Computer Vision

CS308

Feng Zheng

SUSTech CS Vision Intelligence and Perception

Week 3

南方科技大学

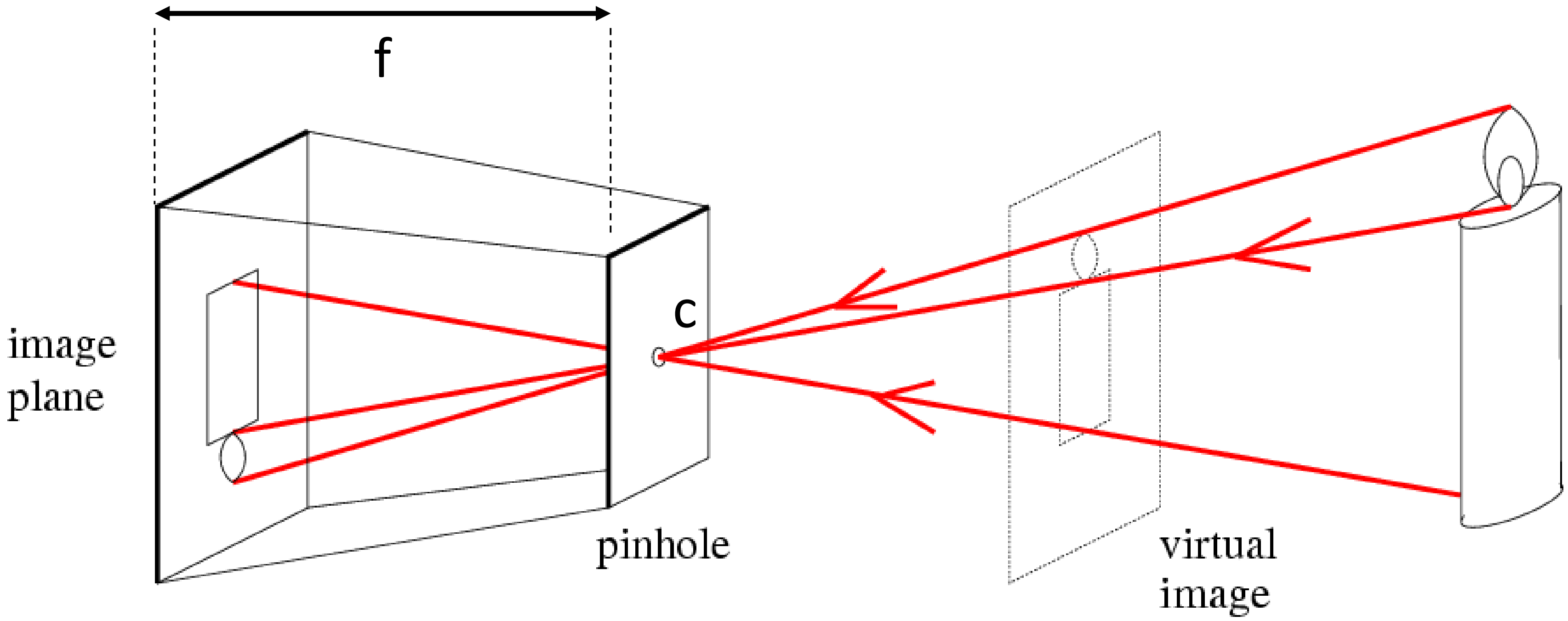
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Content

- Brief Review
- Point Operators
- Linear Filtering
- More Neighborhood Operators

