Image processing operators (2)

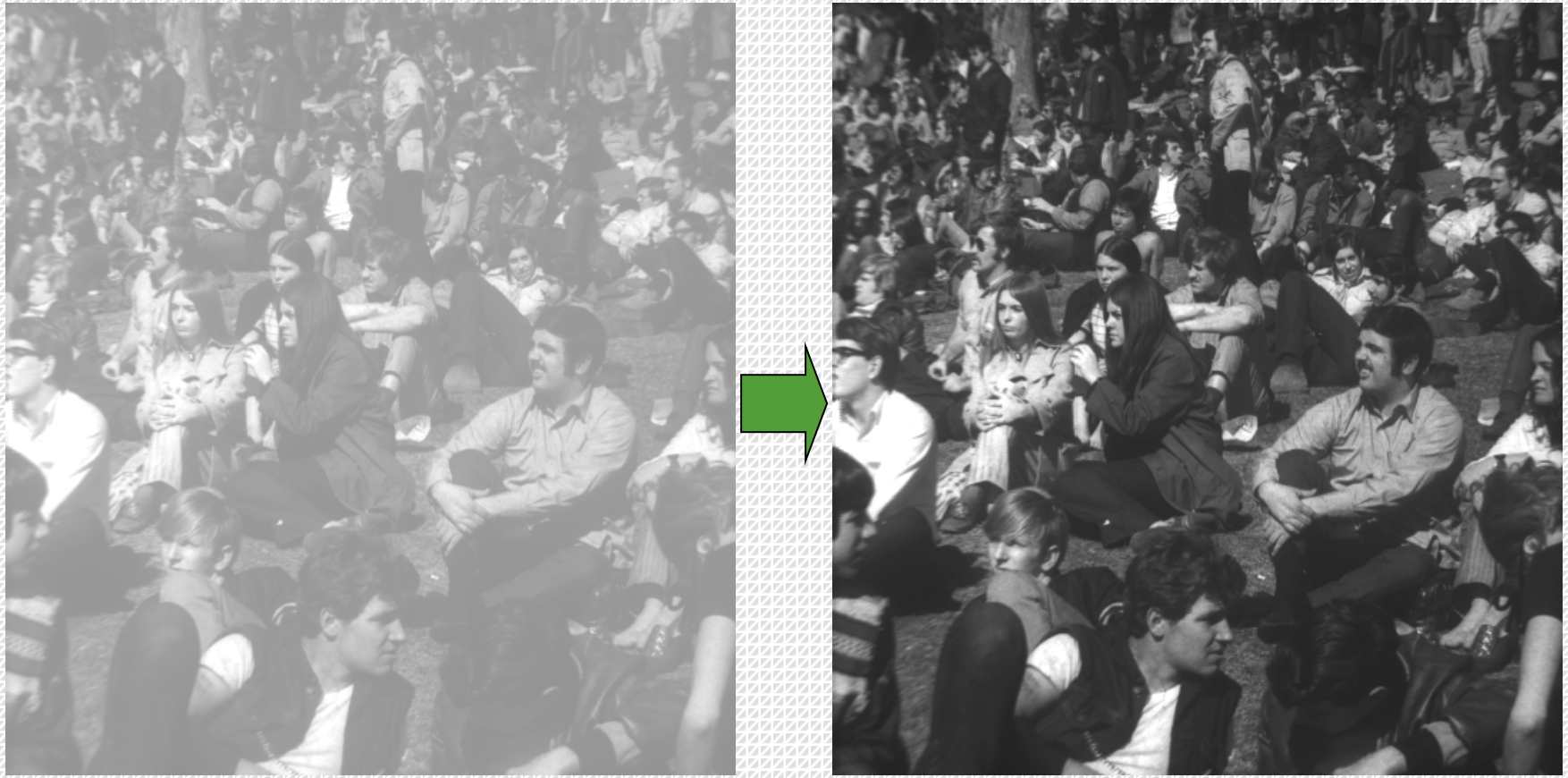
Bing Gong
（巩冰）
gongbing@shnu.edu.cn

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

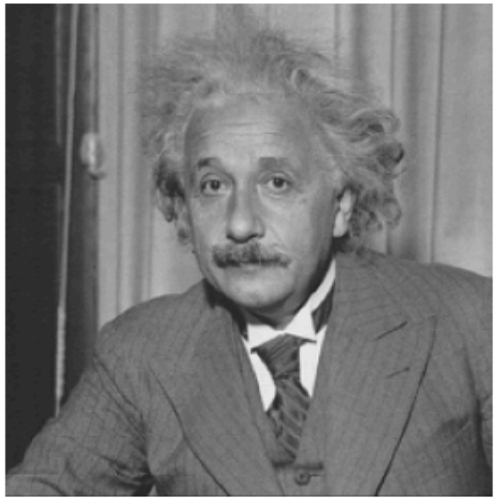
Figure 3.1 Some common image processing operations: (a) original image; (b) increased contrast; (c) change in hue; (d) "posterized" (quantized colors); (e) blurred; (f) rotated.

Why?



Why?

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

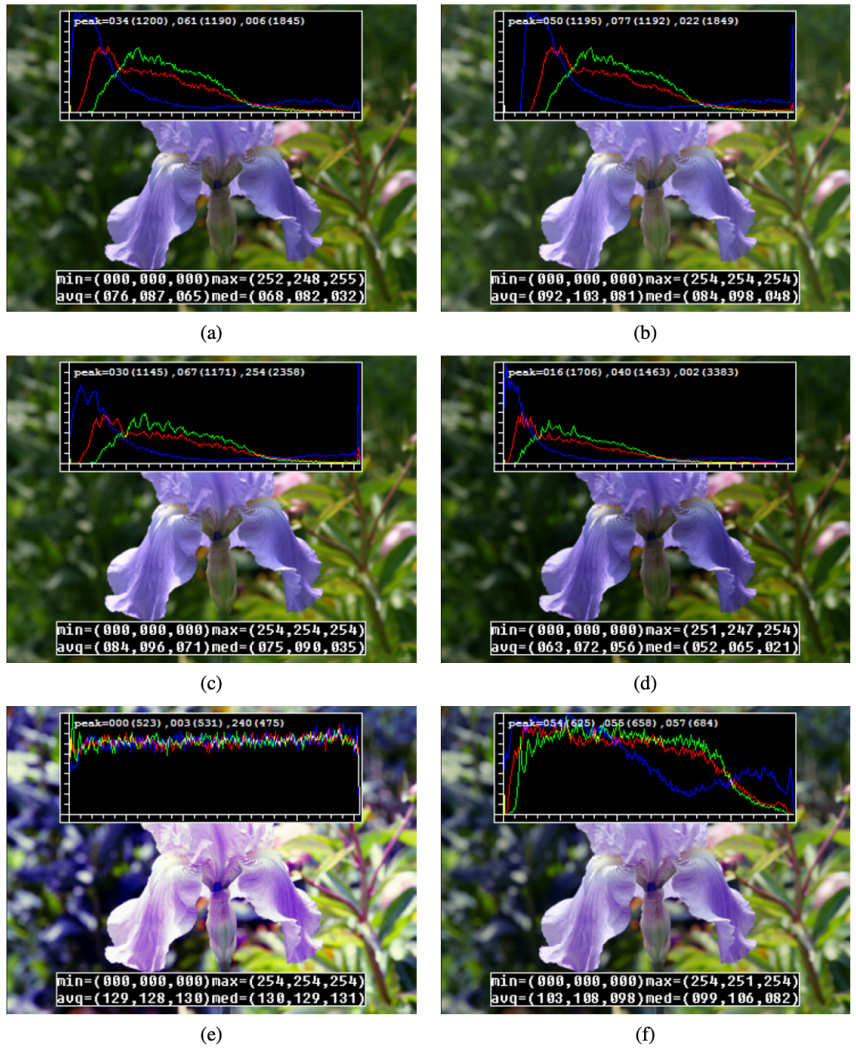


Figure 3.2 Some local image processing operations: (a) original image along with its three color (per-channel) histograms; (b) brightness increased (additive offset, $b = 16$); (c) contrast increased (multiplicative gain, $a = 1.1$); (d) gamma (partially) linearized ($\gamma = 1.2$); (e) full histogram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

Point operators

- **Pixel transforms:**

$$g(x) = h(f(x)) \; or \; g(x) = h(f_0(x), \ldots, f_n(x))$$

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:**

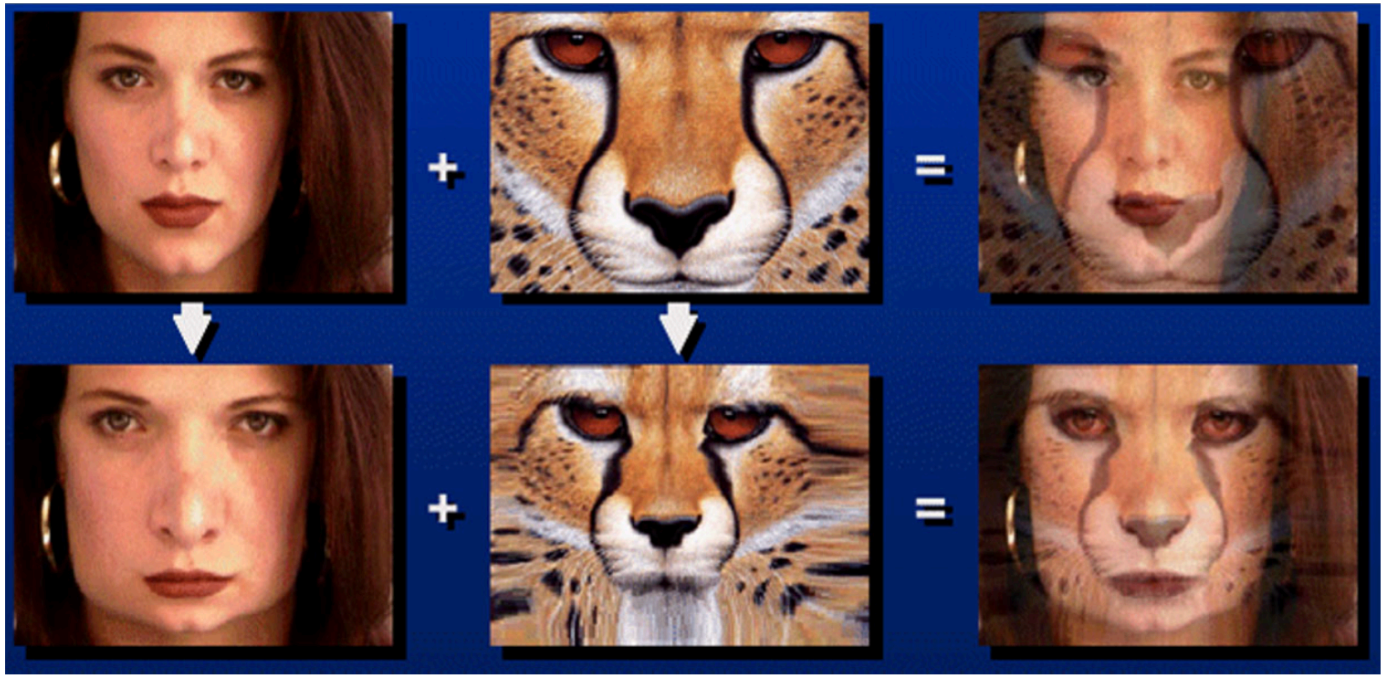
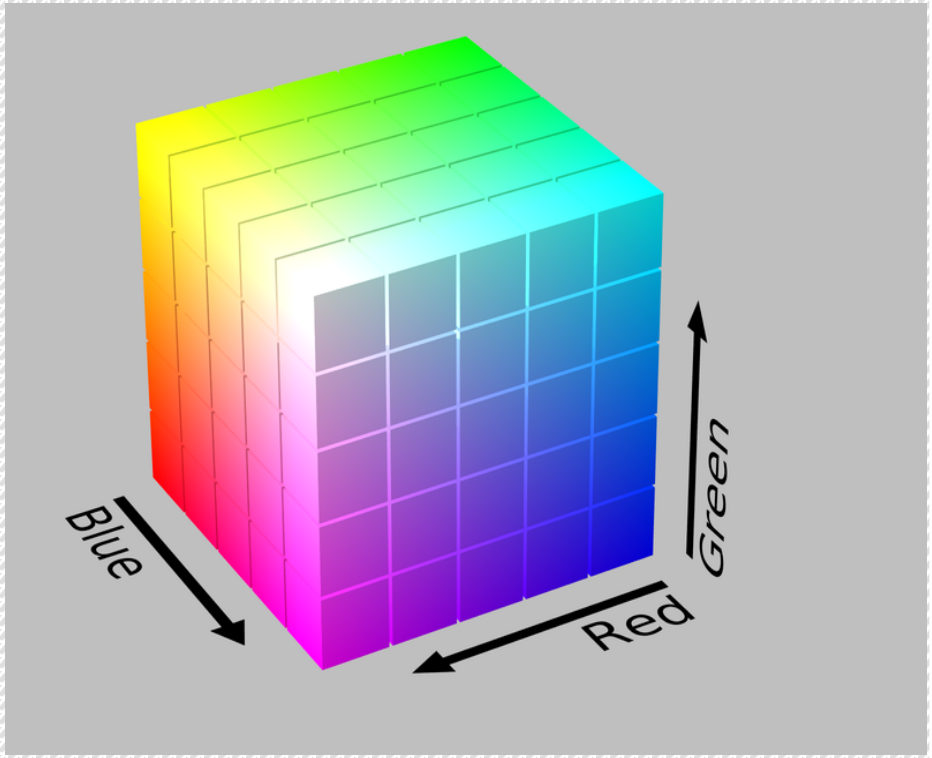
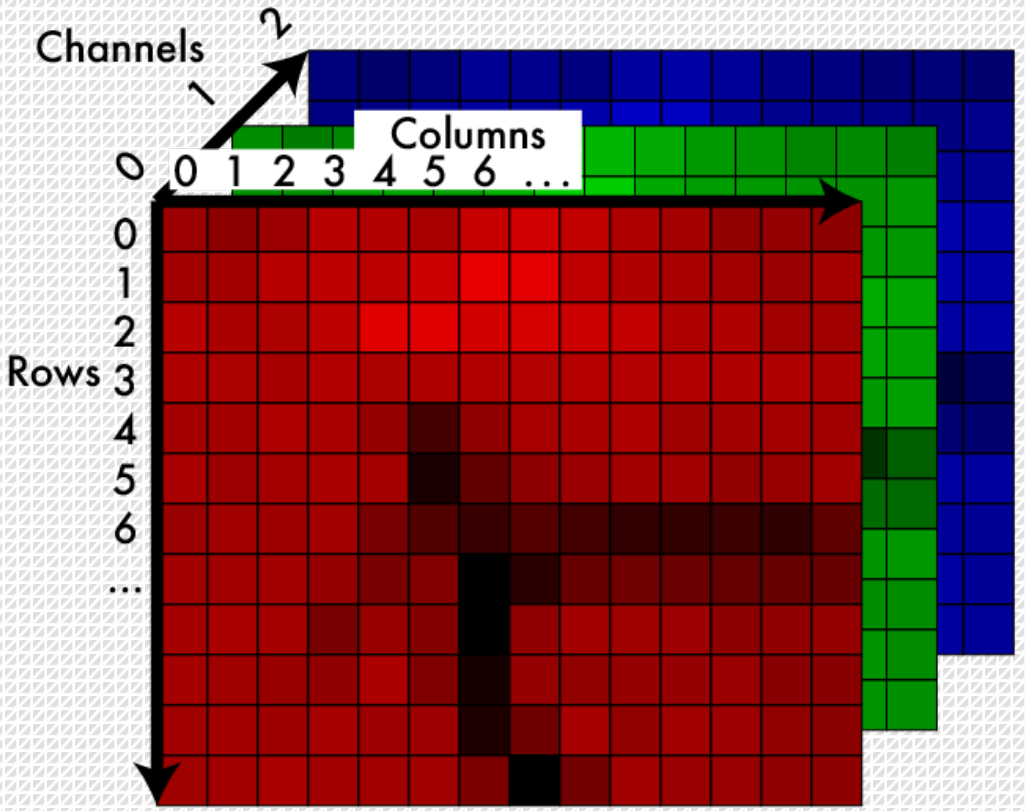


