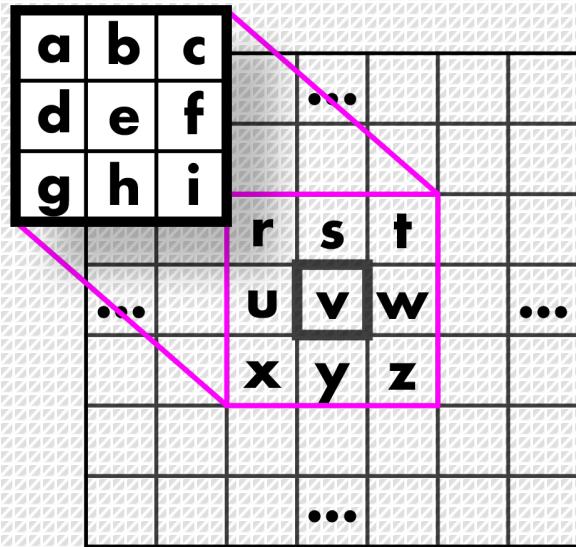


Neighborhood operators

- How does the convolution work?

Source:

<https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>

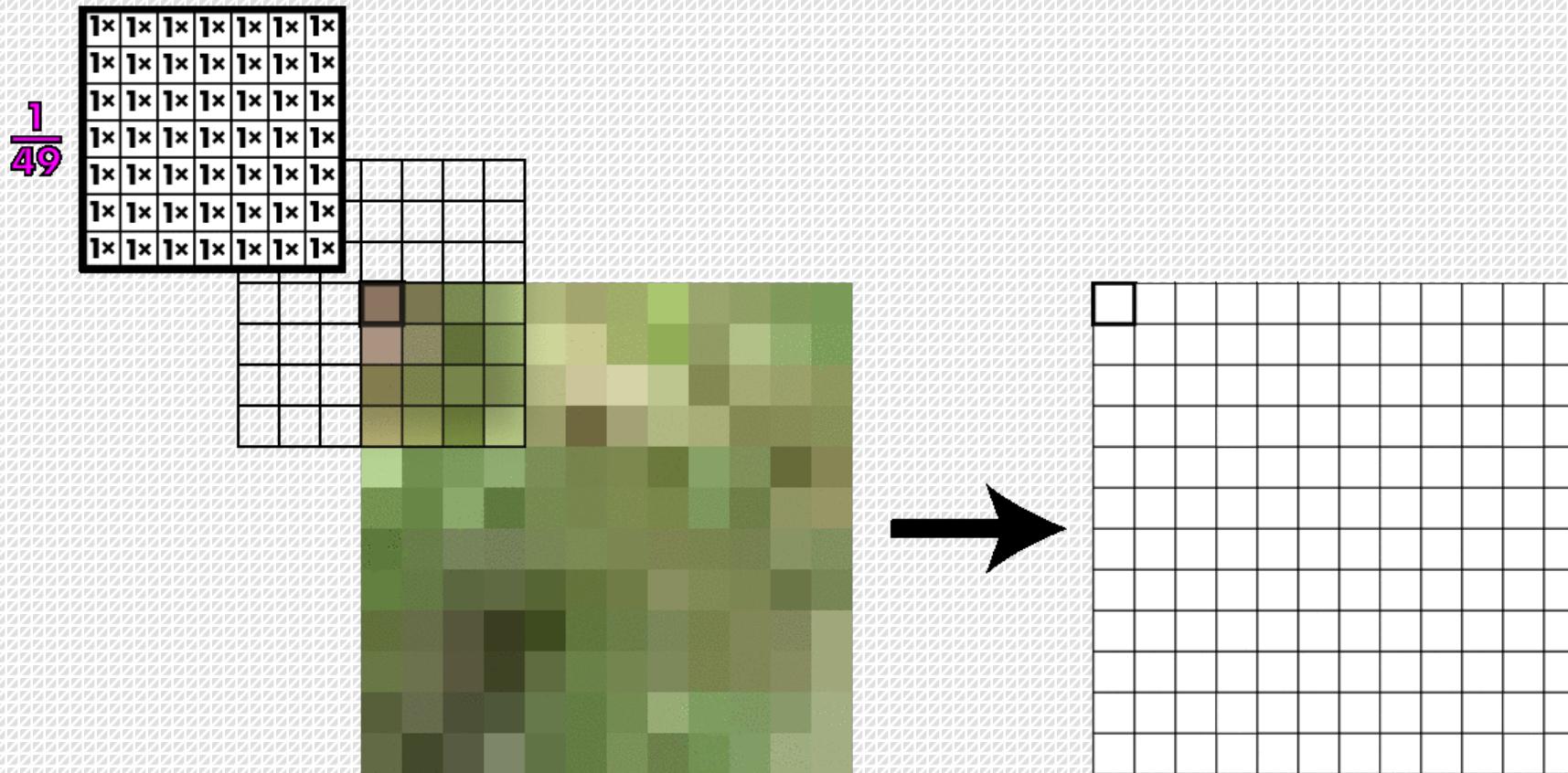


$$q = a \times r + b \times s + c \times t + d \times u + e \times v + f \times w + g \times x + h \times y + i \times z$$

Neighborhood operators



- How does the convolution work?
 - Kernel slides across image



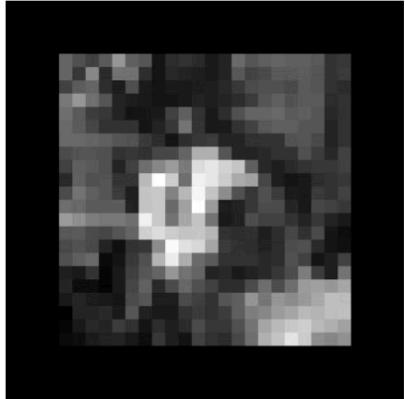
Neighborhood operators



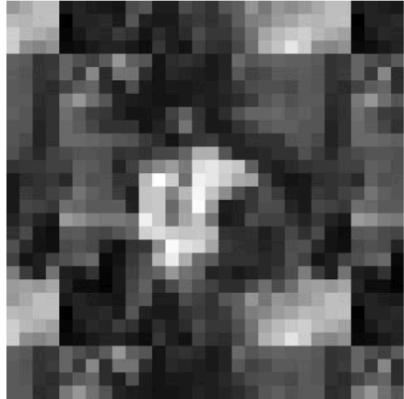
• Padding strategies

- *zero*: set all pixels outside the source image to 0 (a good choice for alpha-matted cutout images);
- *constant (border color)*: set all pixels outside the source image to a specified *border* value;
- *clamp (replicate or clamp to edge)*: repeat edge pixels indefinitely;
- *(cyclic) wrap (repeat or tile)*: loop “around” the image in a “toroidal” configuration;
- *mirror*: reflect pixels across the image edge;
- *extend*: extend the signal by subtracting the mirrored version of the signal from the edge pixel value.

Neighborhood operators



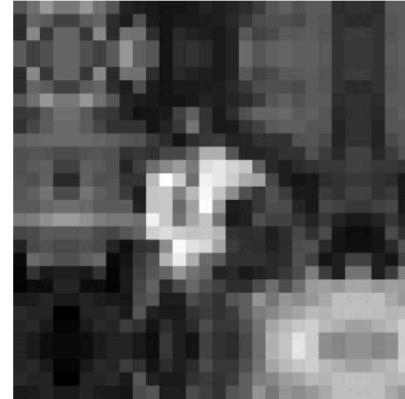
zero



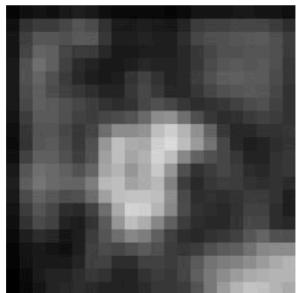
wrap



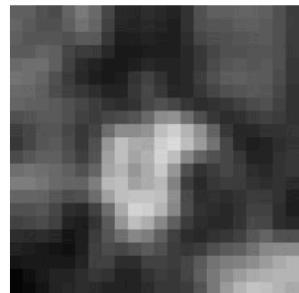
clamp



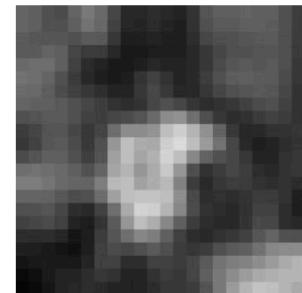
mirror



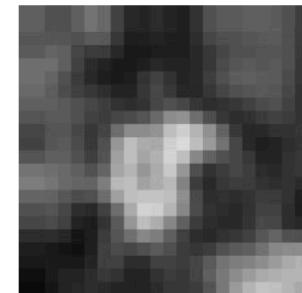
blurred zero



normalized zero



blurred clamp



blurred mirror

Figure 3.13 Border padding (top row) and the results of blurring the padded image (bottom row). The normalized zero image is the result of dividing (normalizing) the blurred zero-padded RGBA image by its corresponding soft alpha value.

Neighborhood operators

- Examples of kernels:



$$\begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix}$$

A diagram illustrating the application of a 3x3 kernel to a 3x3 neighborhood of pixels in a grayscale image. The kernel values are shown in the top-left 3x3 matrix. A pink cross-shaped mask indicates the receptive field of the central pixel in the neighborhood. The resulting output value is shown in the center of the neighborhood grid.