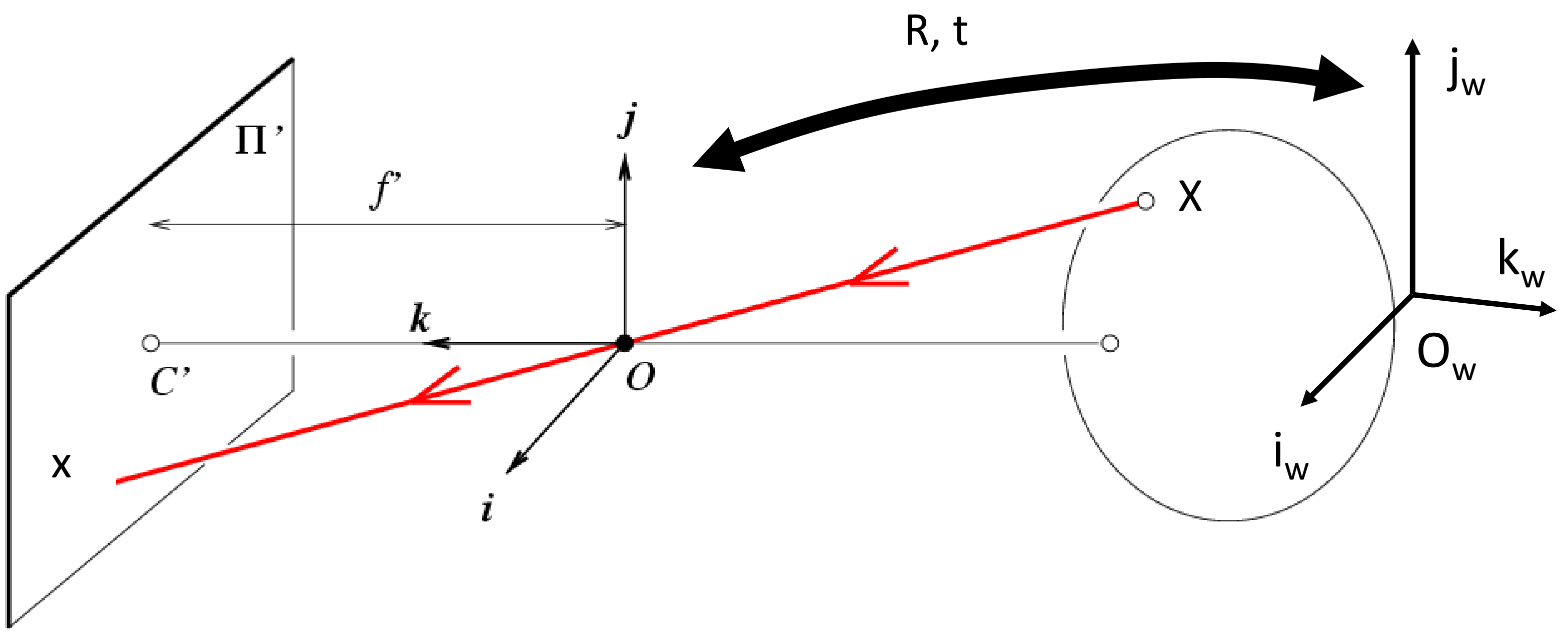
Brief Review

Pinhole Camera



f = focal length
c = center of the camera

Projection Matrix



$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

x: Image Coordinates: (u,v,1)
K: Intrinsic Matrix (3x3)
R: Rotation (3x3)
t: Translation (3x1)
X: World Coordinates: (X,Y,Z,1)

An Example of Application

- Structure from Motion (SfM) for Unordered Image Collections

What is image processing?

- (a) original image
- (b) increased contrast
- (c) change in hue
- (d) quantized colors