Figure 3.53 Image morphing (Gomes, Darsa, Costa *et al.* 1999) © 1999 Morgan Kaufmann. Top row: if the two images are just blended, visible ghosting results. Bottom row: both images are first warped to the same intermediate location (e.g., halfway towards the other image) and the resulting warped images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

Point operators

- **Color transforms:** RGB is a cube…



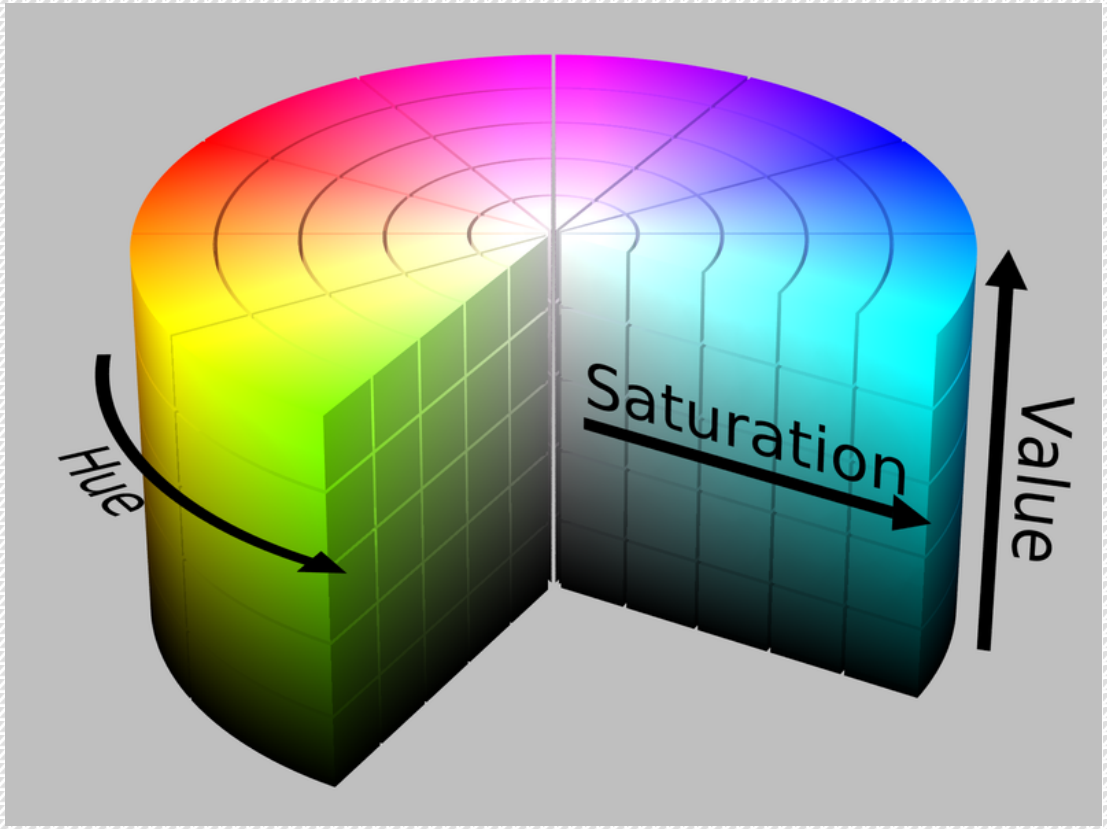


Point operators

- **Color transforms:** RGB is a cube...
- Question: How to improve the brightness of the image based on RGB?

Point operators

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



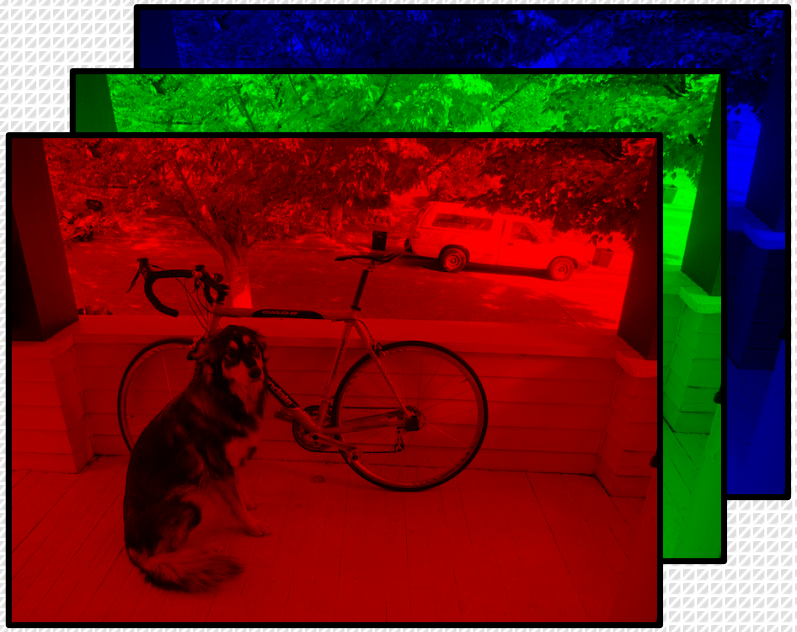
Point operators

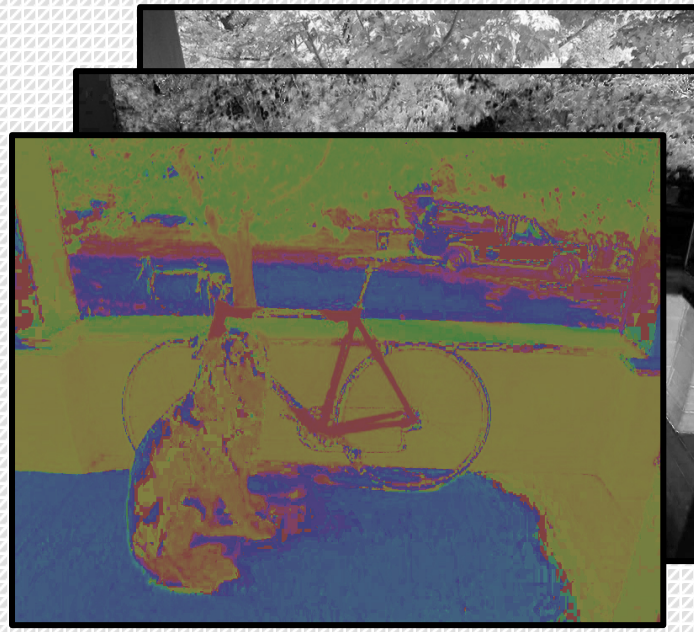
- **Color transforms:** Hue, Saturation, Value: cylinder!