Source:

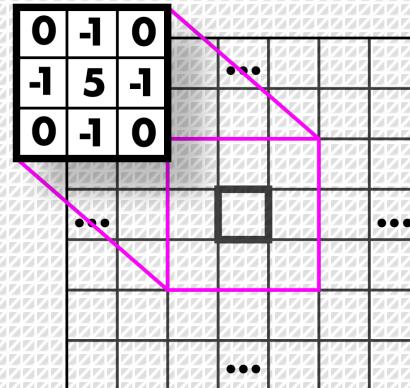
<https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



Highpass Kernel: finds edges

Neighborhood operators

- Examples of kernels:



Source:

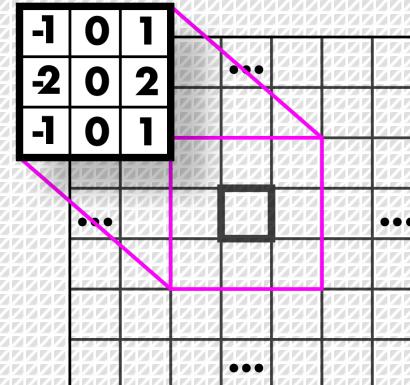
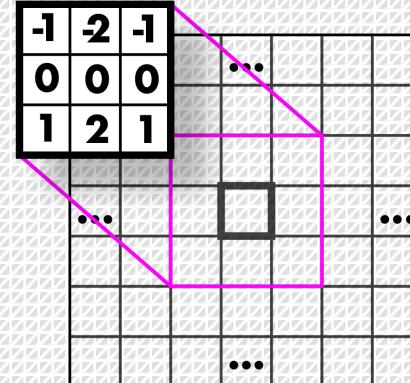
<https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



Sharpen Kernel: sharpens!

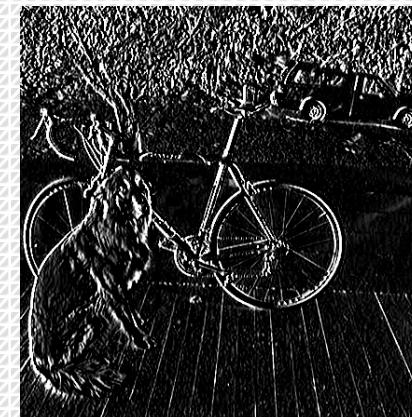
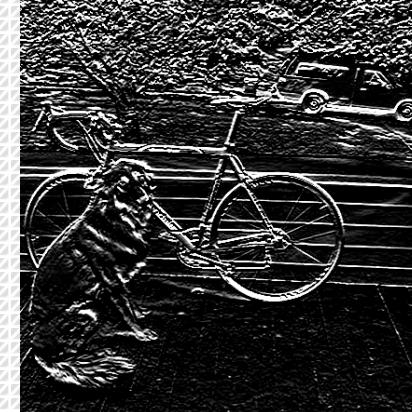
Neighborhood operators

- Examples of kernels:



Source:

<https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



Sobel Kernels: edges

■ Neighborhood operators

- Blurring
- Sharpening
- Edges
- Features
- Derivatives
- Super-resolution
- Classification
- Detection
- Image captioning
- ...

Source:

<https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



■ Neighborhood operators

- There are much more!!!



Neighborhood operators



- **Properties of convolution:** Commutative property

$$f * h = h * f$$

Convolution is commutative, which is a fancy word for saying that order doesn't matter.

Neighborhood operators



- **Properties of convolution:** Associative property

$$f * h * x = f * (x * h)$$

Convolution is also **associative**, which means that if we're chaining a series of convolutions together, it doesn't matter which one you do first.

Neighborhood operators



- **Properties of convolution:** Distributive over addition

$$h \circ (f_0 + f_1) = h \circ f_0 + h \circ f_1,$$

The **distributive** property says that we can equivalently mix the signals first, and then convolve by h :

Neighborhood operators



- **Properties of convolution:** Linearity Shift-invariance

$$h \circ (c_0 f_0 + c_1 f_1) = c_0 h \circ f_0 + c_1 h \circ f_1$$

Linearity is another important characteristic of many systems, including convolution. Broadly speaking, it encapsulates our notions of *gain* and *mixing* of signals

Neighborhood operators



- **Properties of convolution:** Linearity Shift-invariance

A filter h is **shift-invariant** if for all delays (implemented by delay filter Δ) and all input signals f , the following is true:

$$h * (\Delta f) = \Delta * (h * f)$$

Shift invariance is that the operator “behaves the same everywhere”.

■ Neighborhood operators



- **Linear operators:** which involve weighted combinations of pixels in small neighborhoods.

$$g(i, j) = \sum_{k,l} f(i + k, j + l) h(k, l)$$

The entries in the weight *kernel* or *mask* $h(k, l)$ are often called the *filter coefficients*.



■ Neighborhood operators

- **Linear operators:** which involve weighted combinations of pixels in small neighborhoods.

$$g(i, j) = \sum_{k,l} f(i + k, j + l) h(k, l)$$

The entries in the weight *kernel* or *mask* $h(k, l)$ are often called the *filter coefficients*.

Neighborhood operators

- Examples of Linear operators: *moving average* or *box filter*

$\frac{1}{49}$

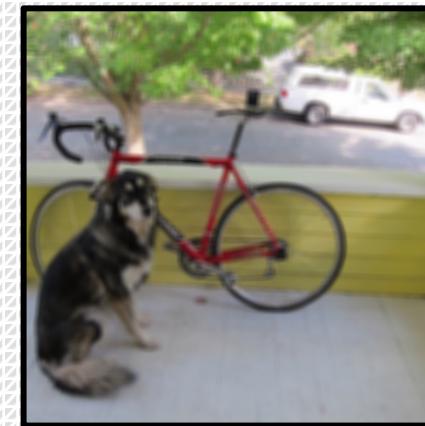
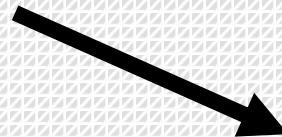
| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 1x |
| 1x |
| 1x |
| 1x |
| 1x |
| 1x |
| 1x |



Box filters

$$\frac{1}{N \times M}$$

| N | | | | M |
|-----|----|----|----|---|
| 1x | 1x | 1x | 1x | |
| 1x | 1x | 1x | 1x | |
| 1x | 1x | 1x | 1x | |
| 1x | 1x | 1x | 1x | |
| ... | | | | |
| 1x | 1x | 1x | 1x | |

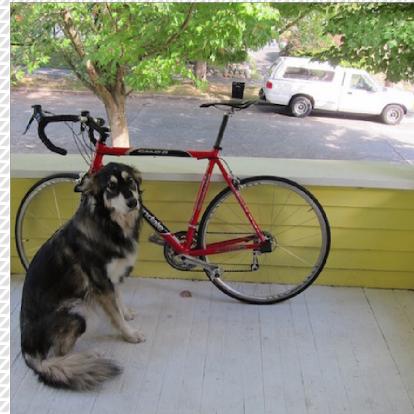


Neighborhood operators

- Examples of Linear operators: *moving average* or *box filter*

$\frac{1}{49}$

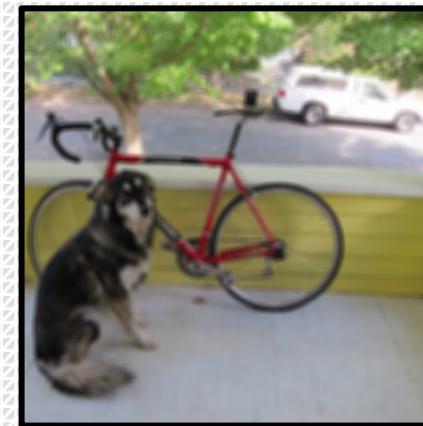
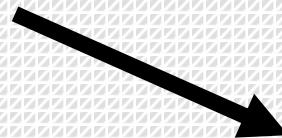
| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 1x |
| 1x |
| 1x |
| 1x |
| 1x |
| 1x |
| 1x |



Box filters

$$\frac{1}{N \times M}$$

| N | | | | M |
|-----|----|----|----|---|
| 1x | 1x | 1x | 1x | |
| 1x | 1x | 1x | 1x | |
| 1x | 1x | 1x | 1x | |
| 1x | 1x | 1x | 1x | |
| ... | | | | |
| 1x | 1x | 1x | 1x | |



■ Neighborhood operators



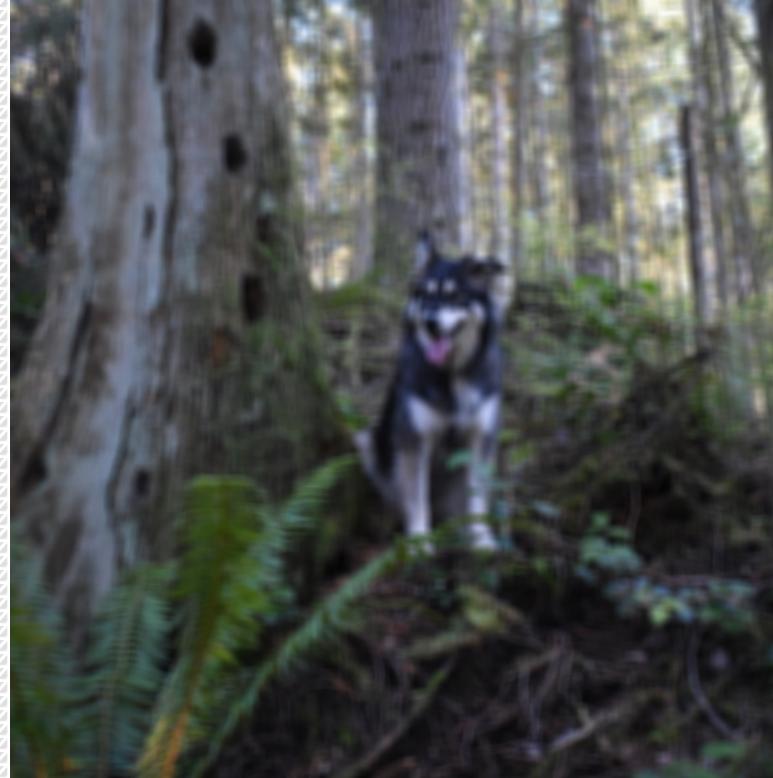
- **Examples of Linear operators:** *moving average* or *box filter*

Are also blurring (smoothing) or *low-pass* kernels (since they pass through the lower frequencies while attenuating higher frequencies).