- (e) blurred
- (f) rotated

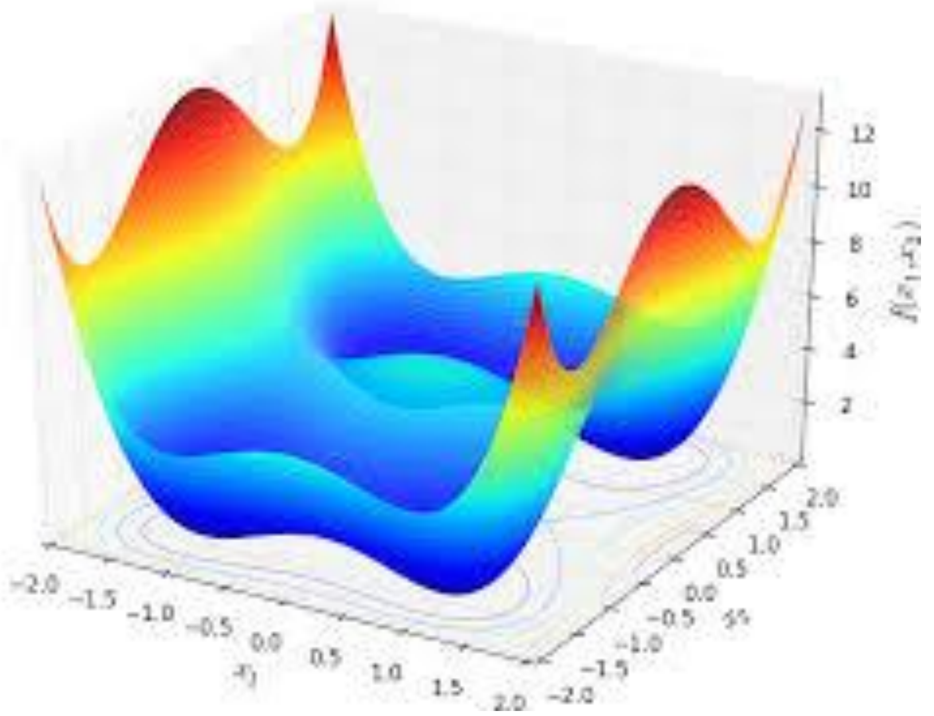
0	0	0	1	1	0	0	0	0	2	2	0	0	0	0	1	1	1
0	0	0	0	0	0	0	0	2	2	0	0	0	0	0	1	0	0
0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	34	19	0	0	4	97	34	0	0	0	0
0	0	0	0	1	1	0	89	152	56	42	145	194	29	0	0	0	0
0	0	0	2	2	0	0	59	208	237	231	251	208	19	0	1	2	3
0	0	2	2	1	0	0	47	229	218	230	220	199	47	0	2	2	1
0	1	1	0	1	1	0	71	236	195	201	218	204	49	0	1	0	0
1	1	0	0	0	0	0	114	231	214	216	198	135	0	0	0	0	0
1	1	0	0	0	0	46	108	66	58	99	107	21	0	0	0	0	0
0	0	0	0	0	7	164	120	0	18	36	36	6	0	0	0	1	1
0	0	1	0	4	126	231	183	62	14	20	44	20	0	0	1	1	1
0	1	1	0	84	223	224	240	150	14	11	80	36	0	0	0	0	0
1	1	0	2	164	224	217	235	186	16	54	187	37	0	0	0	0	0
1	1	0	28	211	221	218	234	201	46	165	204	12	0	0	0	0	0
0	0	0	57	219	218	224	238	192	59	211	152	0	0	1	1	0	1
0	0	0	121	217	226	230	229	126	45	195	105	0	0	0	2	3	3
0	0	46	209	217	225	238	189	50	94	230	161	49	0	1	3	1	0
0	0	91	235	218	222	243	164	71	201	241	250	195	7	0	0	0	0
1	0	72	233	231	223	244	157	134	243	243	255	185	3	0	0	0	0
0	0	36	206	244	232	244	183	198	249	244	255	178	1	0	0	0	0
0	0	14	93	109	185	251	137	137	221	181	168	120	5	0	1	1	0
0	0	7	22	1	65	130	74	61	77	42	9	0	0	0	1	1	1
0	0	0	0	3	26	23	7	0	2	4	0	0	0	1	1	0	0
1	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0
1	1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0