  - 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

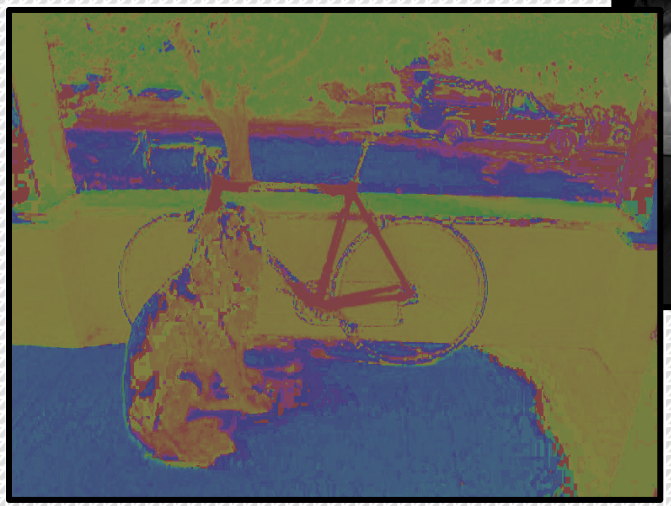




Point operators

- **Color transforms:**

Hue

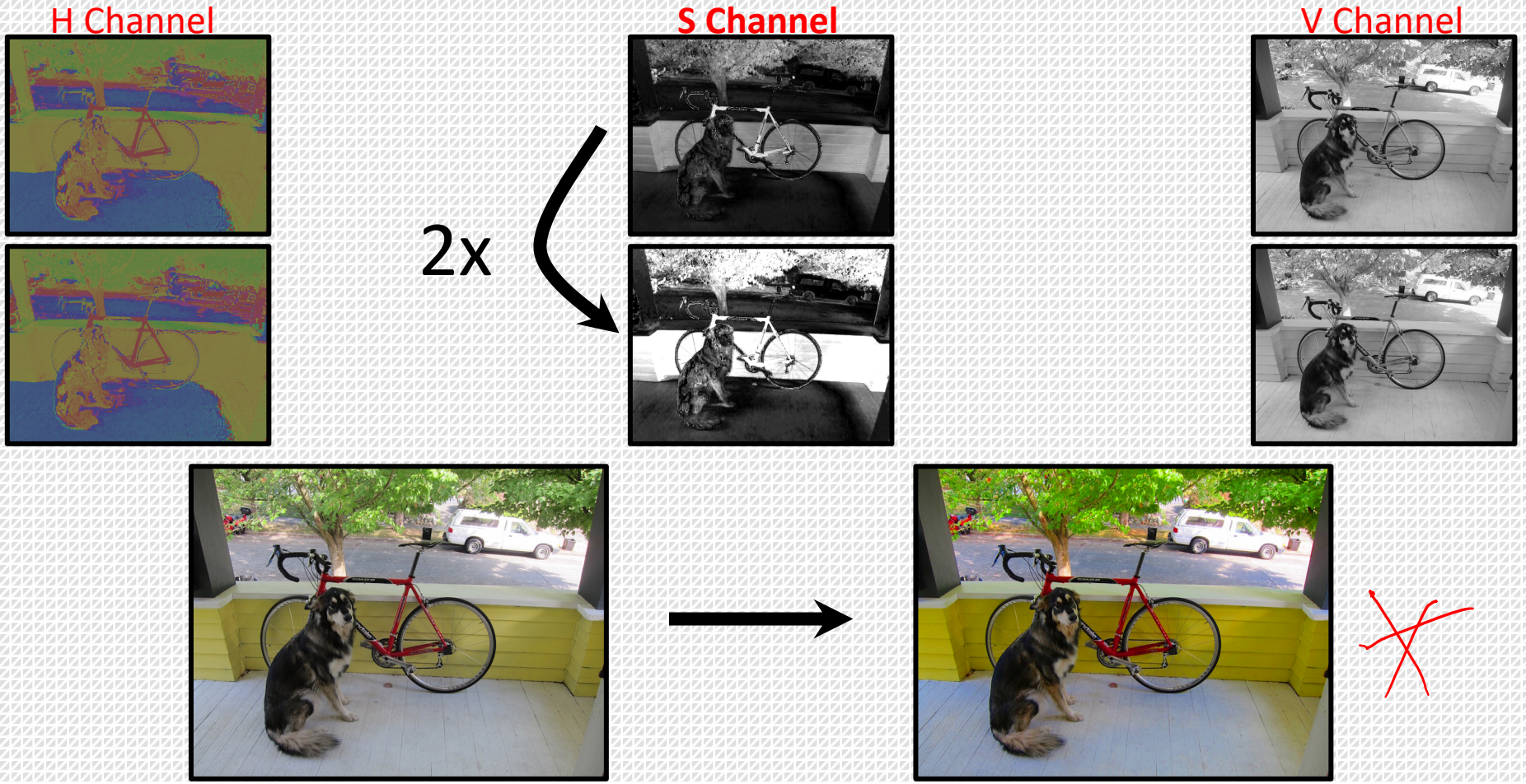


Saturation

Value

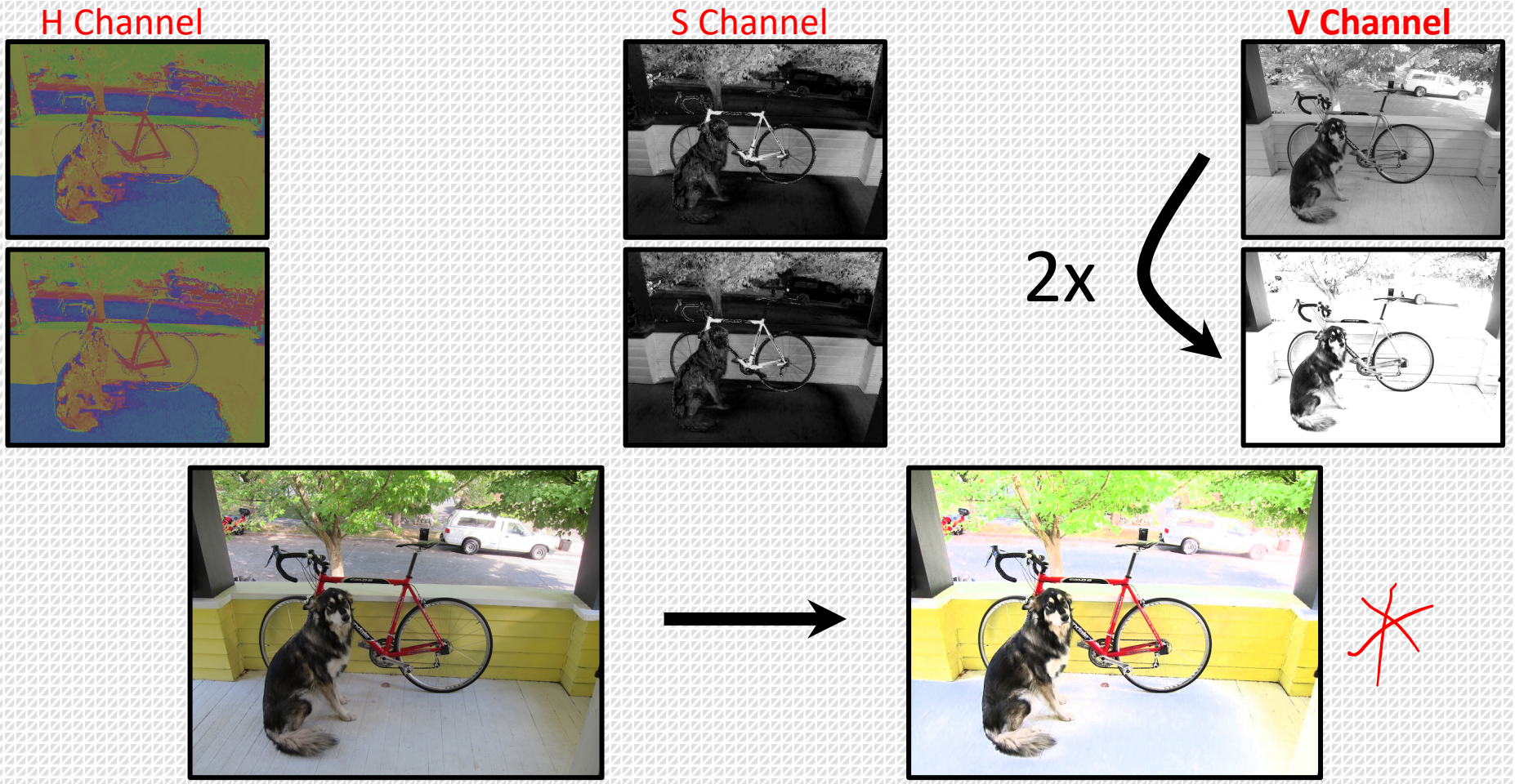
Point operators

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



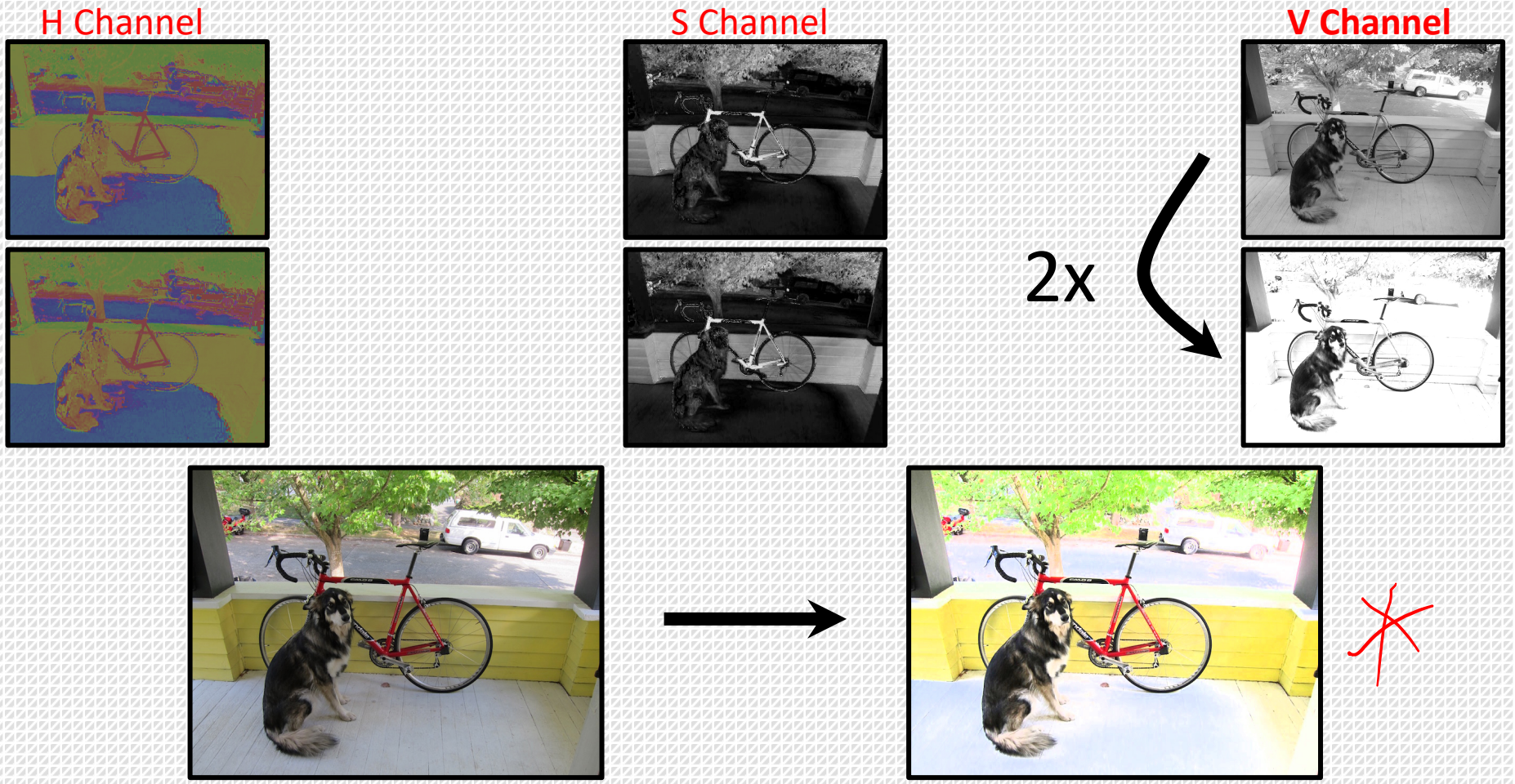
Point operators

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



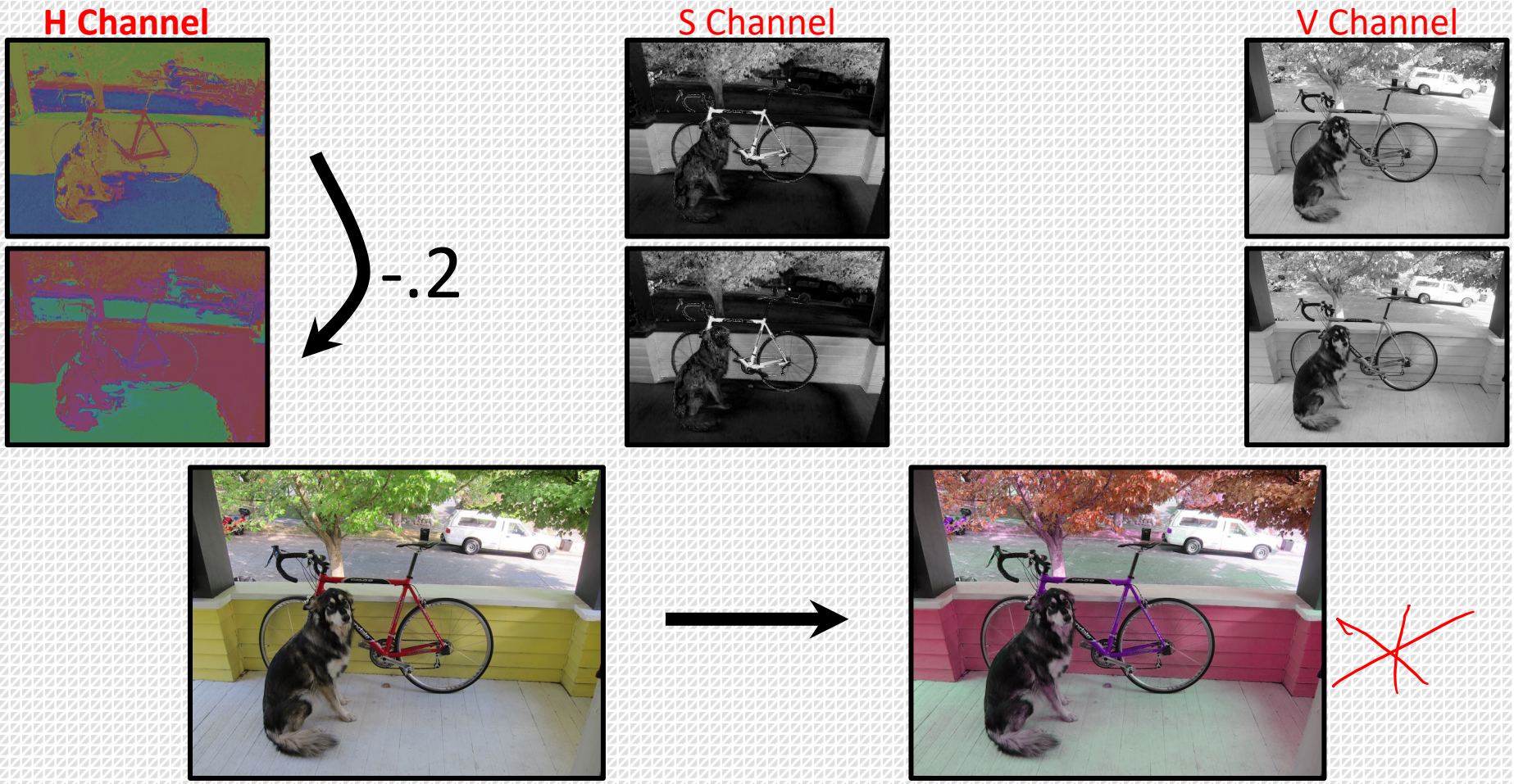
Point operators

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



Point operators

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



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 :**

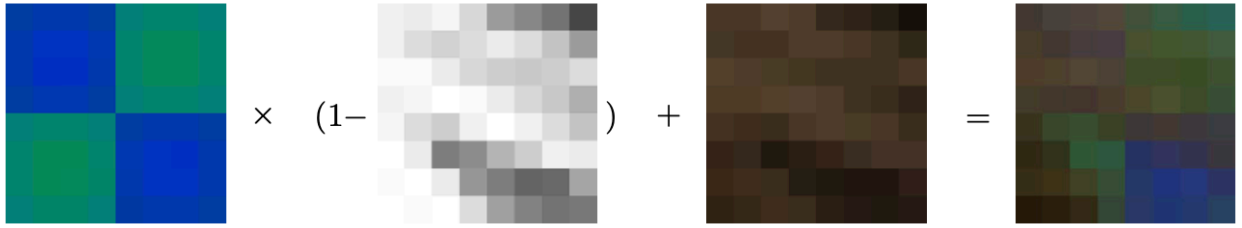
In many photo editing and visual effects applications, it is often desirable to cut a *foreground* object out of one scene and put it on top of a different *background.* The process of extracting the object from the original image is often called *matting,* while the process of inserting it into another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

Point operators

- **Compositing and matting :**

(a) (b) (c) (d)

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.



B α αF C

(a) (b) (c) (d)

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.

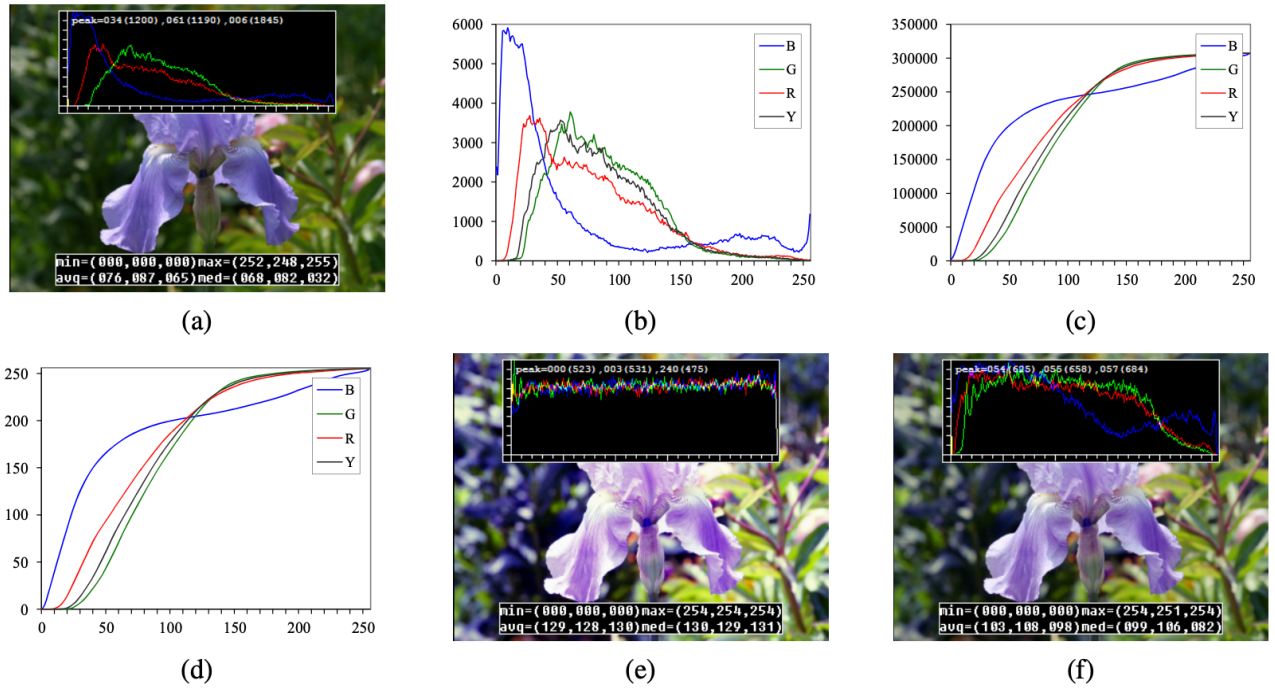


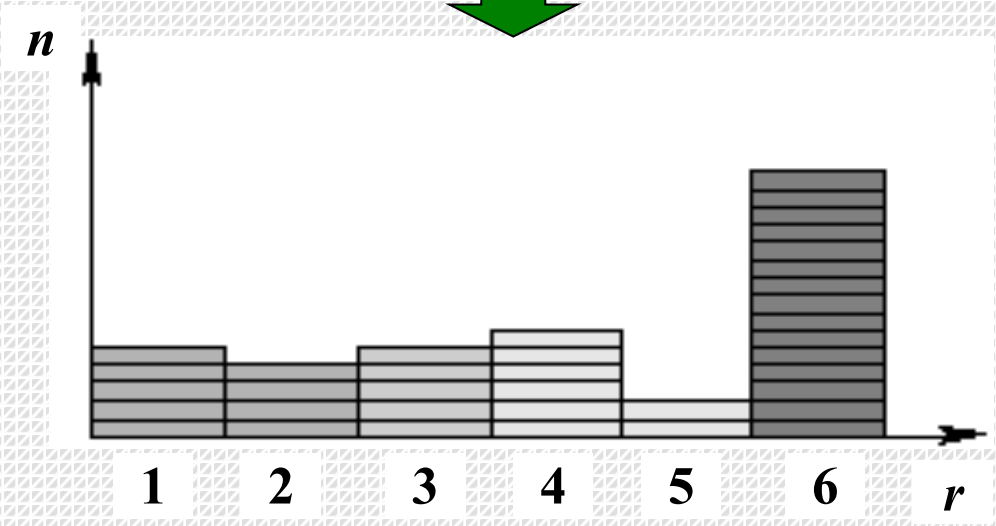
Figure 3.7 Histogram analysis and equalization: (a) original image (b) color channel and intensity (luminance) histograms; (c) cumulative distribution functions; (d) equalization (transfer) functions; (e) full histogram equalization; (f) partial histogram equalization.