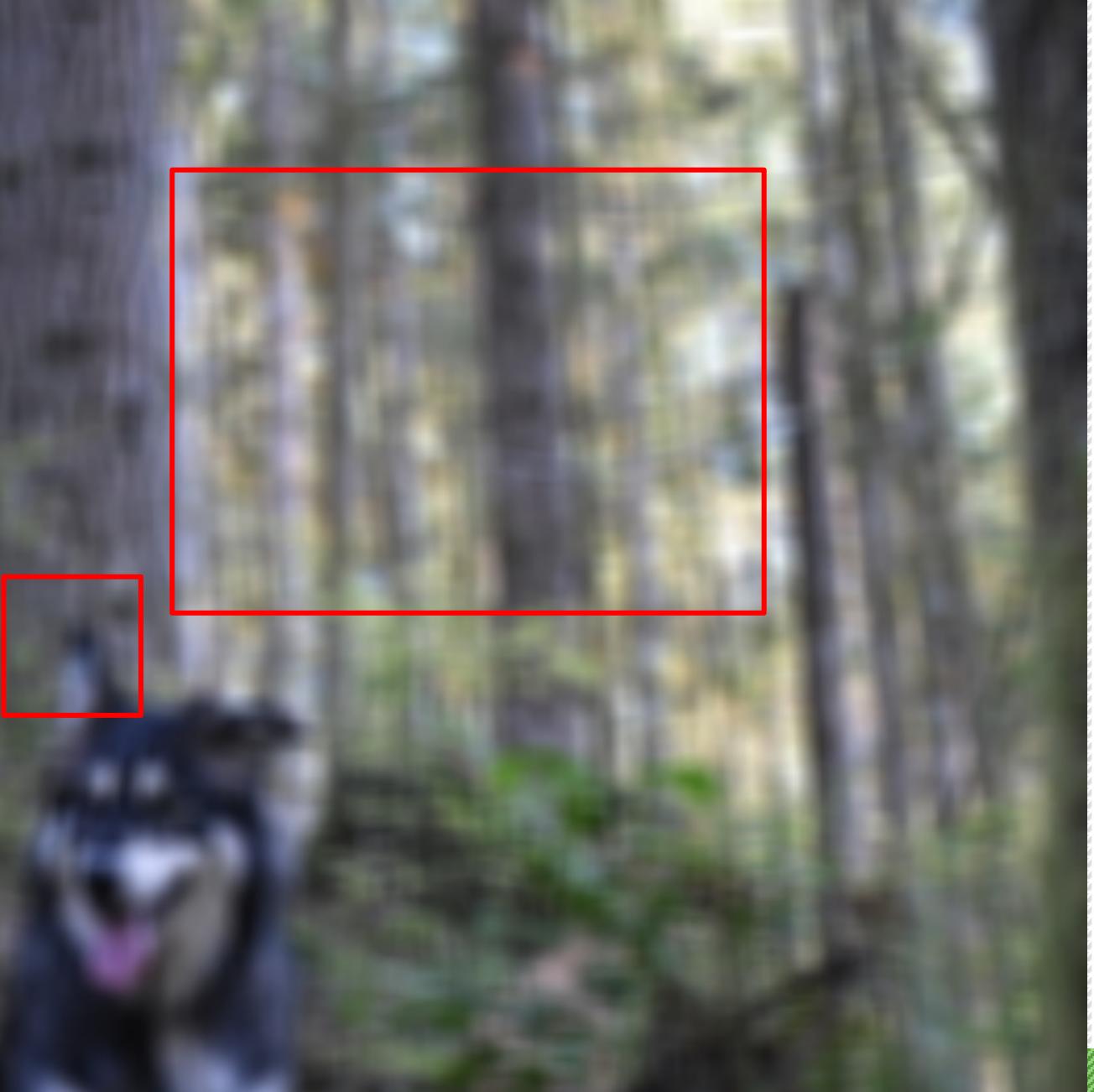
■ Neighborhood operators



- *Moving average or box filter – generate artifact*



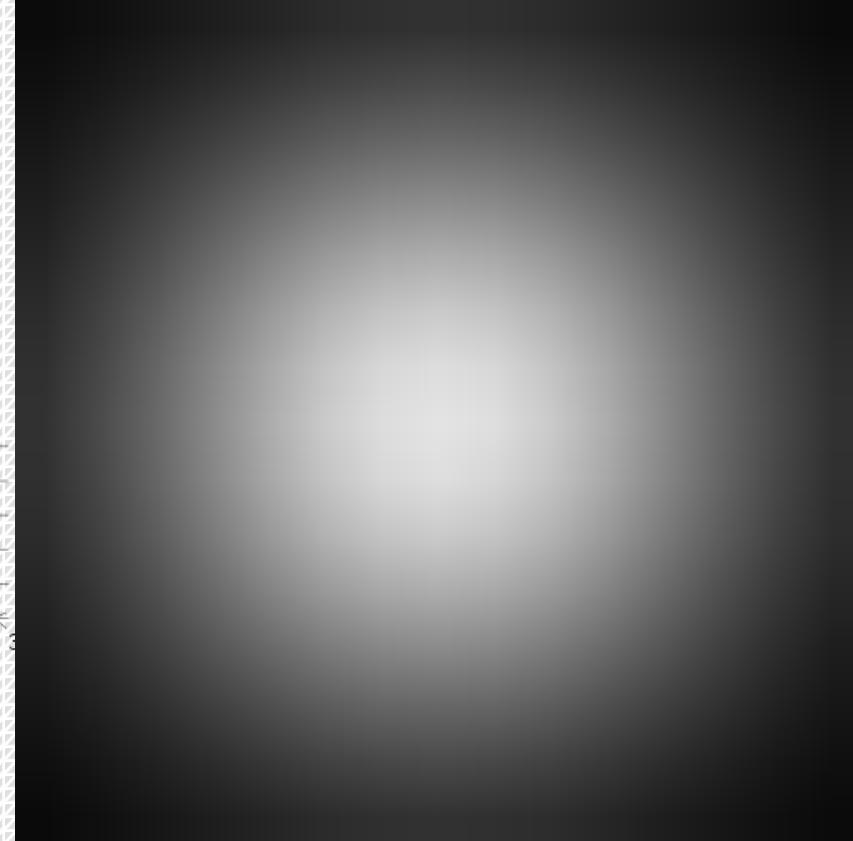
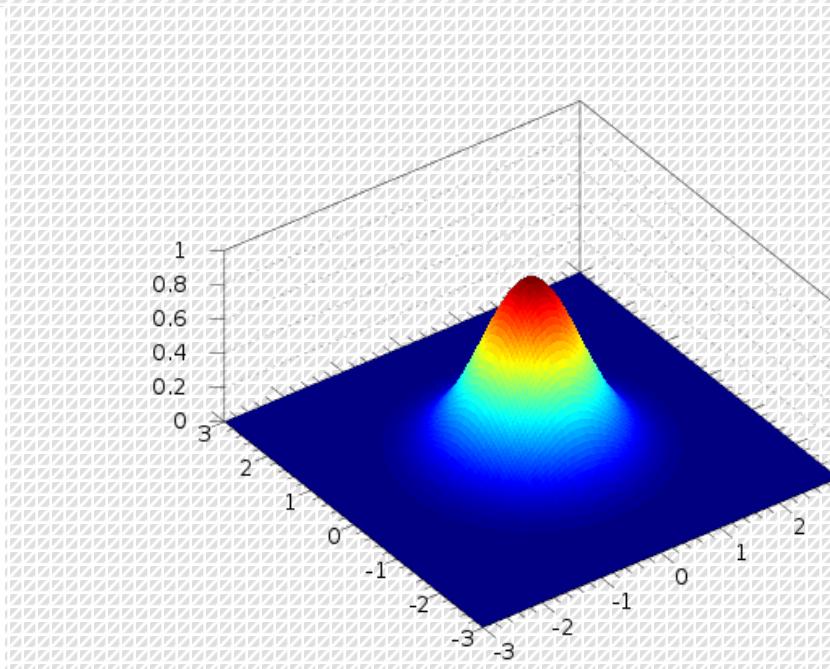
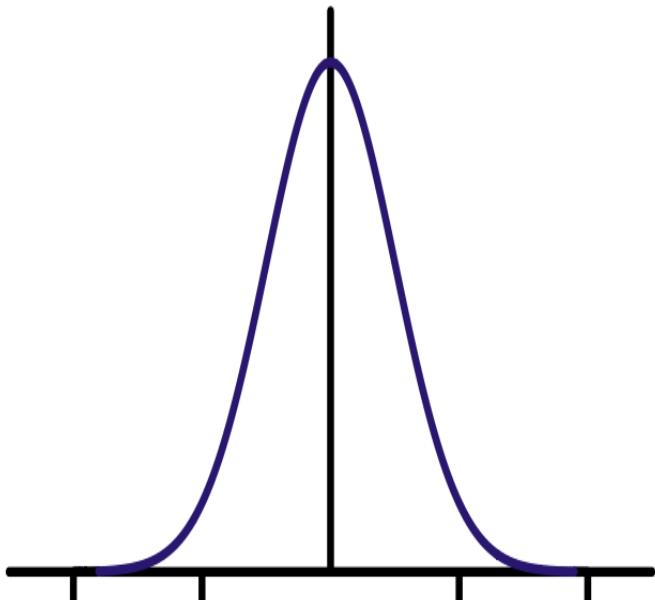
Neighborhood operators