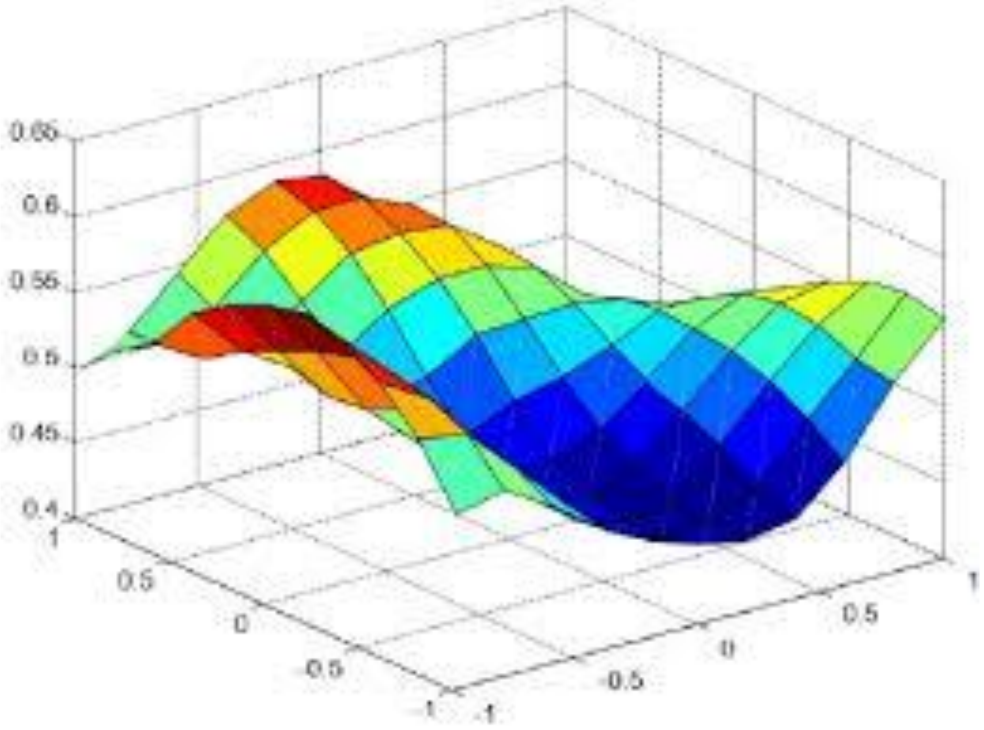
(a) (b) (c) (d) (e) (f)

Point (Pixel) Operators

Image Functions

- Image as a function in two-dimensional space





Point Operators

- A general image processing operator
  - Continuous domain $g(\mathbf{x}) = h(f(\mathbf{x}))$ or $g(\mathbf{x}) = h(f_0(\mathbf{x}), \ldots, f_n(\mathbf{x}))$
  - Discrete images $g(i, j) = h(f(i, j))$
  - Multiplication and addition $g(\mathbf{x}) = a f(\mathbf{x}) + b$

    gain controls

    brightness

    spatially varying
  - Dyadic (two-input) operator $g(\mathbf{x}) = (1 - \alpha) f_0(\mathbf{x}) + \alpha f_1(\mathbf{x})$