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

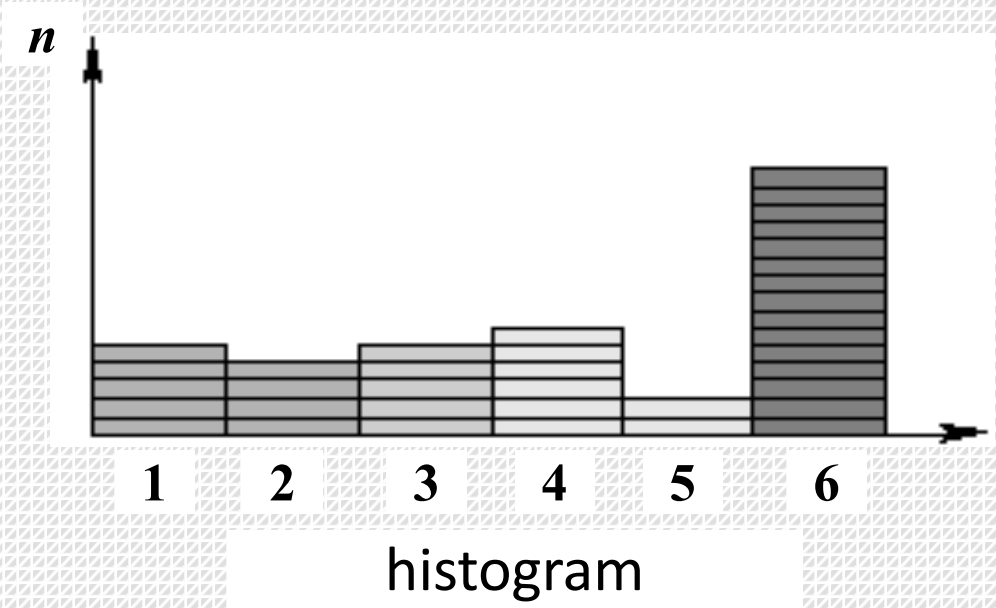
1	2	3	4	5	6
5	4	5	6	2	14



histogram

Point operators

- **What is histogram:**



histogram

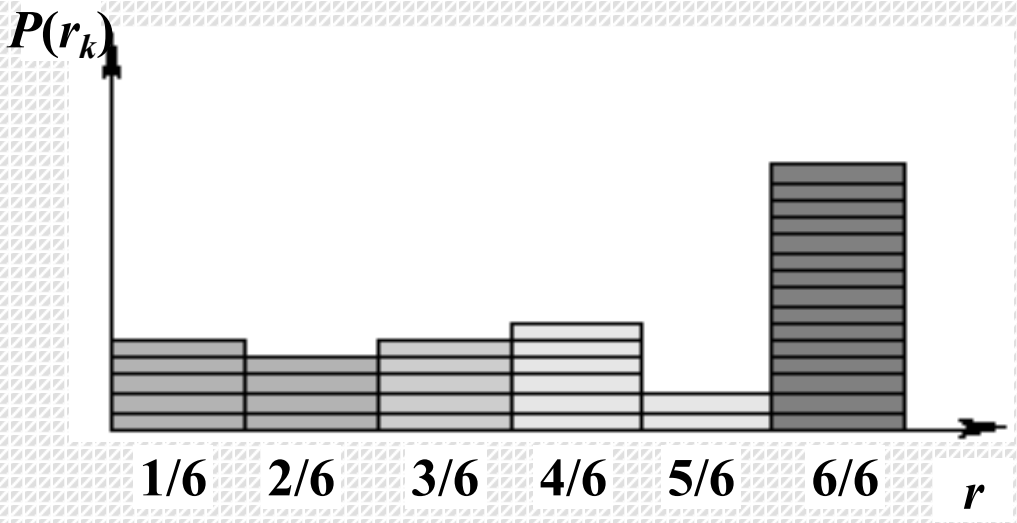
The histogram provides a visual representation of the distribution of the data, showing the number of observations that fall within each bin.

Point operators

- **What is density:**

$$P_r(r_k) = \frac{n_k}{n} \qquad 0 \le r_k \le 1$$

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

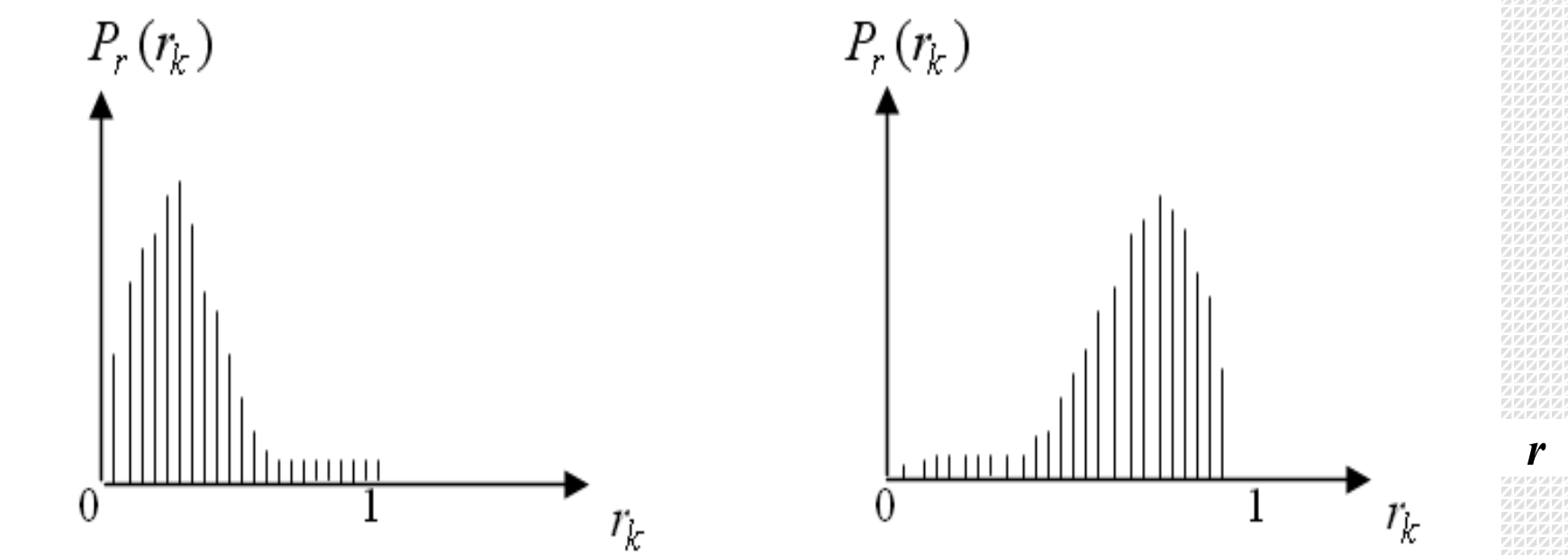


Density plot

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

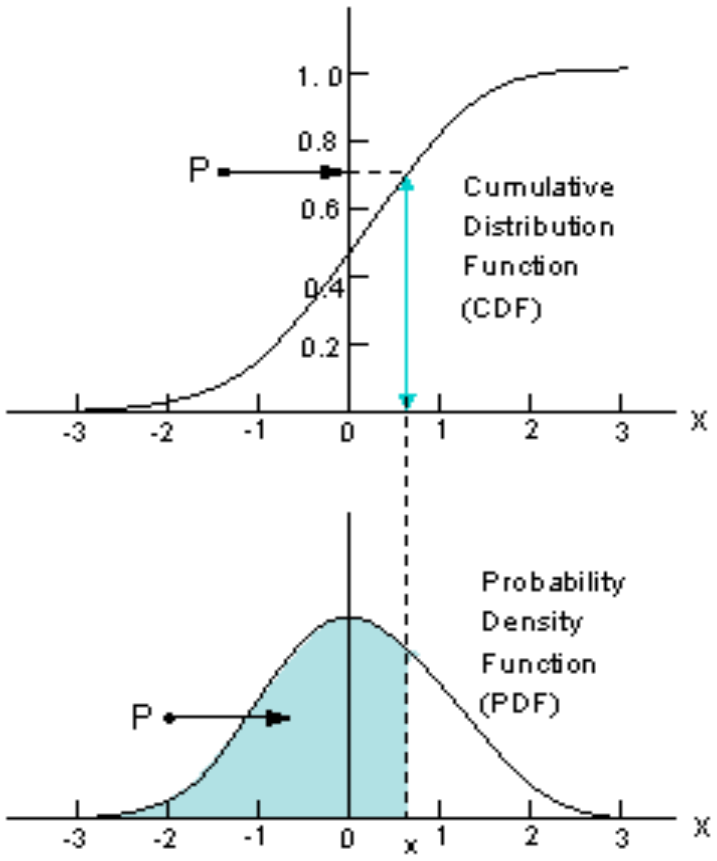
Point operators

- **What is density:**



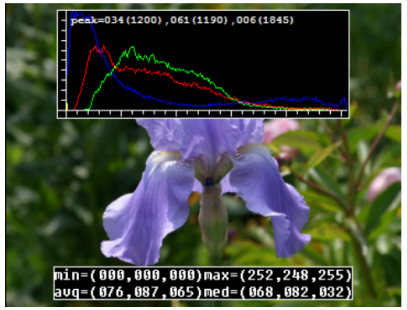
Point operators

- **What is cumulative distribution function :**

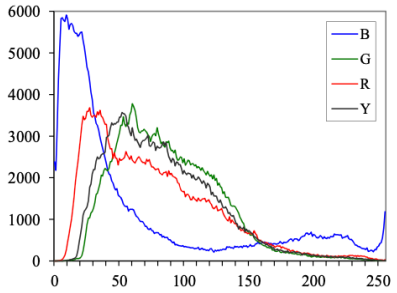


Relations Between Two Different Typical Representations of a Population

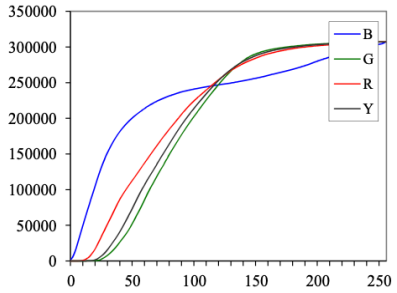
Point operators



(a)



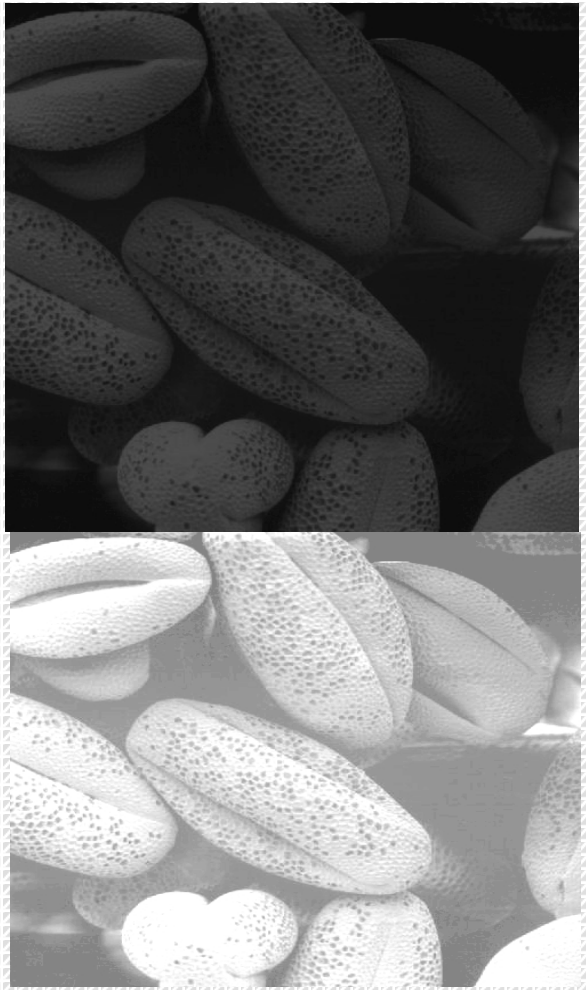
(b)

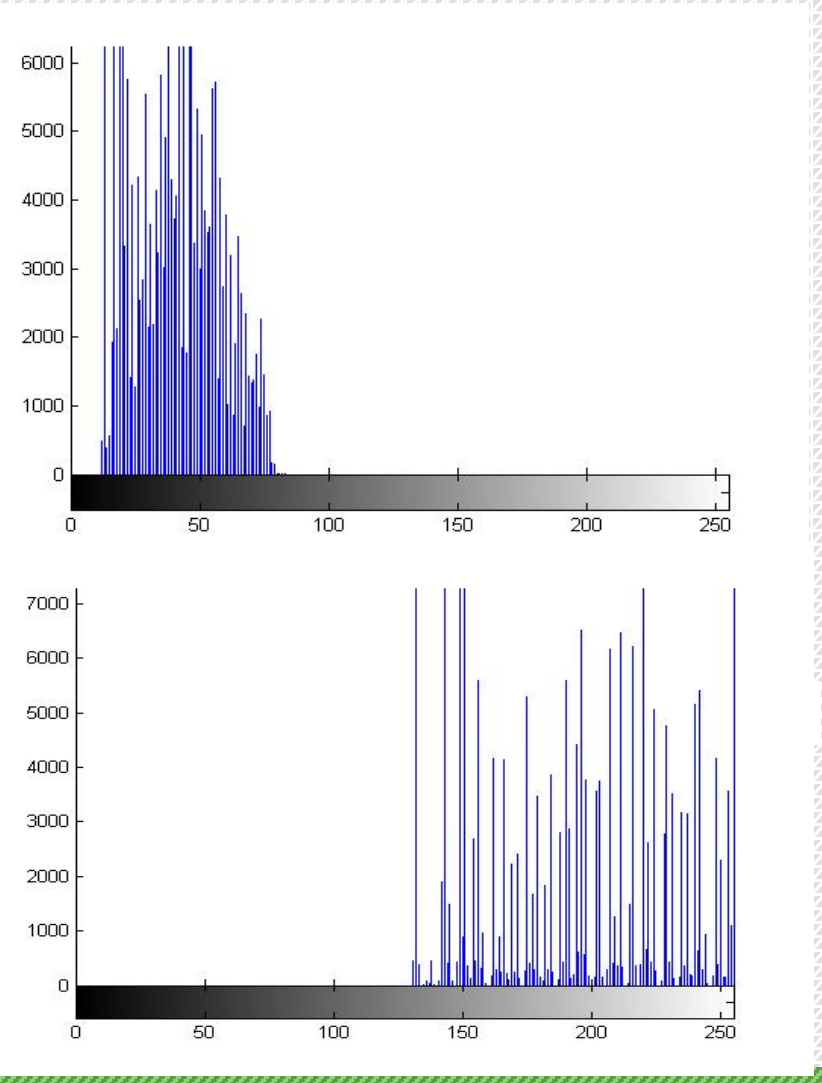


(c)