■ Neighborhood operators

- A *Smoothly* weighted kernel

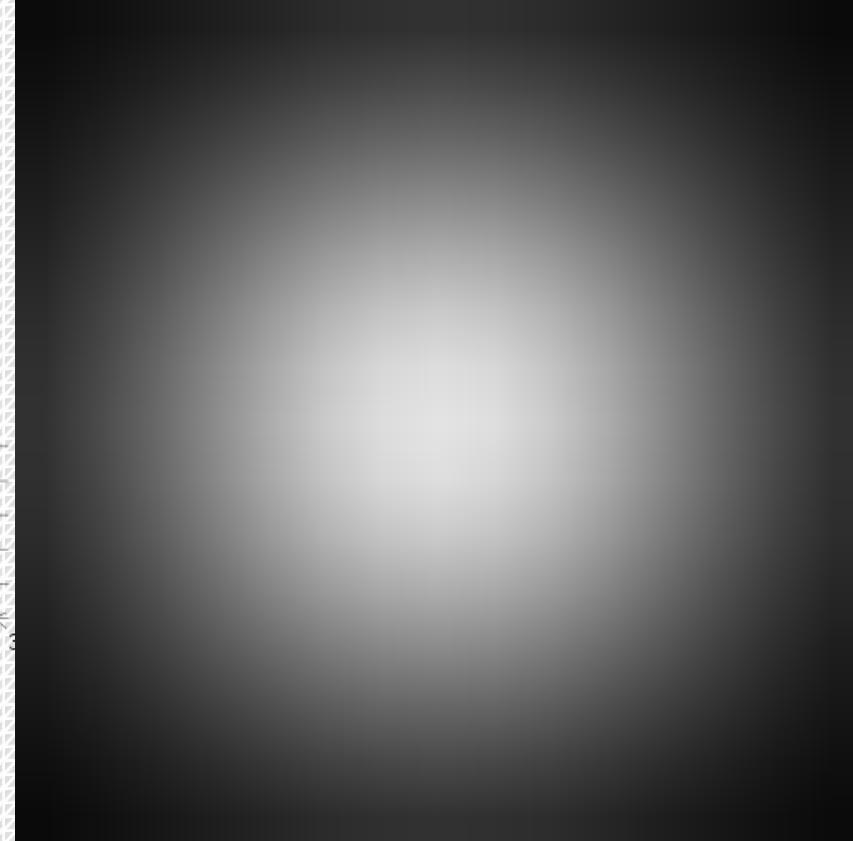
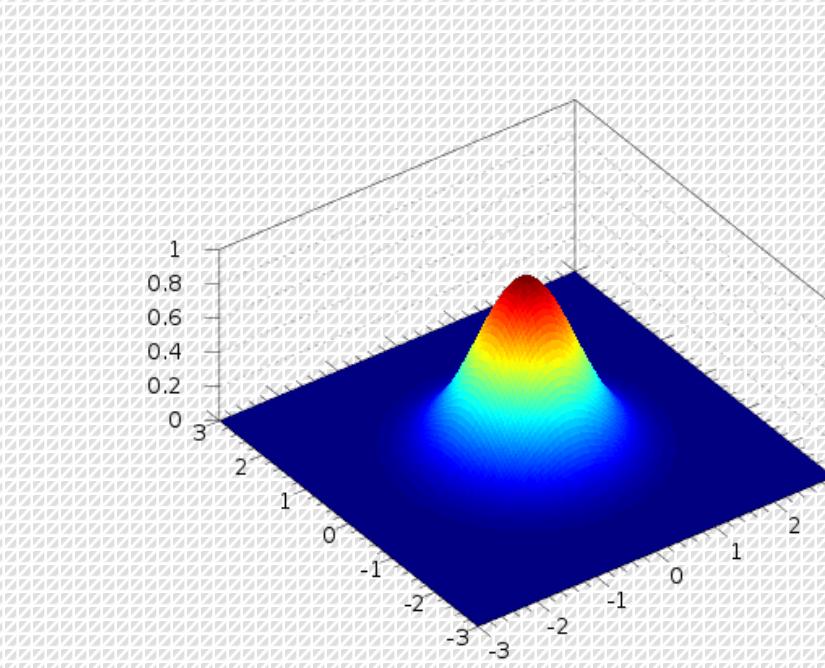
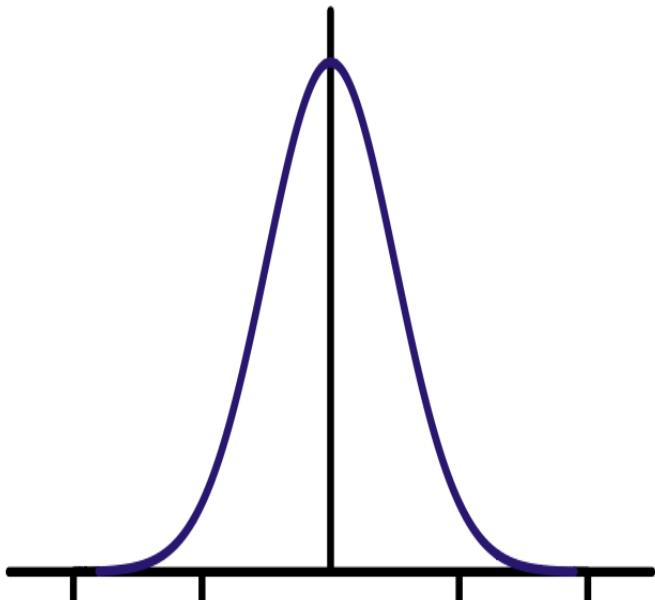
$$g(x, y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$



■ Neighborhood operators

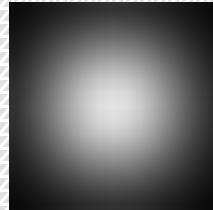
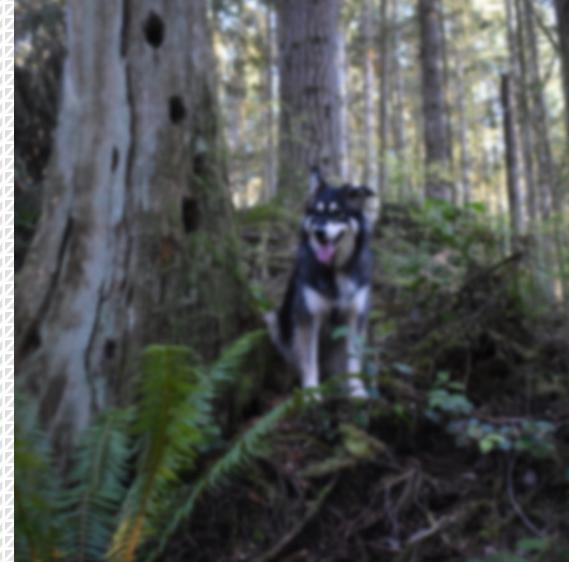
- A *Smoothly* weighted kernel

$$g(x, y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$



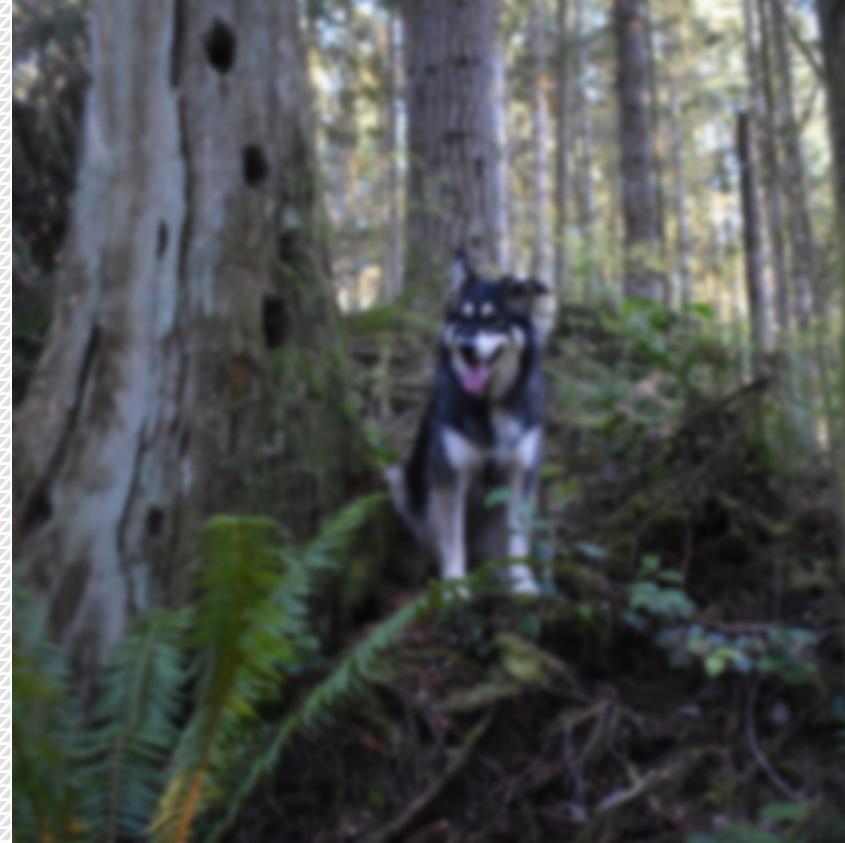
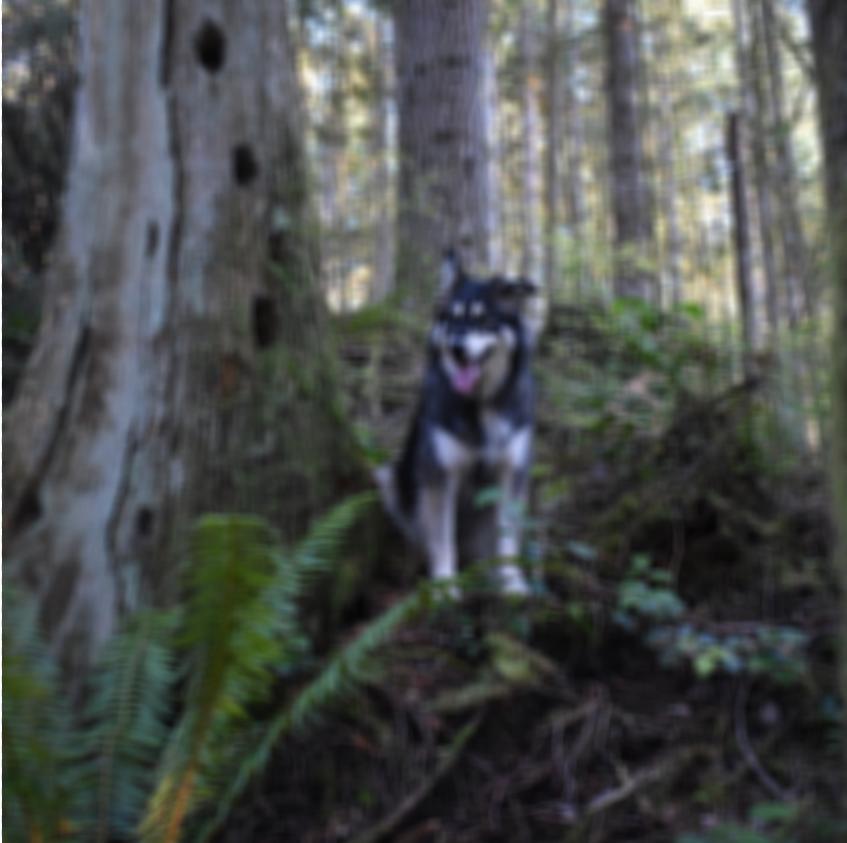
Neighborhood operators