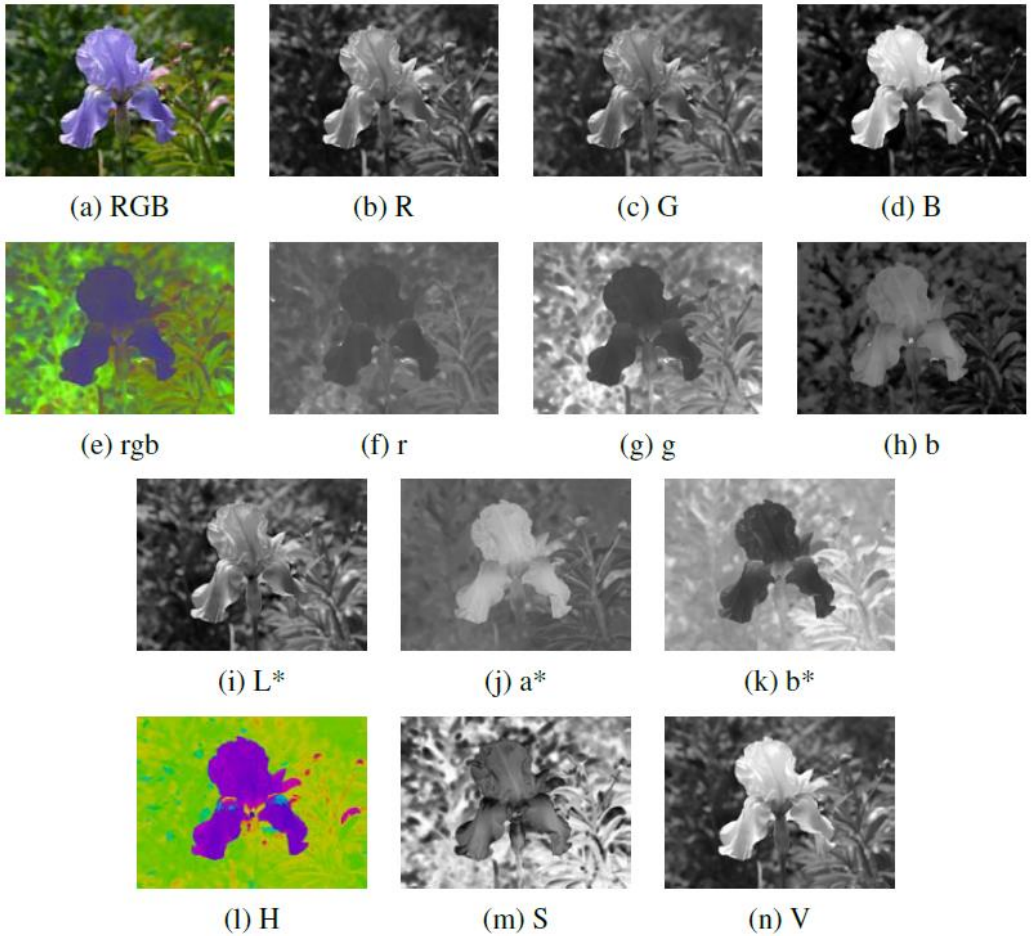
    linear blend operator
  - Gamma correction
    - ✓ Remove the non-linear mapping between input radiance and quantized pixel values

$$g(\mathbf{x}) = [f(\mathbf{x})]^{1/\gamma} \qquad \gamma \approx 2.2$$

pixel locations

Color Transforms

- (a-d) RGB
- (e-h) rgb
- (i-k) L*a*b*
- (l-n) HSV
- Brightening a picture by adding a constant value to all three channels
  - ➢ Whether this achieves the desired effect of making the image look brighter?
  - ➢ Is there any undesirable side-effects or artifacts?



Some color ratio images multiplied by the middle gray value for better visualization

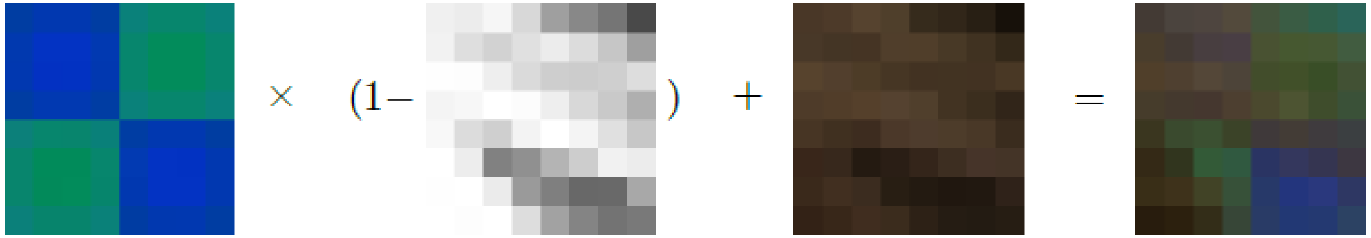
Compositing and Matting

- **Matting:** extracting the object from the original image
- **Compositing:** inserting one image into another image