Point operators

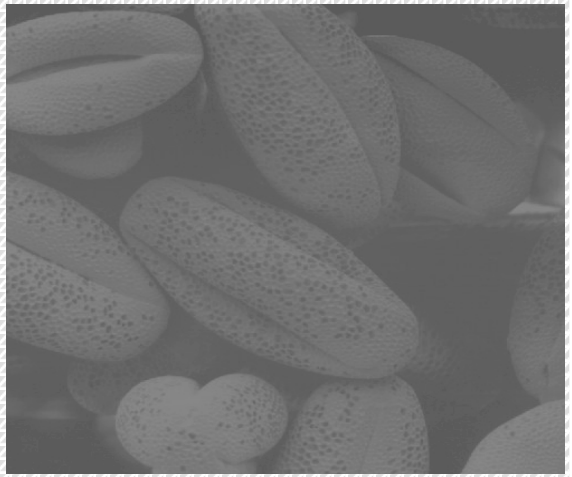
- **Histogram equalization:**

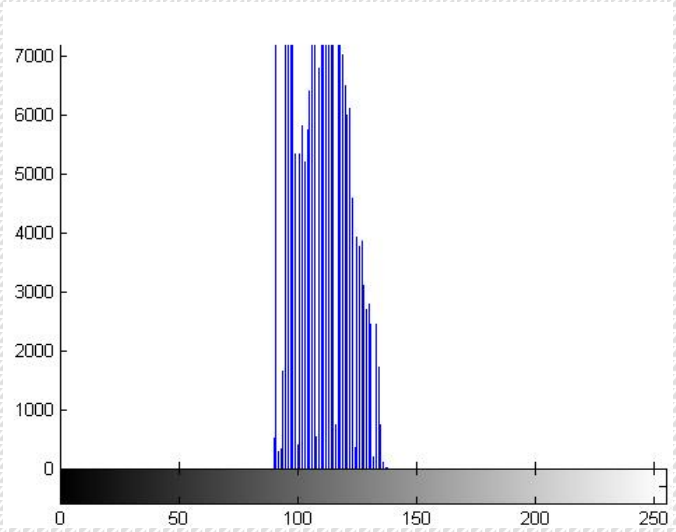


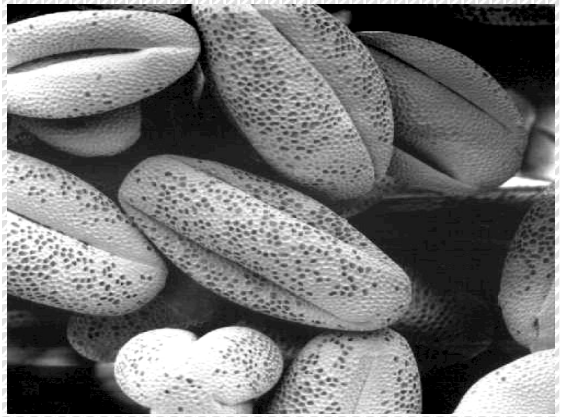


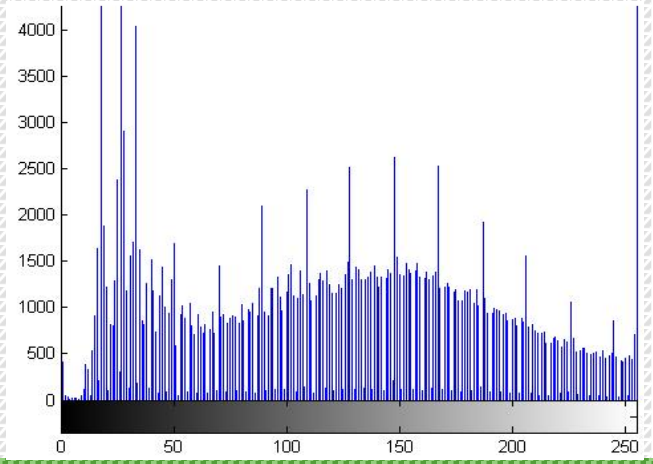
Point operators

- **Histogram equalization:**









Point operators

- **Histogram equalization:**

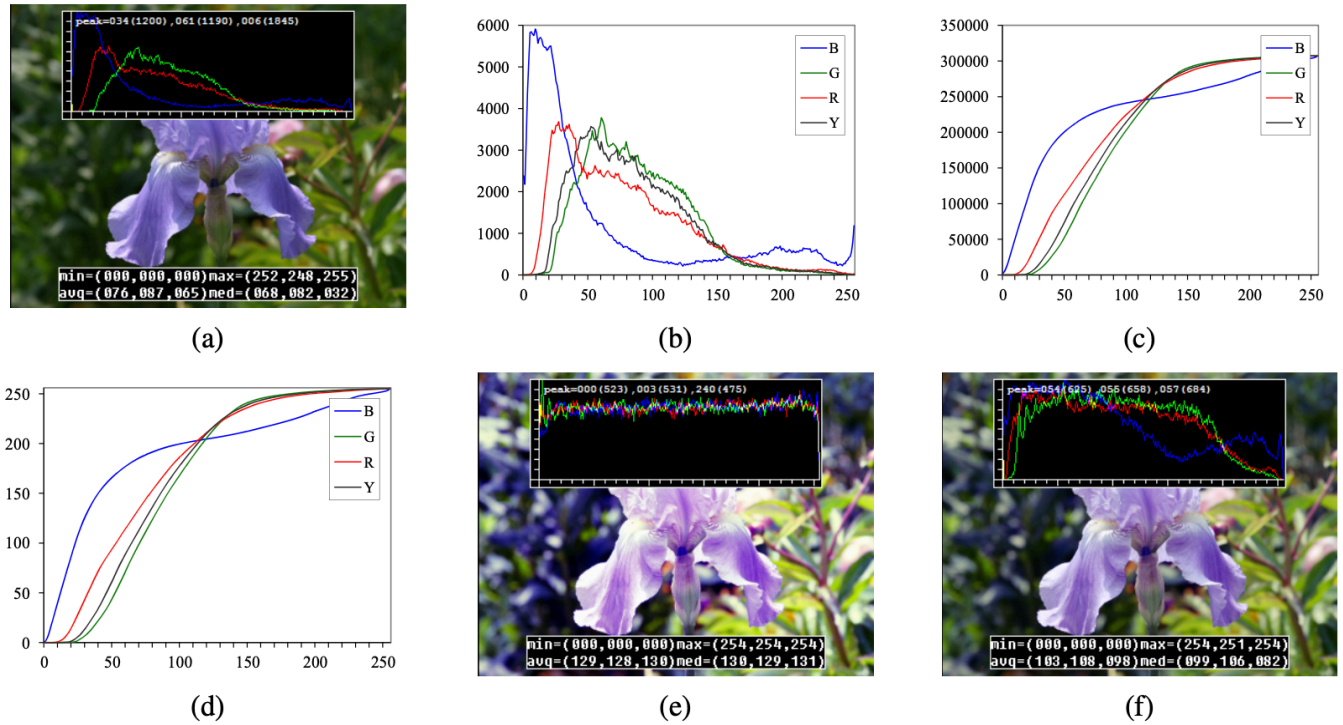
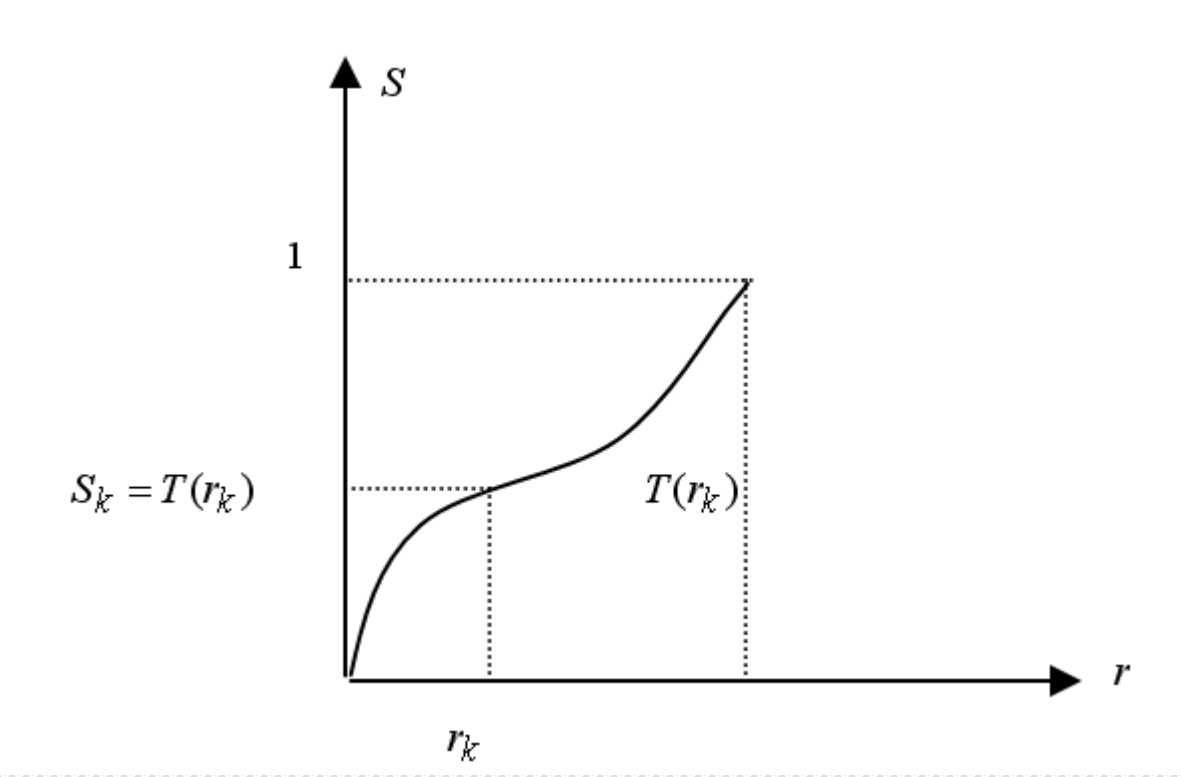


Figure 3.7 Histogram analysis and equalization: (a) original image (b) color channel and intensity (luminance) histograms; (c) cumulative distribution functions; (d) equalization (transfer) functions; (e) full histogram equalization; (f) partial histogram equalization.

Point operators

- **How to perform histogram equalization?**

$$s = T(r)$$



An example of histogram equalization function

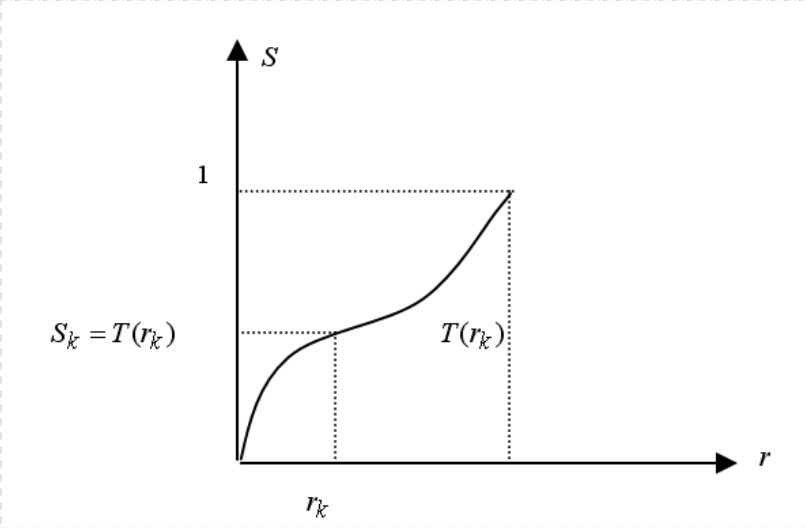
Point operators

- **How to perform histogram equalization?**

The transform function $\mathrm{T(r)}$ should follow the assumption :

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

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



Point operators

- **How to perform histogram equalization?**

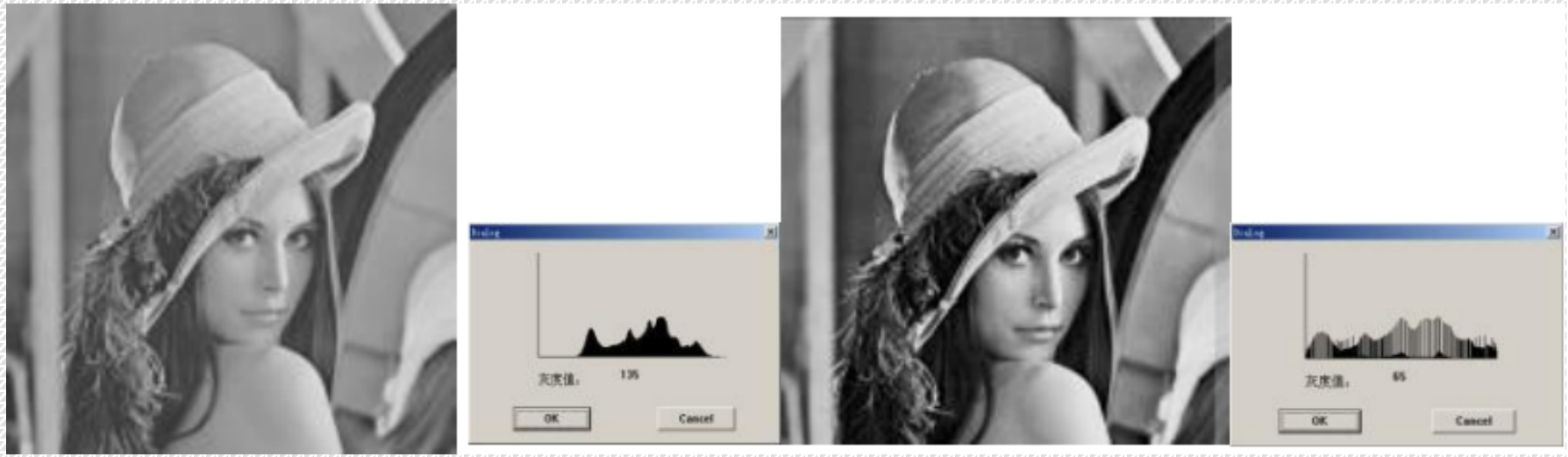
1	3	9	9	8
2	1	3	7	3
3	6	0	6	4
6	8	2	0	5
2	9	2	6	0

Pix	Ni	Pi=Ni/image	sumPi	sumPI * 256-1	Approximate
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:**