- A *Smoothly* weighted kernel

 $*$  $=$ 

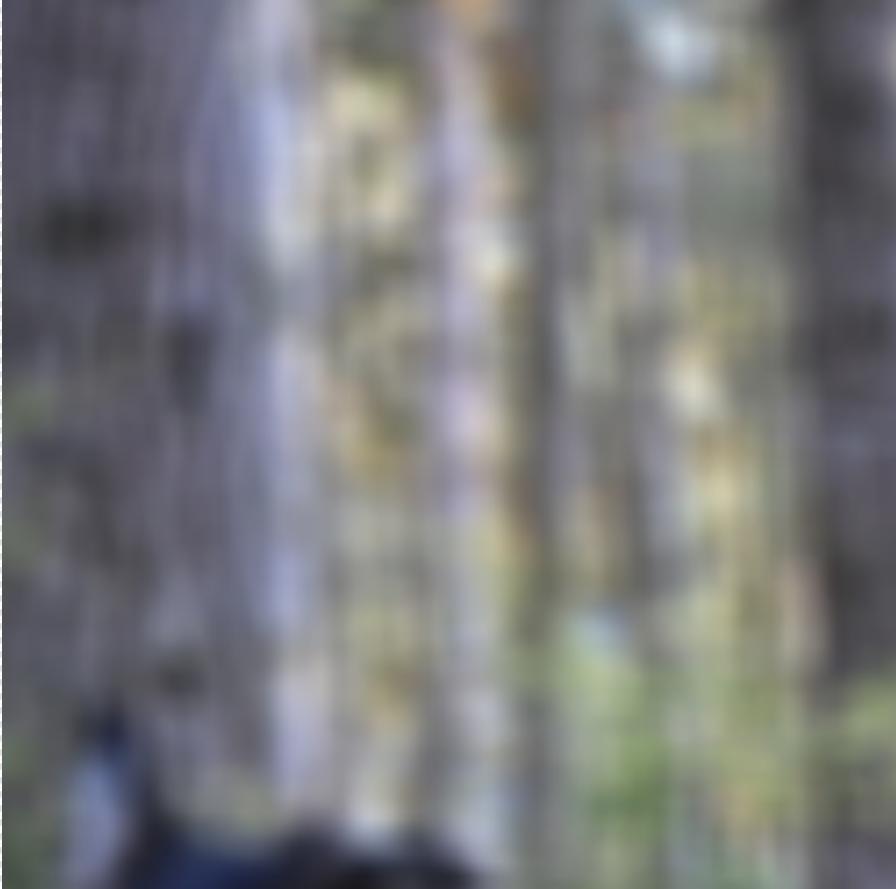
Neighborhood operators

- Better smoothing with Gaussians

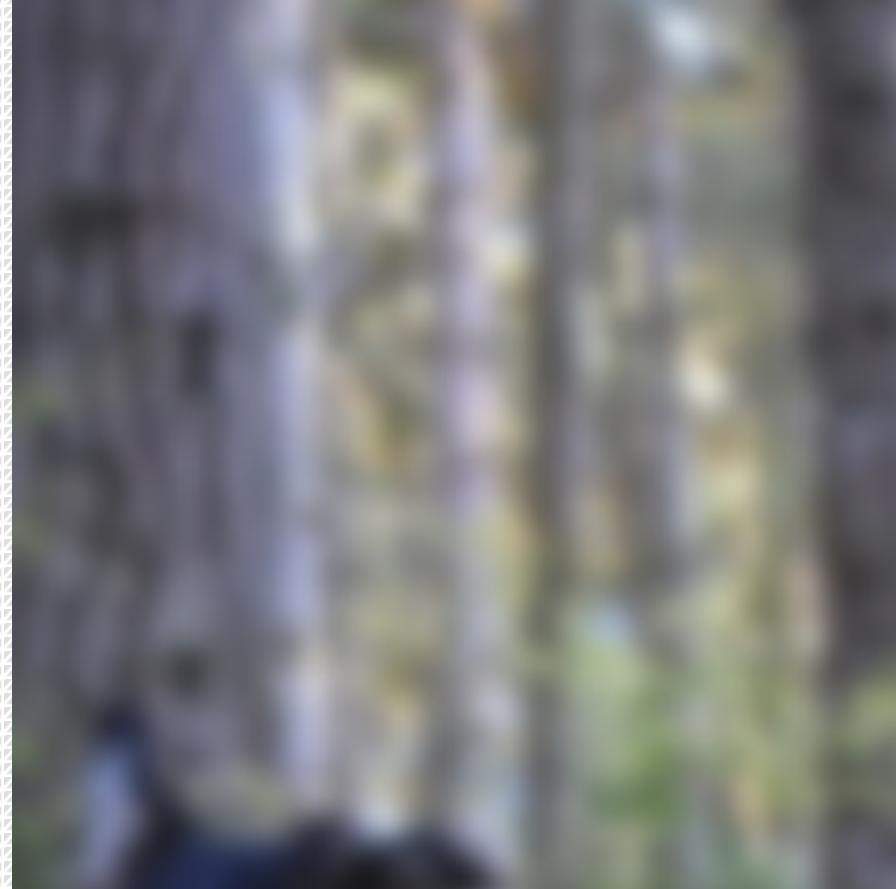


Neighborhood operators

- Better smoothing with Gaussians



Box Filtered



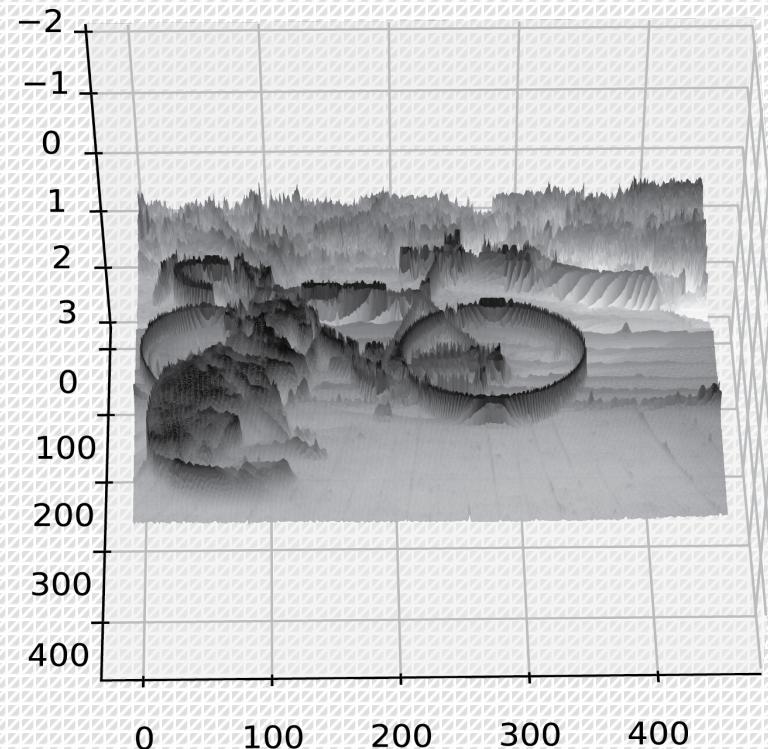
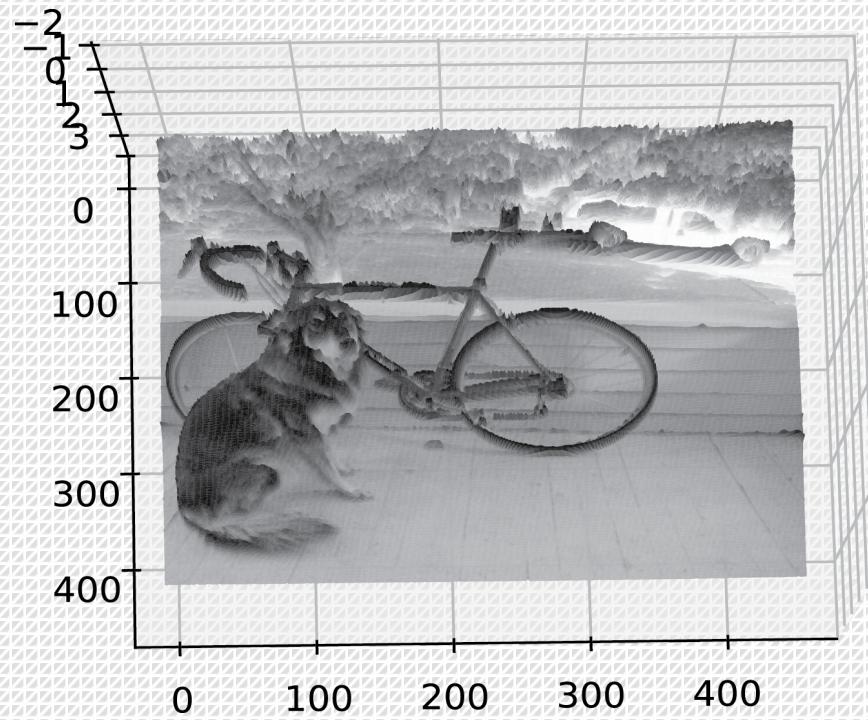
Gaussian Filtered

Edges

Source: <https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



- What is an “edge”?