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

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

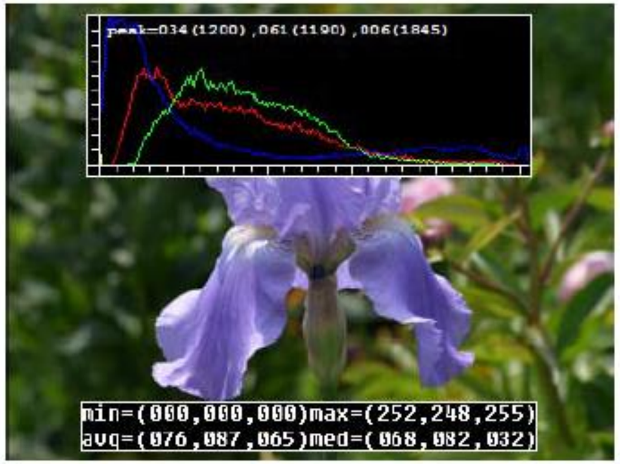
(a) source image
(b) extracted foreground object
(c) alpha matte shown in grayscale
(d) new composite

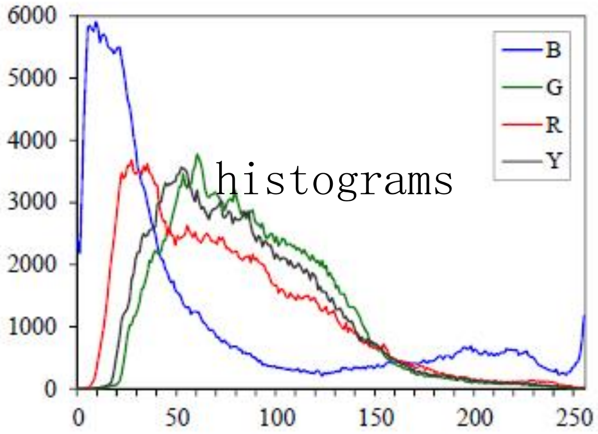


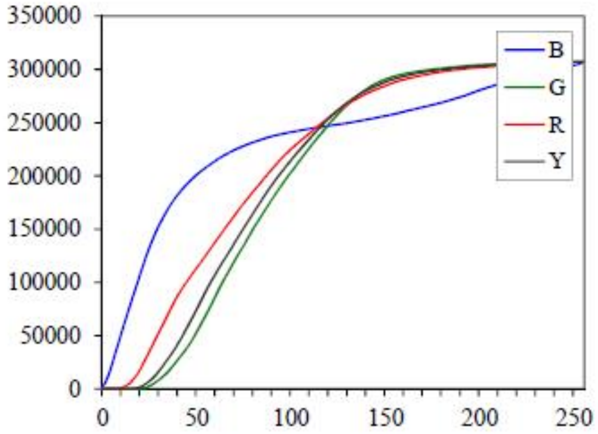
B (a) α (b) αF (c) C (d)

Histogram Equalization

- How can we automatically determine **best values** of pixels?
  - ➢ Look at the darkest and brightest values and map them to pure black and white
  - ➢ Find the average value, push it towards middle gray, and expand the range so that it more closely fills the displayable values
  - ➢ Histogram equalization: find an intensity mapping function *f(I)* such that the resulting histogram is flat