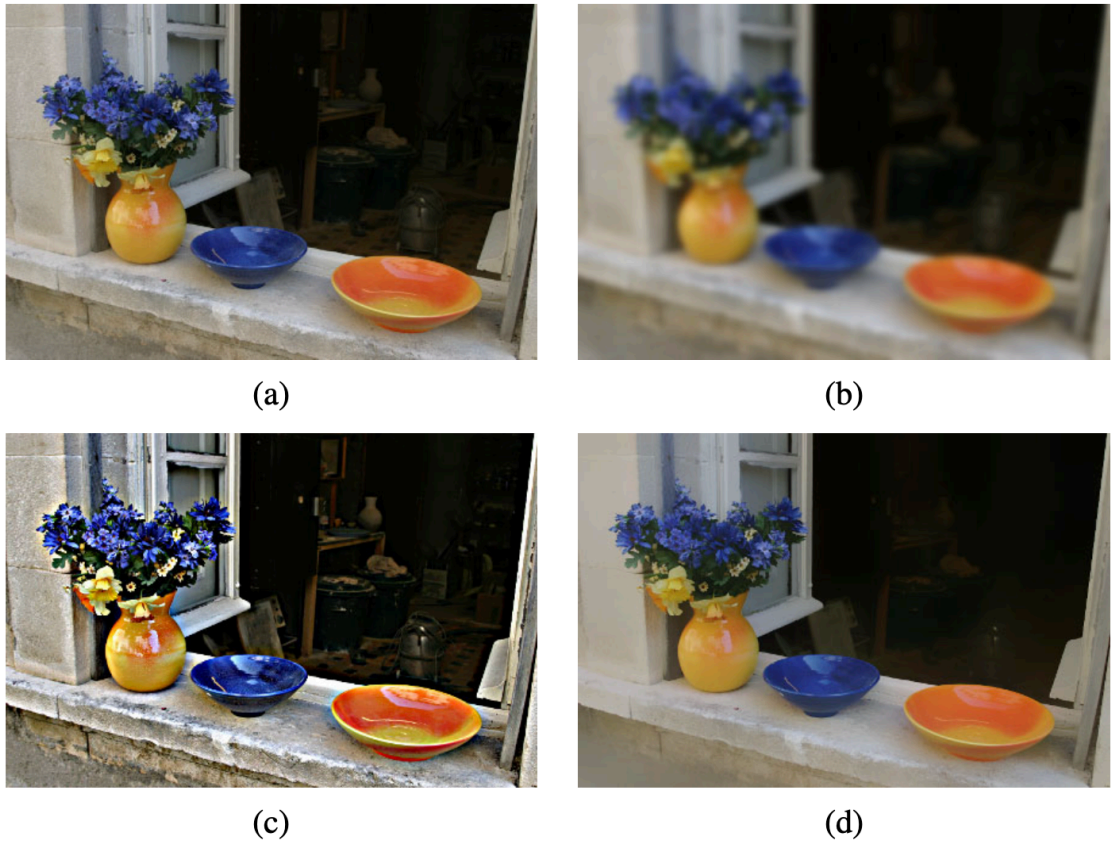


Methods of image transform

- *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)

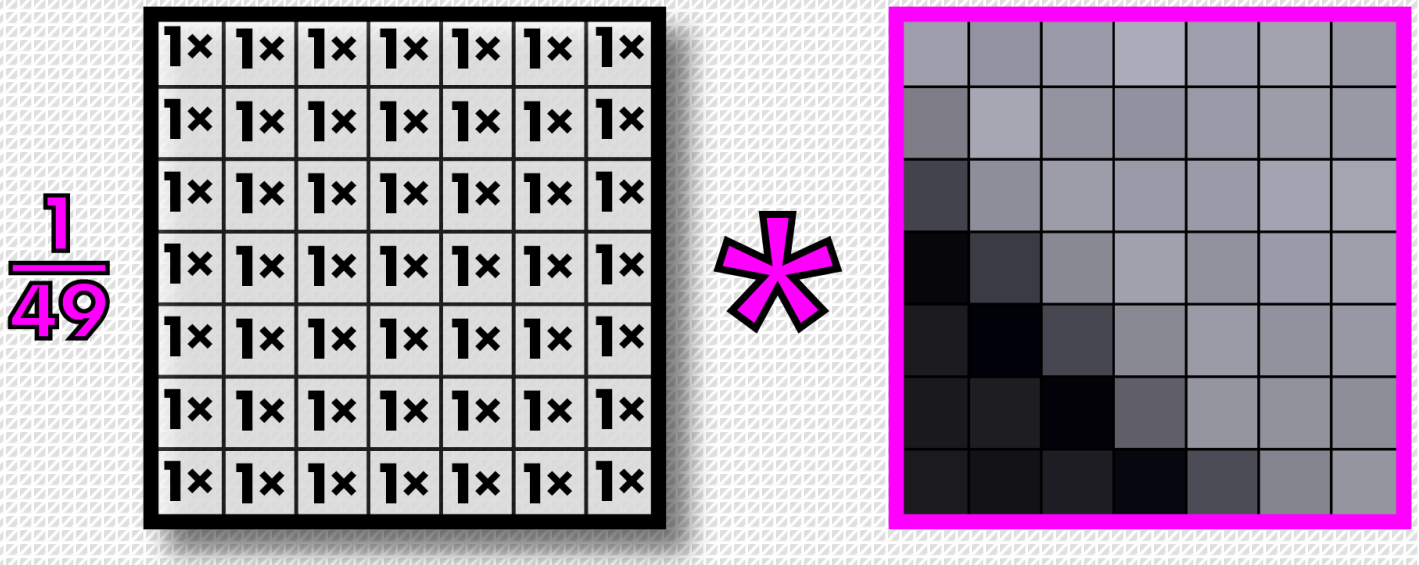
Neighborhood operators

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

- **What is the convolutional?**

Call this operation "*convolution*"

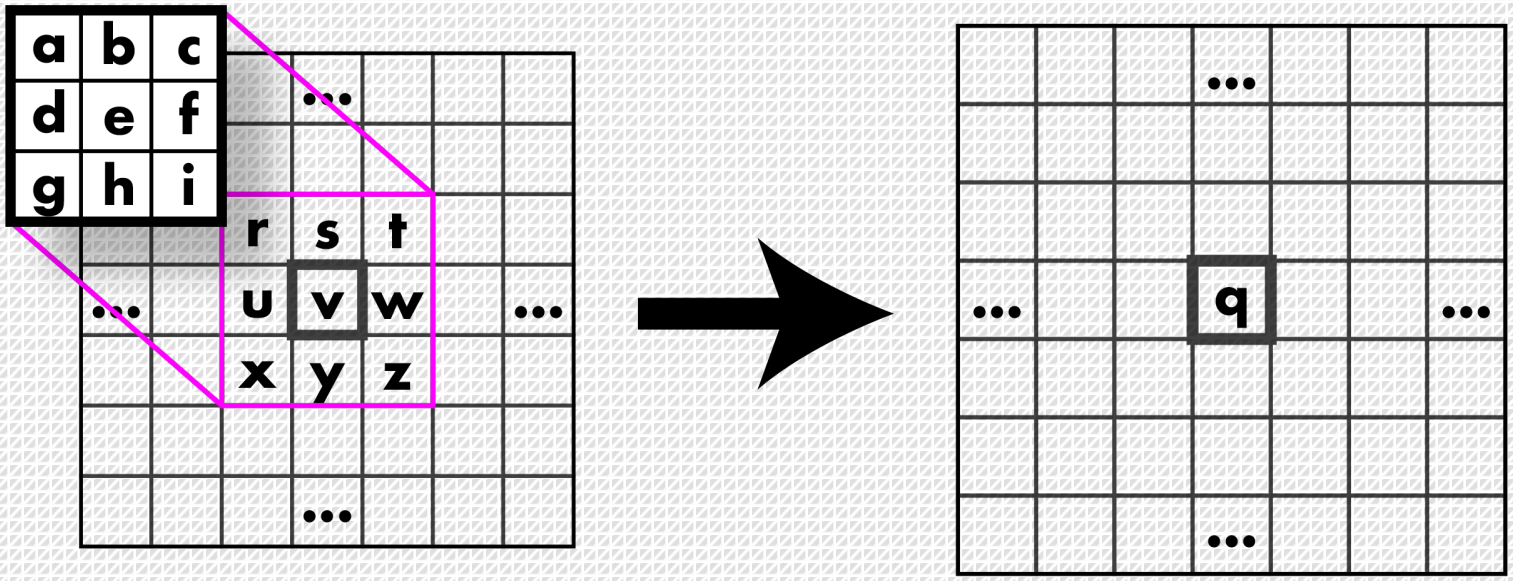
Filter or *kernel*



Neighborhood operators

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

- **How does the convolution work?**

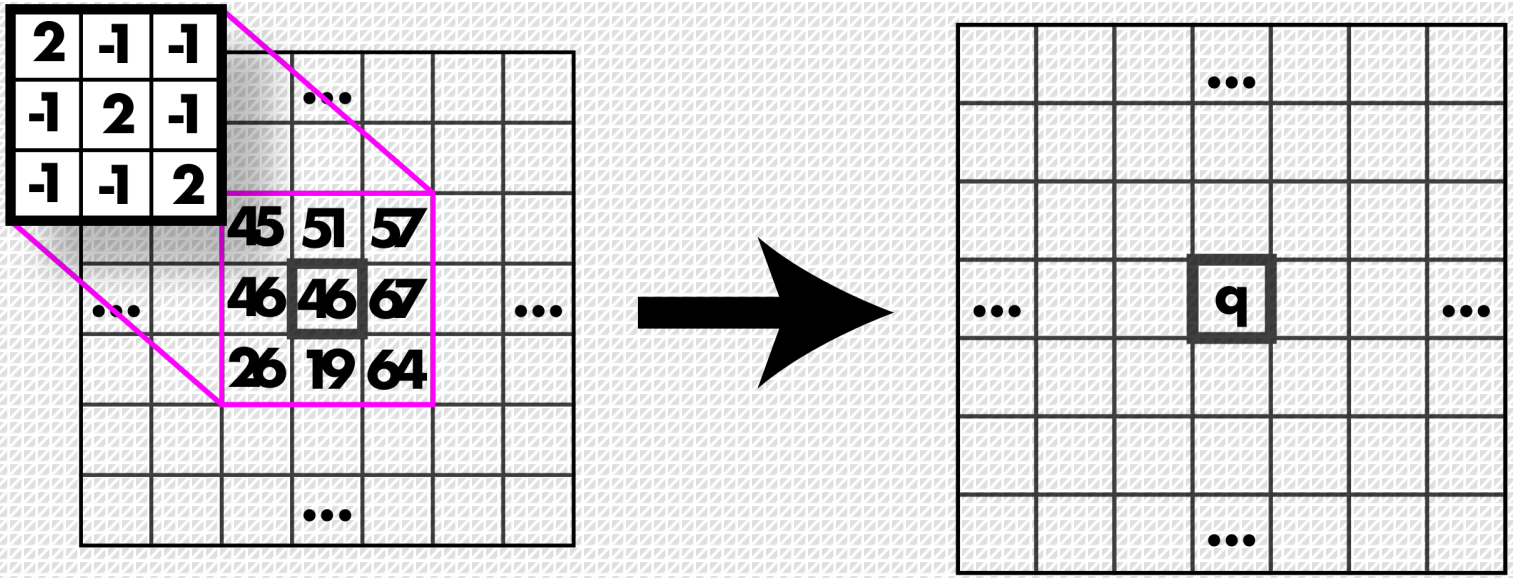


$$q = a \times r + b \times s + c \times t + d \times u + e \times v + f \times w + g \times x + h \times y + i \times z$$

Neighborhood operators

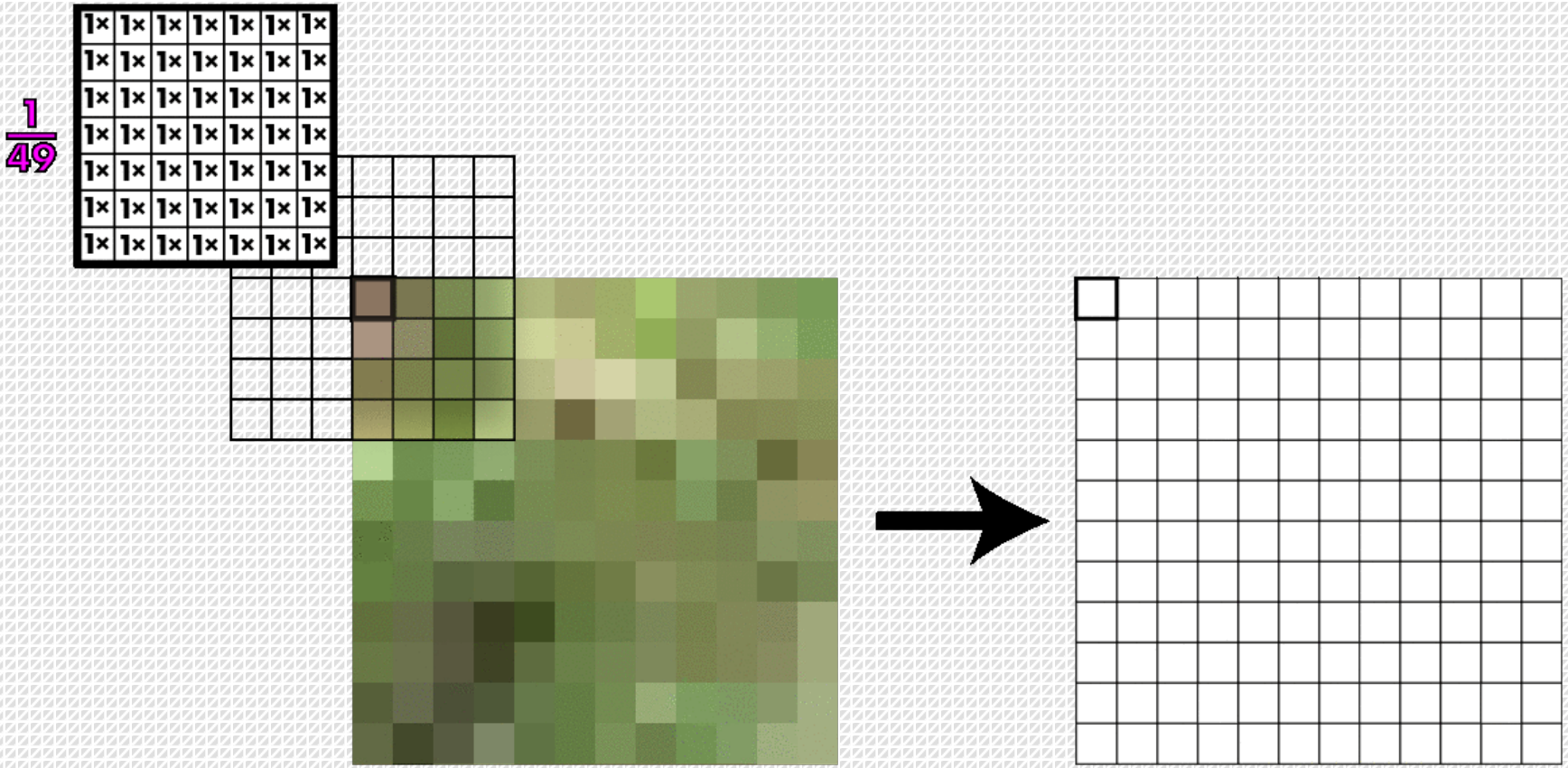
Source:
https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

- **How does the convolution work?**



Neighborhood operators

- Kernel slides across image



Neighborhood operators

- What happens at the edges of the images?

1. Zero padding: easiest but can give bad results

2. Duplicate the outermost row or column

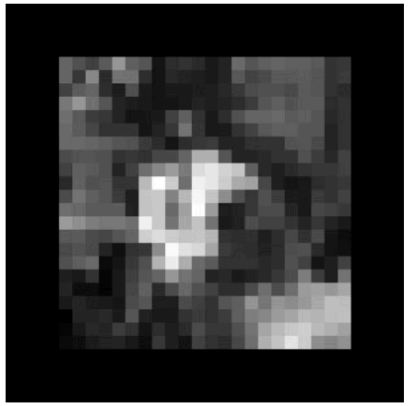
3. Use wraparound

boundary effects !