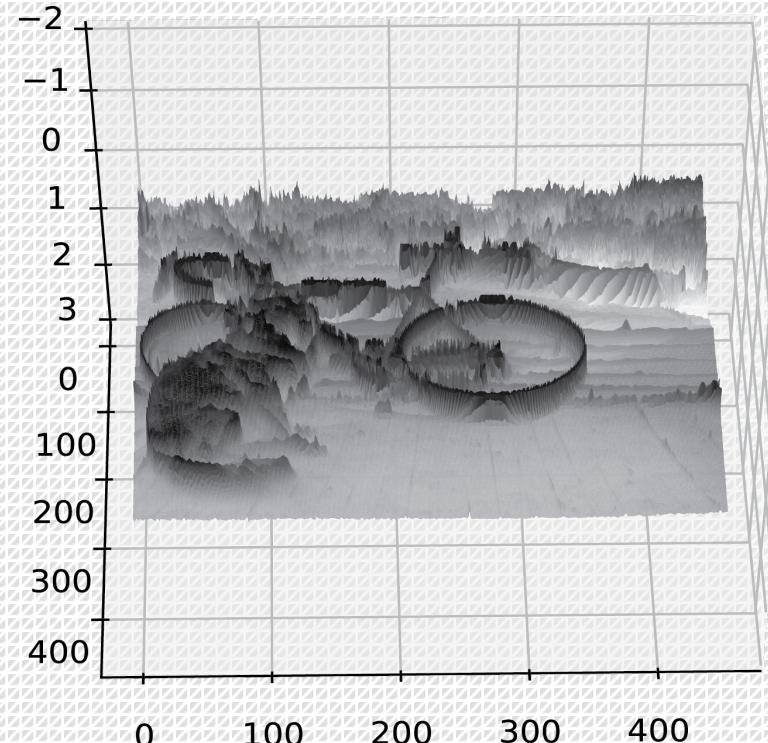
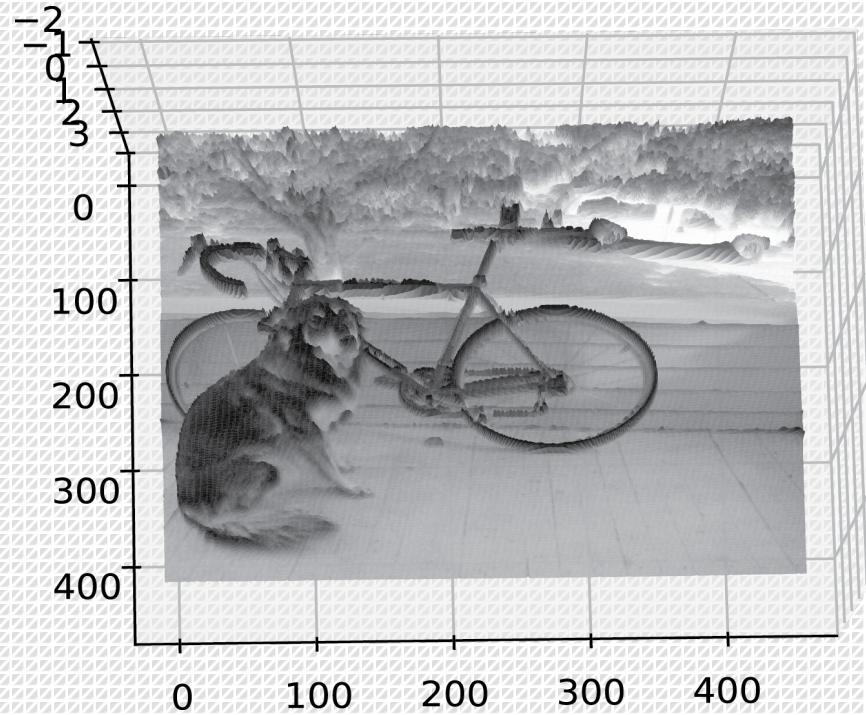


Edges

Source: <https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



- Image is a function.
- Think of the gray tones as HEIGHTS.
- Edges are rapid changes in this function

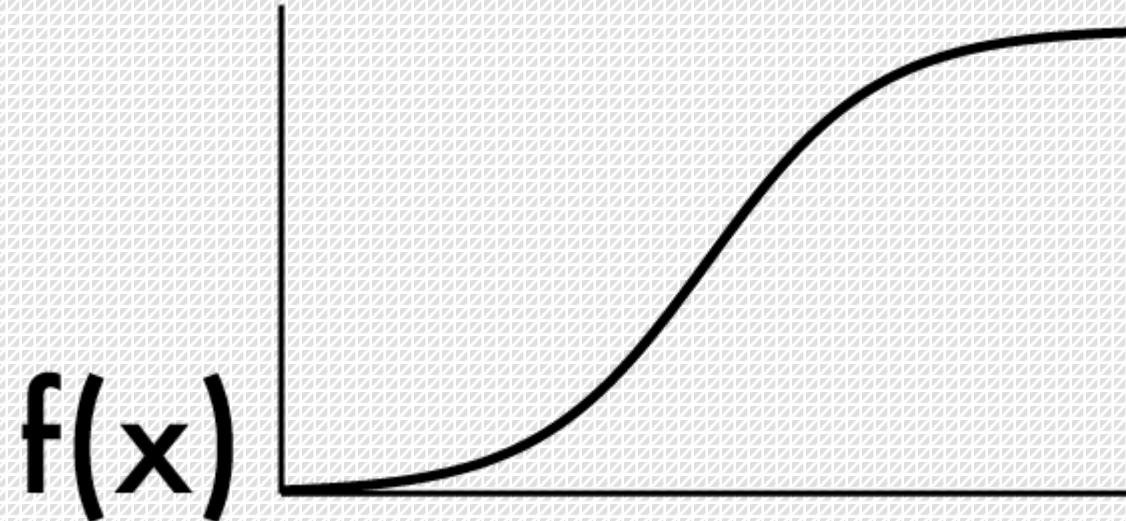


Edges

Source: <https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



- Image is a function
- Edges are rapid changes in this function

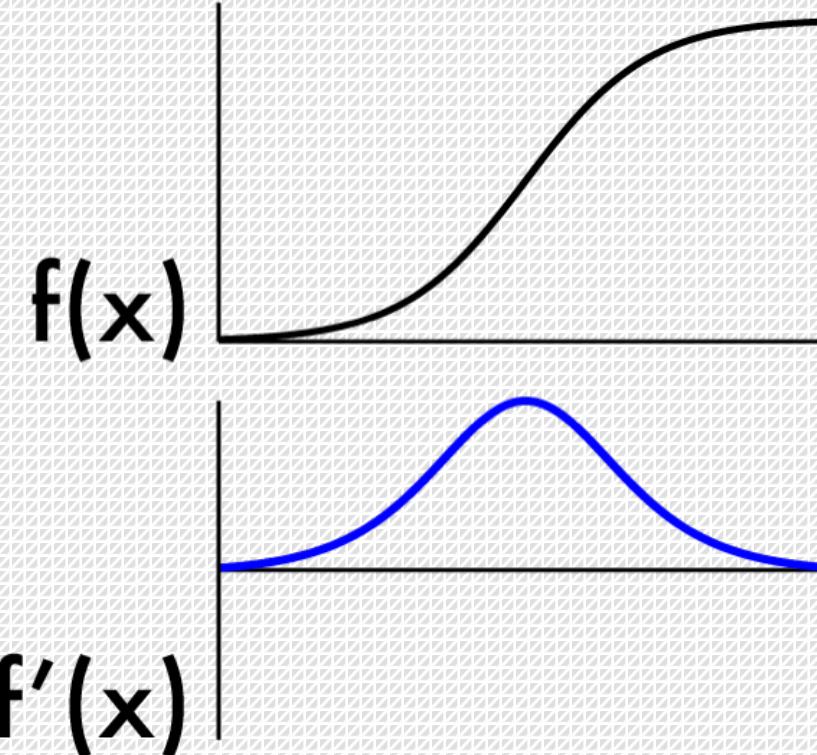


Edges

Source: <https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



- Image is a function
- Edges are rapid changes in this function



Edges

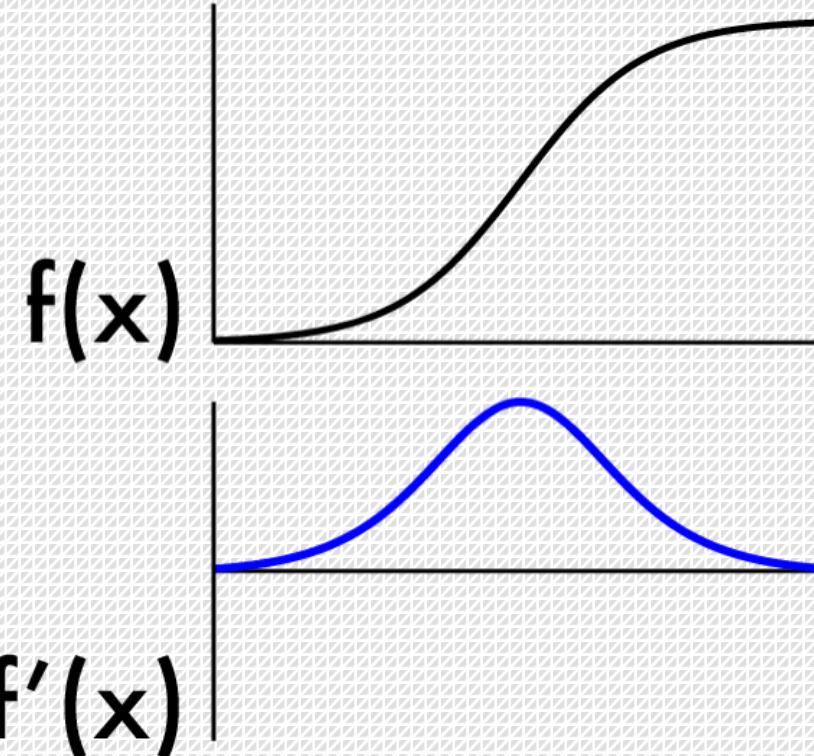
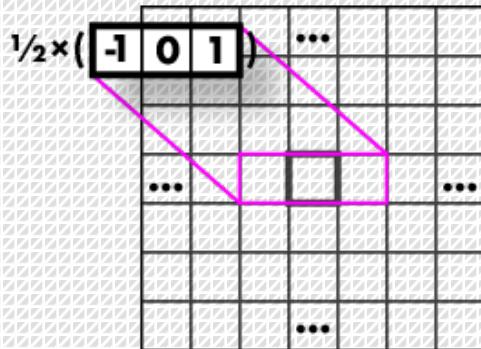
Source: <https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



- Recall:

$$f'(a) = \lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h}.$$

- We don't have an "actual" function, must estimate
- Possibility: set $h = 2$
- What will that look like?