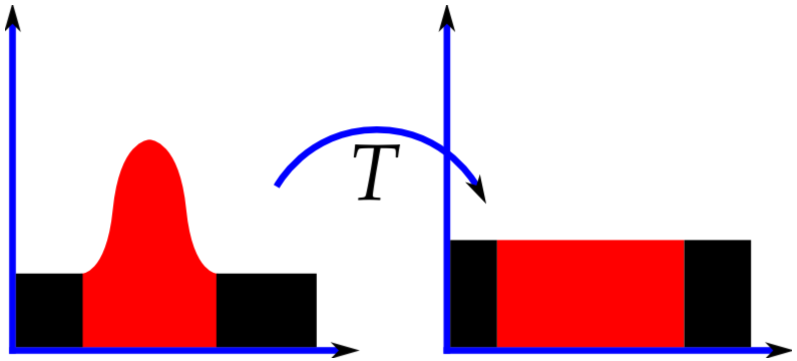


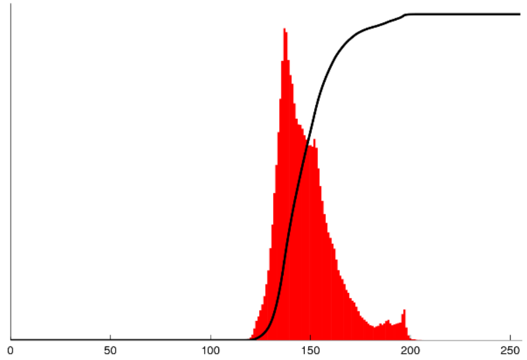


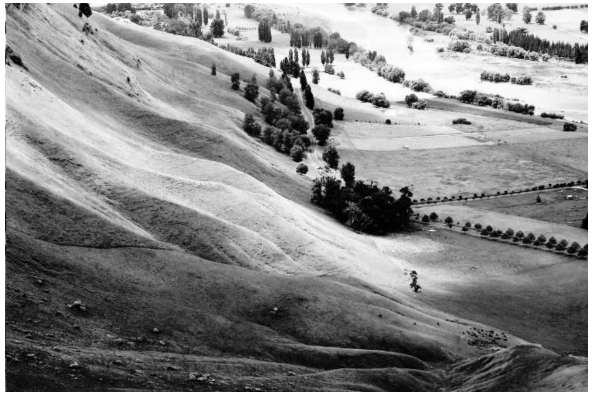


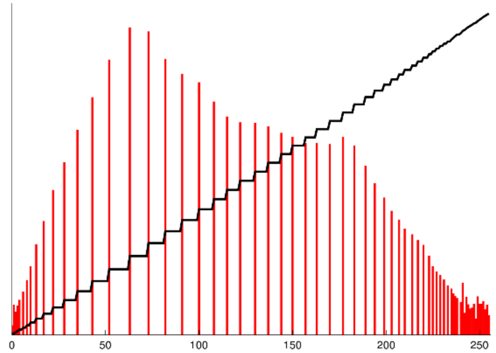
Histogram Equalization

- A technique for adjusting image **intensities** to enhance **contrast**
  - ➢ An image would have a linearized cumulative distribution function (CDF)