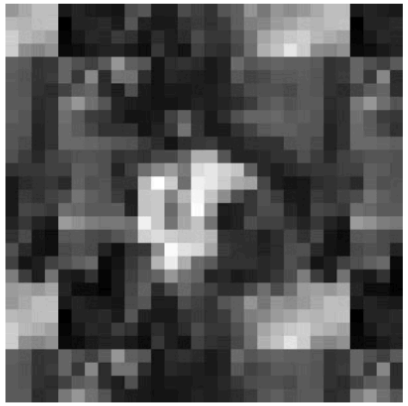
Neighborhood operators

- **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

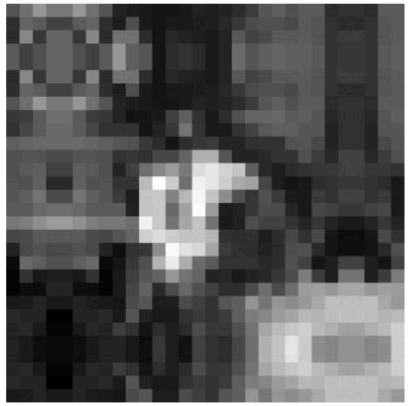


zero

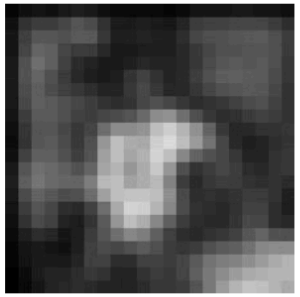


wrap

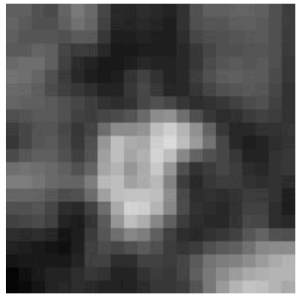
clamp



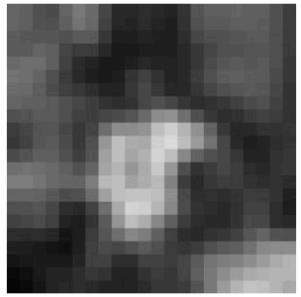
mirror



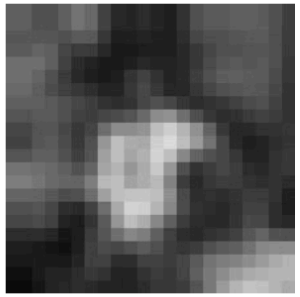
blurred zero



normalized zero



blurred clamp



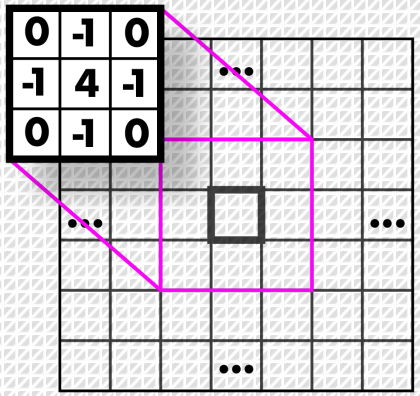
blurred mirror

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

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

- **Examples of kernels:**

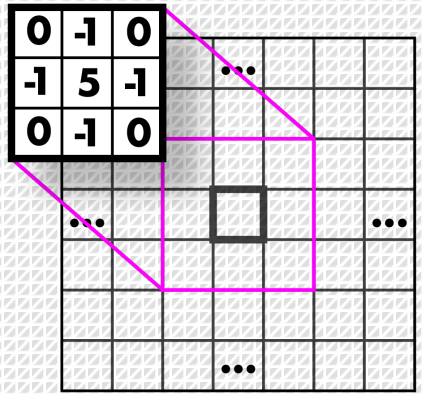


Highpass Kernel: finds edges

Neighborhood operators

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

- **Examples of kernels:**



Sharpen Kernel: sharpens!

Neighborhood operators

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

- **Examples of kernels:**

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Sobel Kernels: edges

Neighborhood operators

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

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

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.

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

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

Neighborhood operators

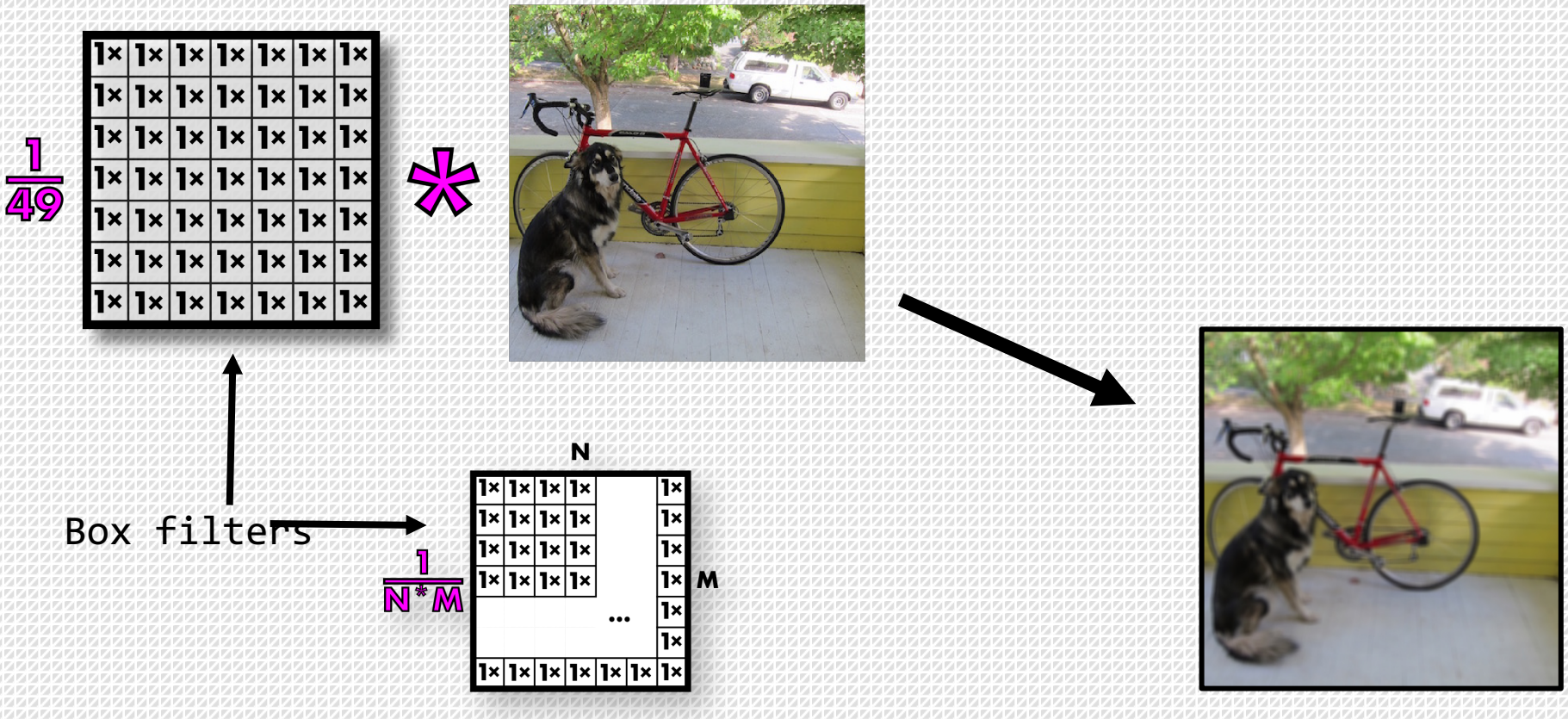
- **Linear operators:** which involve weighted combinations of pixels in small neighborhoods.

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

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

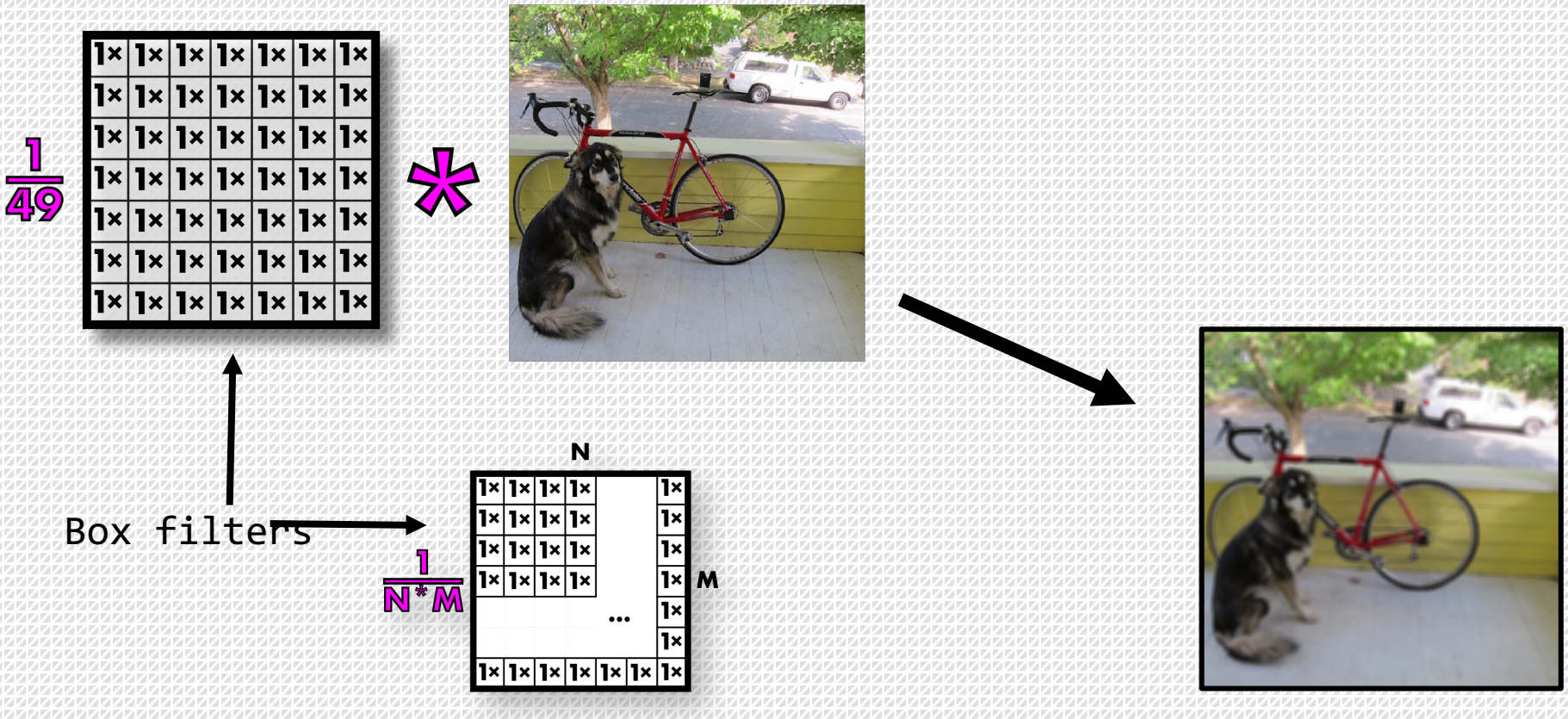
Neighborhood operators

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



Neighborhood operators

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



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

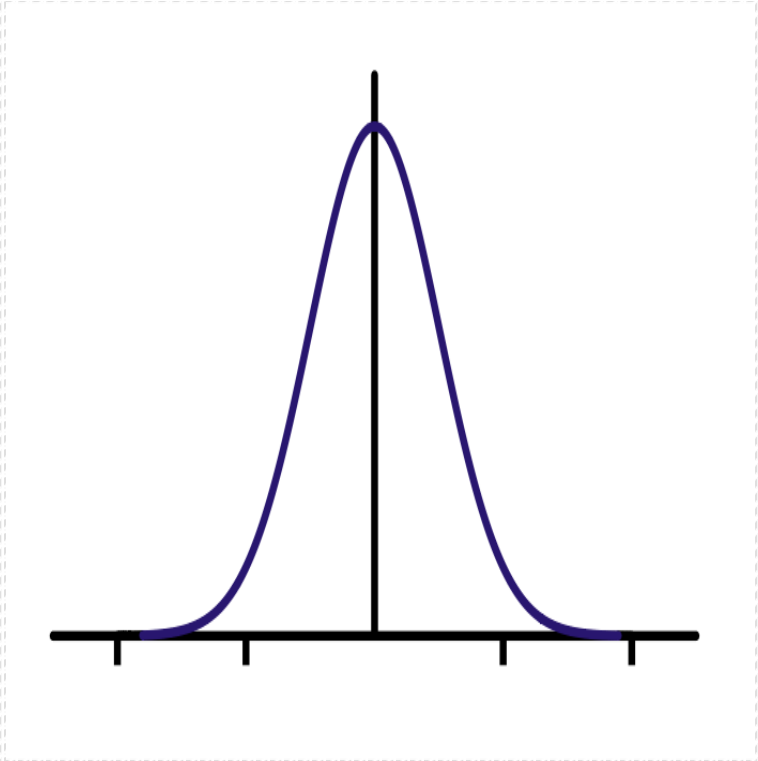
- *Moving average* or *box* filter – generate artifact