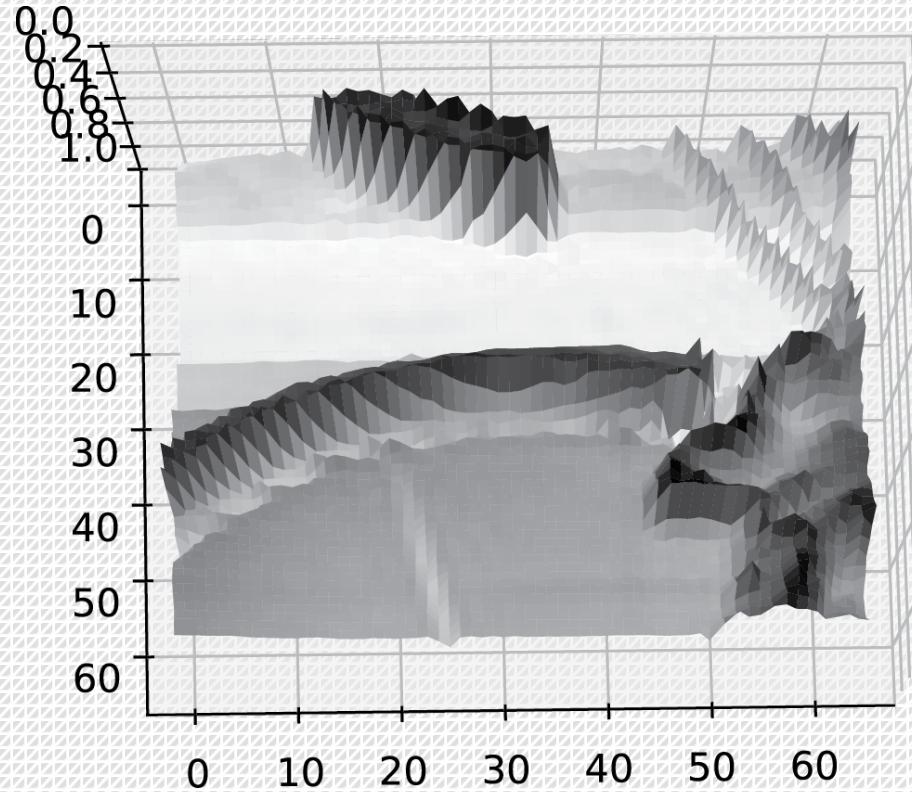
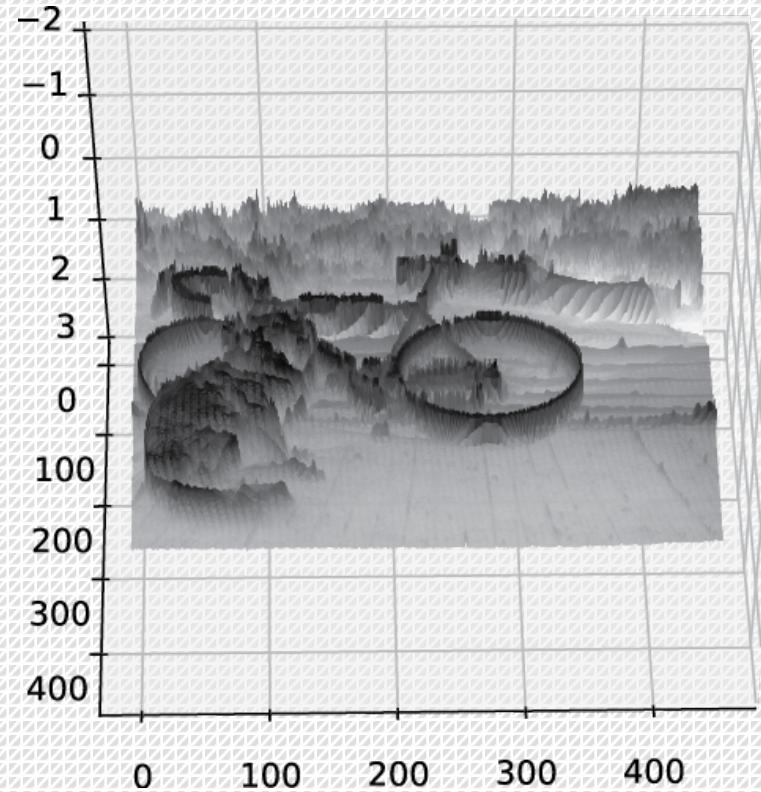


Edges

Source: <https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



- Images are noisy!

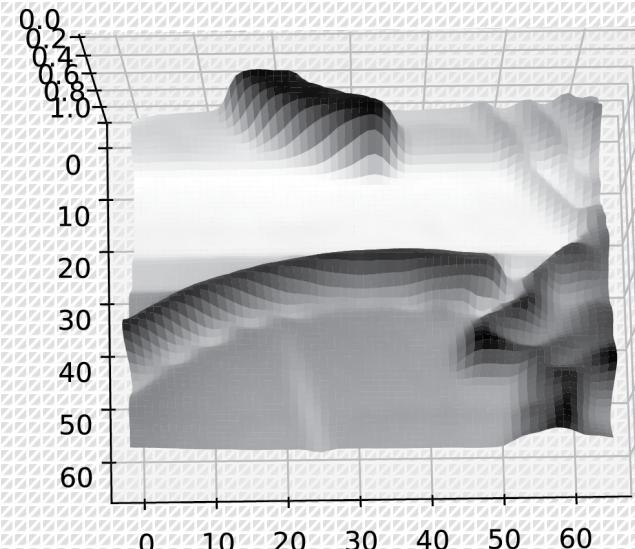
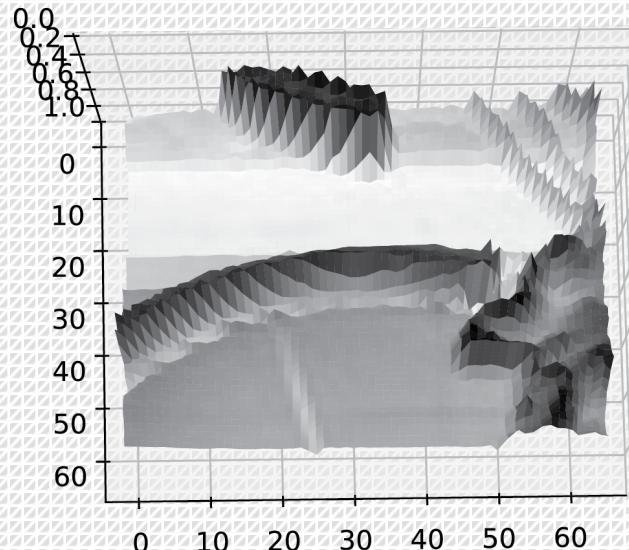
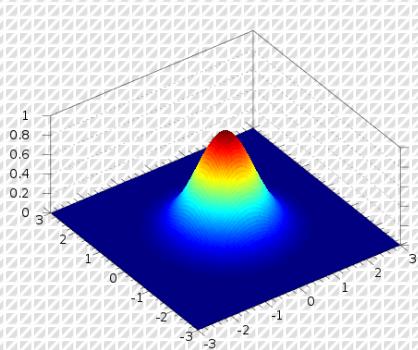


Edges

Source: <https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



- Images are noisy!



Edges

Source: <https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



- Images are noisy!

Diagram illustrating the convolution operation for edge detection:

Input image (G) is a 3x3 grid:

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

Kernel (K) is a 1x3 filter:

| | | |
|----|---|---|
| -1 | 0 | 1 |
|----|---|---|

The result of the convolution is a 3x3 output grid:

| | | | | |
|--|--|--|--|--|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Handwritten annotations include a red 'G' above the input grid and a red bracket above the kernel.

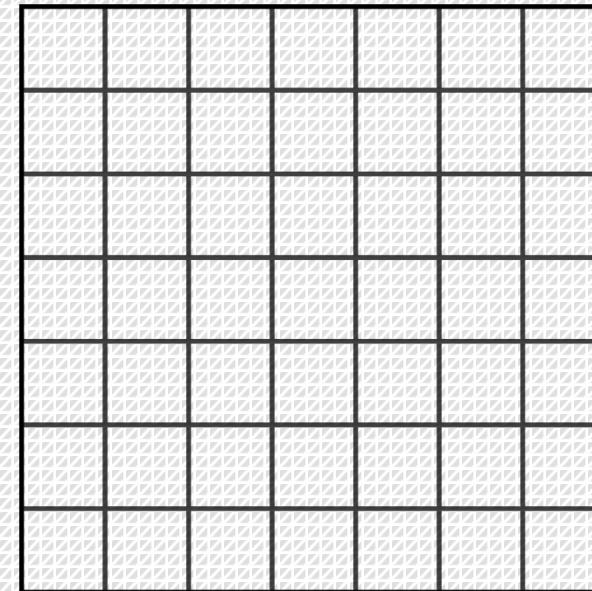
Edges

Source: <https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



- Smooth first, then derivative

$$\frac{1}{2} \times (-1 \ 0 \ 1) * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} *$$



Edges

Source: <https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



- Smooth first, then derivative

$$\frac{1}{2} \times \left(\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$

The diagram illustrates the convolution operation. A 1x3 kernel $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ is applied to a 3x3 input matrix $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$. The result is a 1x1 output $\begin{bmatrix} 2 \end{bmatrix}$, indicated by a large arrow. A pink line connects the top-left element of the kernel to the top-left element of the input matrix, highlighting the receptive field of that output unit.