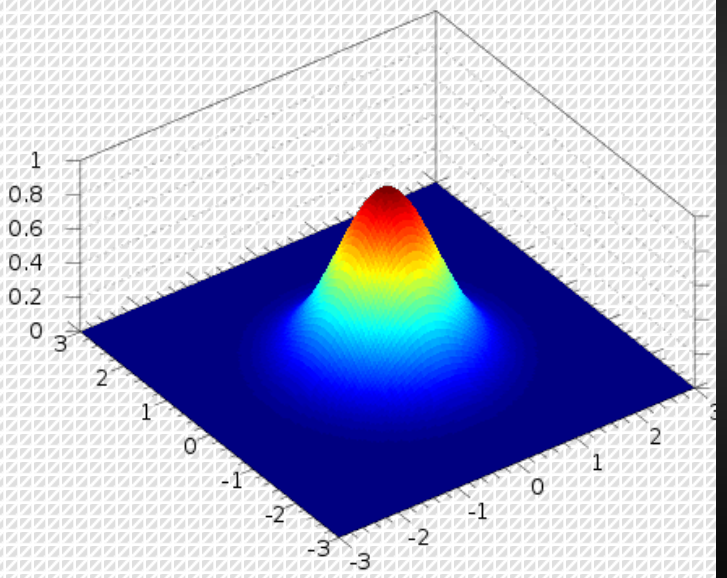
Neighborhood operators

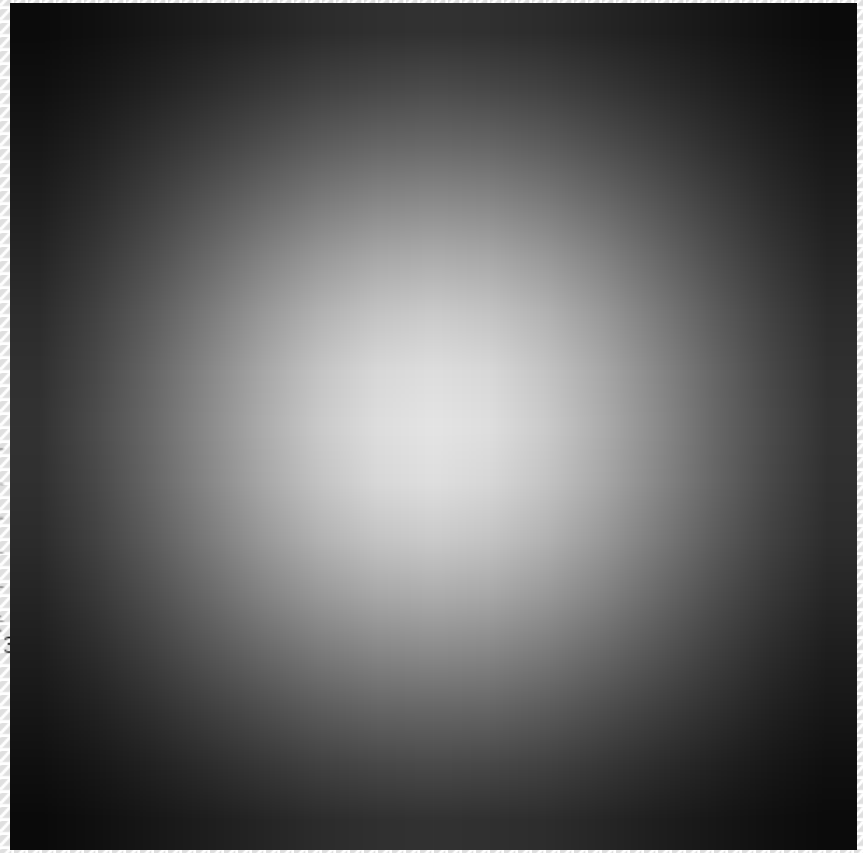
Neighborhood operators

- *A Smoothly* weighted kernel

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



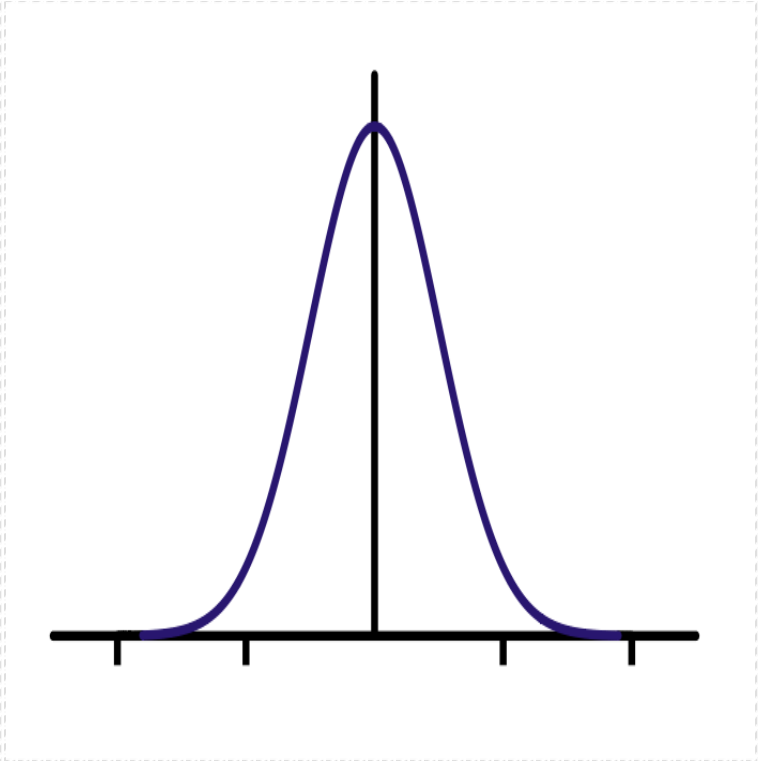


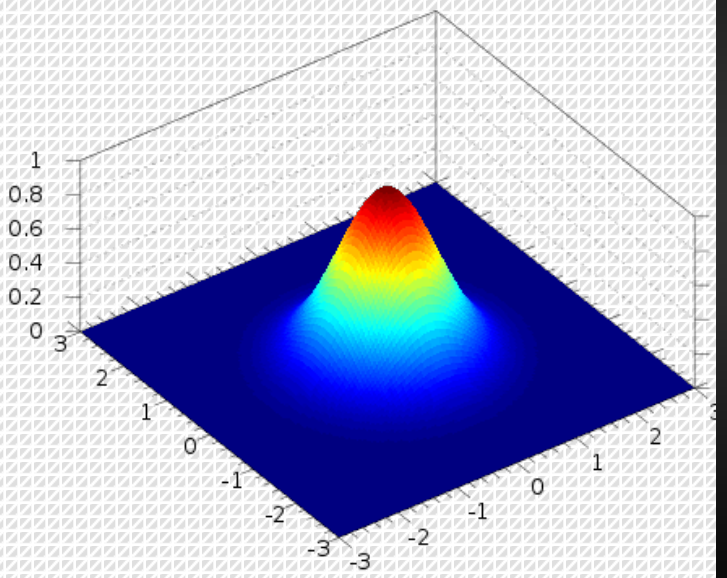


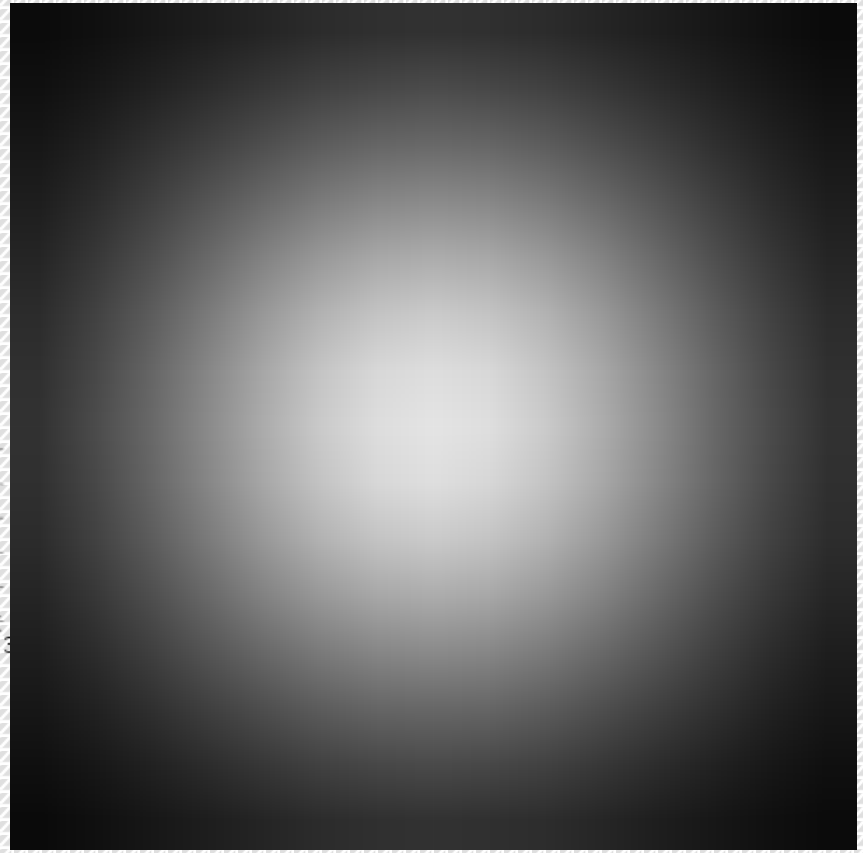
Neighborhood operators

- *A Smoothly* weighted kernel

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

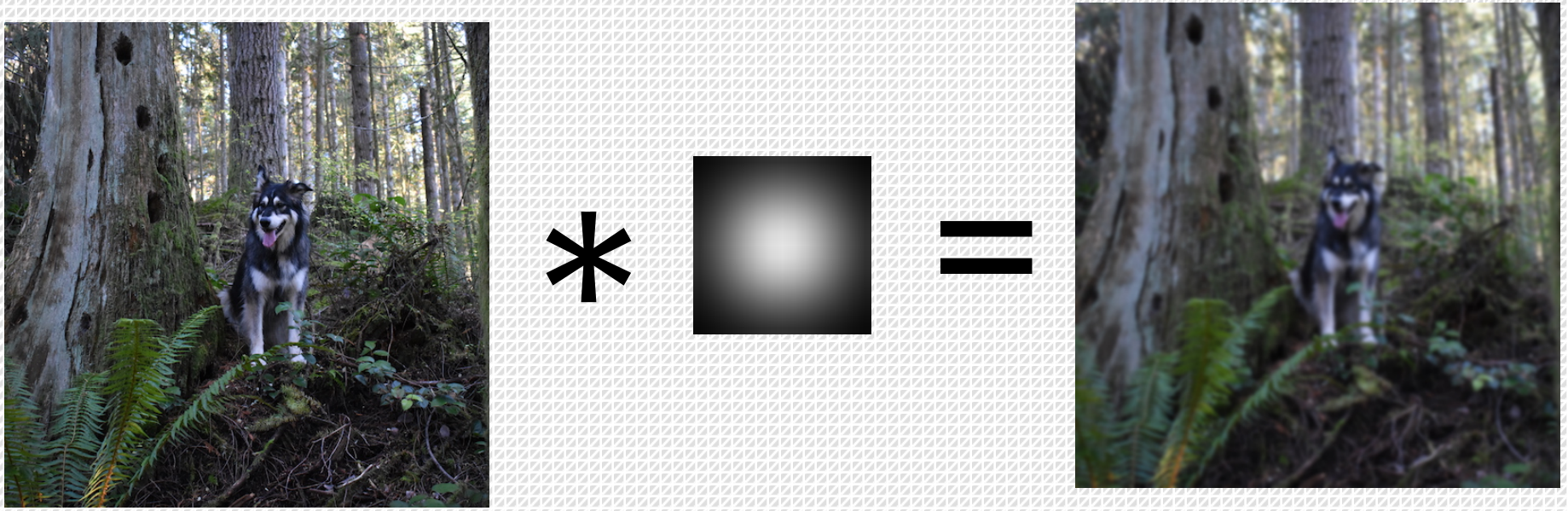






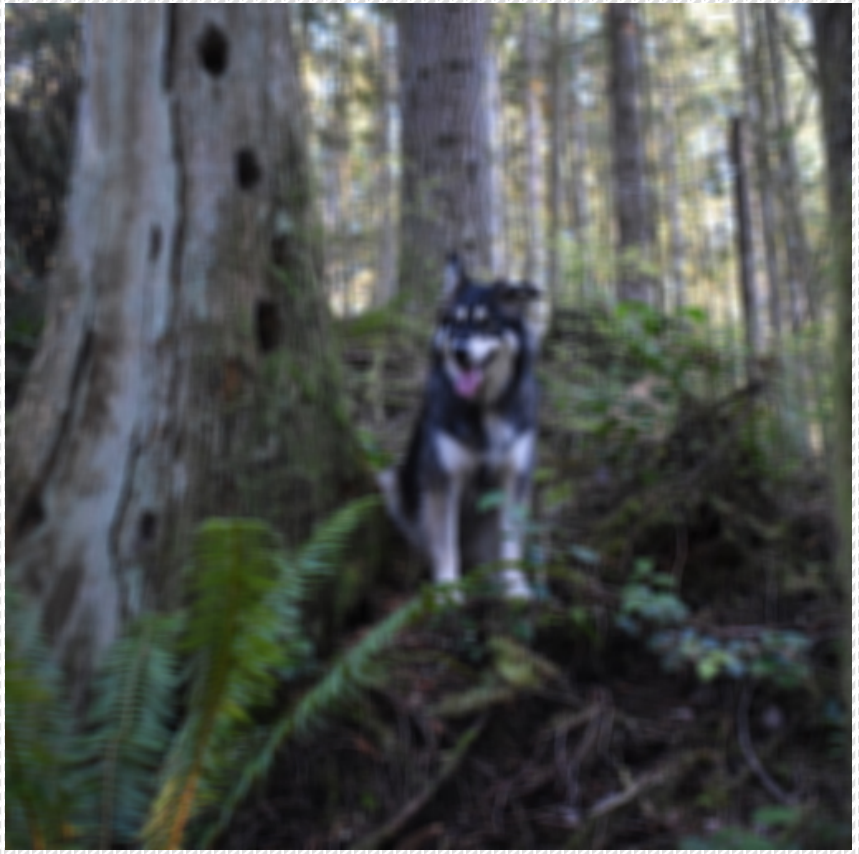
Neighborhood operators

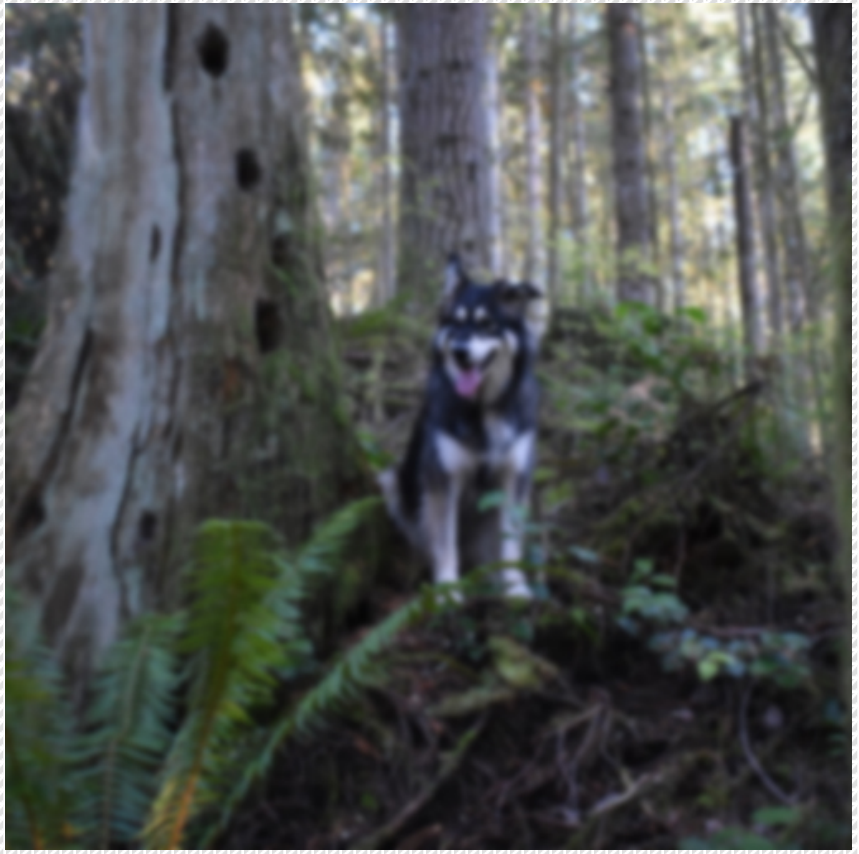
- *A Smoothly* weighted kernel



Neighborhood operators

- Better smoothing with Gaussians





Neighborhood operators

- Better smoothing with Gaussians

Box Filtered

Gaussian Filtered

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