Edges

Source: <https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



- Smooth first, then derivative

$$\frac{1}{2} \times \left(\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$

$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$

The diagram illustrates a convolution operation. A 3x3 input matrix is multiplied by a 3x3 kernel. The result is then scaled by $\frac{1}{2}$ and reduced to a 1x2 output matrix. The input matrix has its top-right 2x2 submatrix highlighted in pink, indicating it is not used due to stride 2. The output matrix has its bottom-right cell empty, indicating it is also not used.

$$\frac{1}{2} \times \begin{bmatrix} 2 & 0 \\ & \end{bmatrix}$$

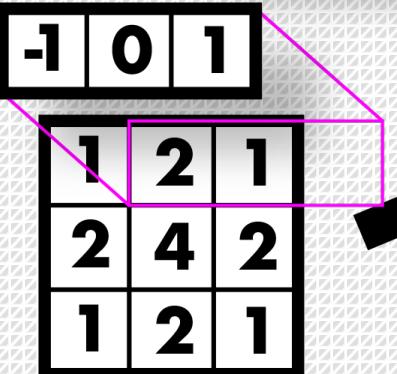
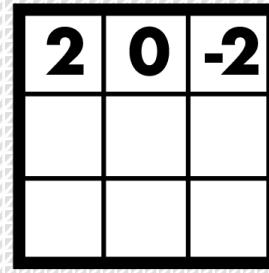
Edges

Source: <https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



- Smooth first, then derivative

$$\frac{1}{2} \times \left(\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$

$\frac{1}{2} \times$  \rightarrow $\frac{1}{2} \times$ 

A diagram illustrating a convolution operation. A 3x3 input matrix is multiplied by a 3x3 kernel. The result is then scaled by $\frac{1}{2}$. The input matrix has values 1, 2, 1 in its first row, and 2, 4, 2, 1, 2, 1 in its second and third rows. The kernel has values -1, 0, 1 in its first row, and 1, 2, 1, 2, 1 in its second and third rows. The result matrix has values 2, 0, -2 in its first row, and empty cells in its second and third rows. A pink arrow points from the input matrix to the result matrix, indicating the flow of data through the convolution process.

Edges

Source: <https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



- Smooth first, then derivative

$$\frac{1}{2} \times \left(\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$

The diagram illustrates a convolution operation. A 3x3 input matrix (bottom) is multiplied by a 3x3 kernel (top). The result is a 2x2 output matrix (right). A pink arrow points from the top-left element of the kernel to the top-left element of the input, indicating the receptive field. The output matrix is scaled by $\frac{1}{2}$.

| | | | | | | | | |
|----|---|---|--|--|--|--|--|--|
| -1 | 0 | 1 | | | | | | |
| 2 | 4 | 2 | | | | | | |
| 1 | 2 | 1 | | | | | | |

| | | | | | | | | |
|---|---|---|--|--|--|--|--|--|
| 1 | 2 | 1 | | | | | | |
| 2 | 4 | 2 | | | | | | |
| 1 | 2 | 1 | | | | | | |

| | | | | | | | | |
|----|---|---|--|--|--|--|--|--|
| -1 | 0 | 1 | | | | | | |
| 2 | 4 | 2 | | | | | | |
| 1 | 2 | 1 | | | | | | |

| | | | | | | | | |
|---|---|----|--|--|--|--|--|--|
| 2 | 0 | -2 | | | | | | |
| 4 | | | | | | | | |
| | | | | | | | | |

Edges

Source: <https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



- Smooth first, then derivative

$$\frac{1}{2} \times \left(\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$

The diagram illustrates a convolution operation. On the left, a 3x3 input matrix with values 1, 2, 1; 2, 4, 2; and 1, 2, 1 is multiplied by a 3x3 kernel with values -1, 0, 1; 1, 2, 1; and 1, 2, 1. The result is scaled by $\frac{1}{2}$. An arrow points from this result to a 2x2 output matrix with values 2, 0, -2; 4, 0, -4; and 2, 0, -2. The input matrix has a pink border around its bottom-right 3x3 submatrix, indicating the receptive field of the output unit at position (2, 2).

$$\frac{1}{2} \times \begin{bmatrix} 2 & 0 & -2 \\ 4 & 0 & -4 \\ 2 & 0 & -2 \end{bmatrix}$$

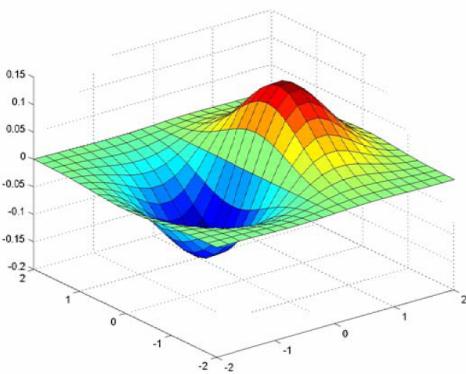
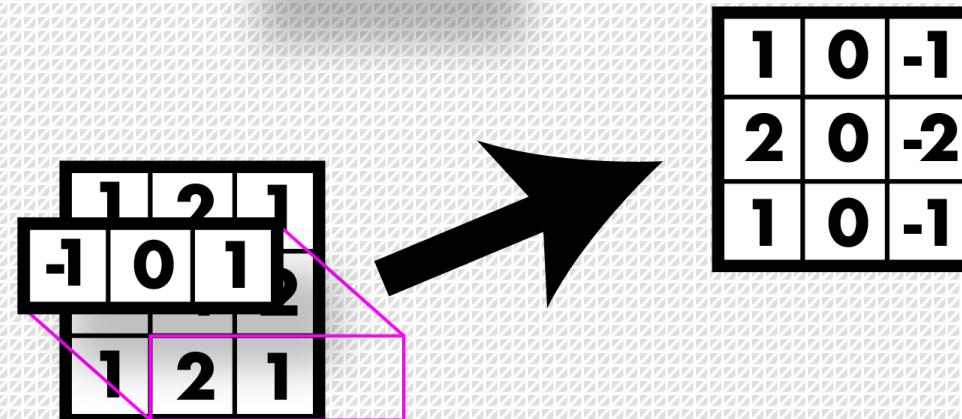
Edges

Source: <https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



- Smooth first, then derivative

$$\frac{1}{2} \times \left(\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$



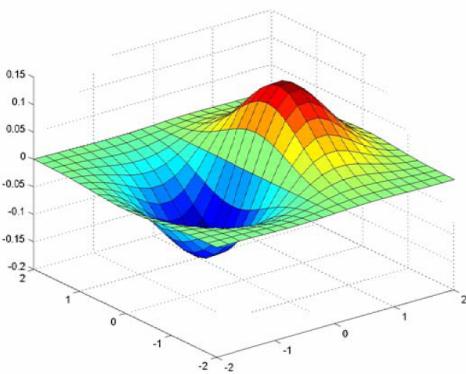
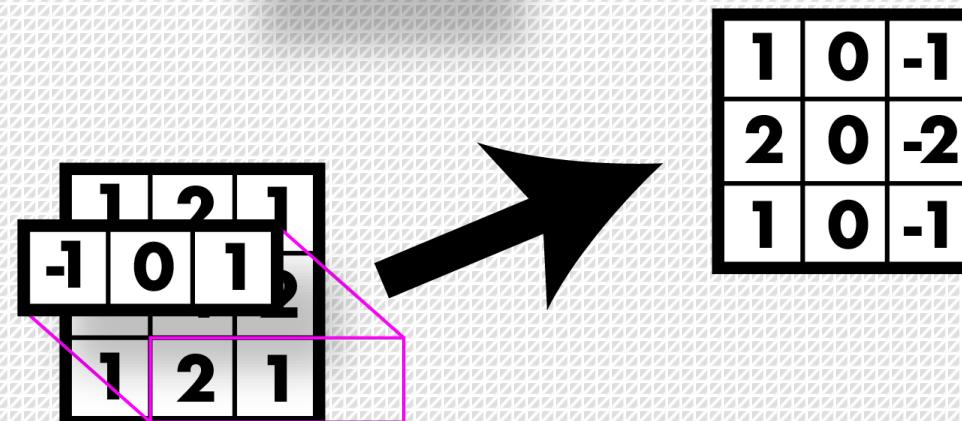
Edges

Source: <https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



- Smooth first, then derivative

$$\frac{1}{2} \times \left(\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$

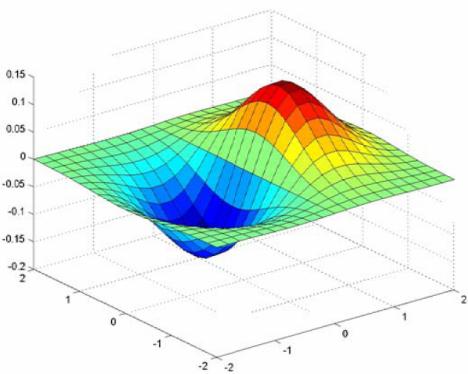
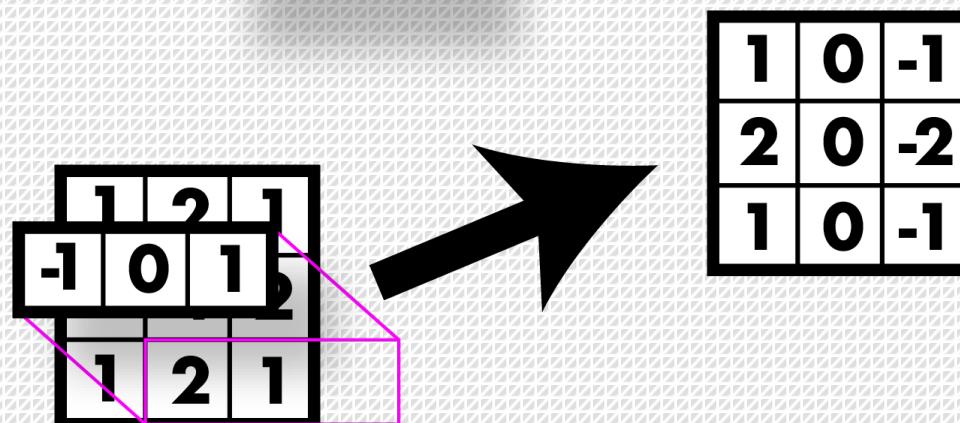


Lines

Source: <https://courses.cs.washington.edu/courses/cse576/23sp/notes.html>



$$\frac{1}{2} \times \left(\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$





Introduce the top scientist in CV

- Fei-Fei Li, Professor at Stanford University



[TED Talk: How we teach computers to understand pictures](#)

<https://profiles.stanford.edu/fei-fei-li>