Histogram Equalization

- An example

$$c(I) = \frac{1}{N}\sum_{i=0}^{I} h(i) = c(I-1) + \frac{1}{N}h(I)$$

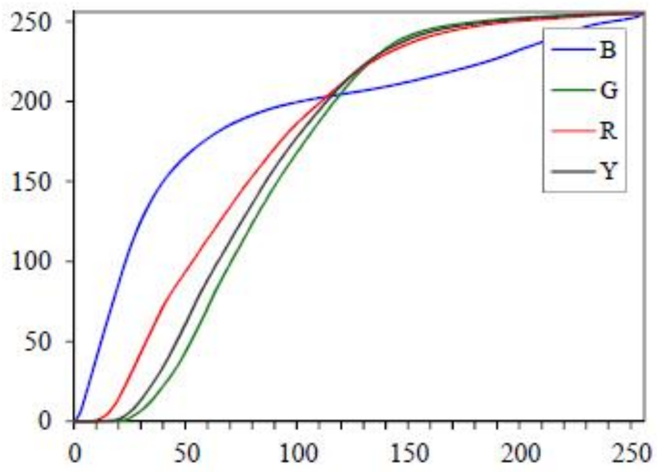
histogram

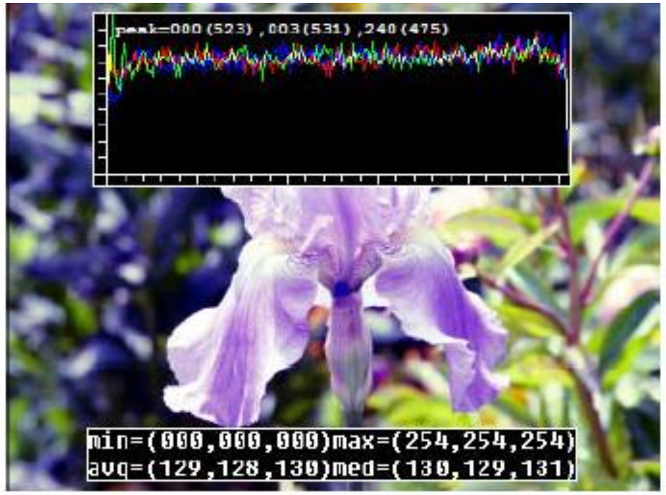
I in [0 255]

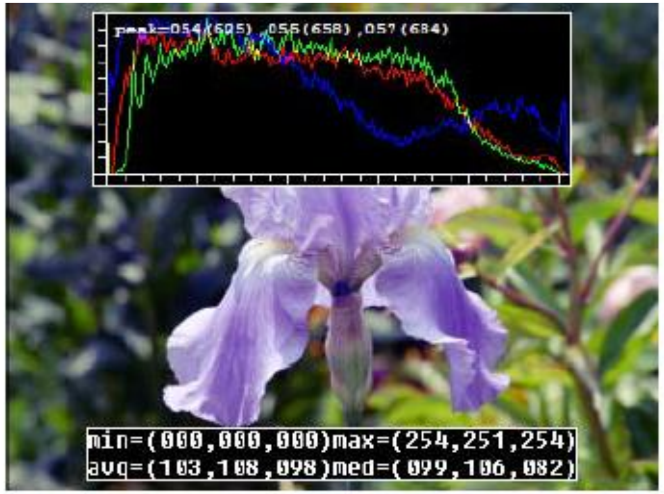
Maintain more of its original grayscale distribution while having a more appealing balance.

- **Why it is equalized?**

$$f(I) = c(I) \qquad f(I) = \alpha c(I) + (1-\alpha)I.$$

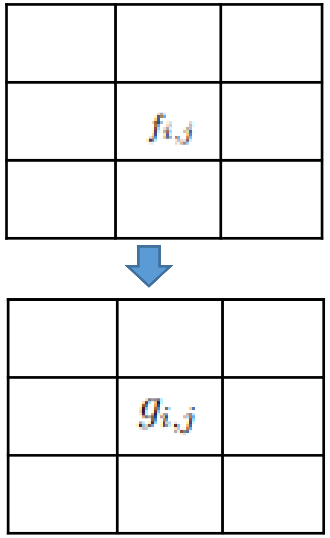






Histogram Equalization

- Calculation



Let f be a given image represented as a m_r by m_c matrix of integer pixel intensities ranging from 0 to $L-1$. L is the number of possible intensity values, often 256. Let p denote the normalized histogram of f with a bin for each possible intensity. So

$$p_n = \frac{\text{number of pixels with intensity } n}{\text{total number of pixels}} \qquad n = 0, 1, ..., L-1.$$

The histogram equalized image g will be defined by

$$g_{i,j} = \text{floor}((L-1)\sum_{n=0}^{f_{i,j}} p_n), \tag{1}$$

where floor() rounds down to the nearest integer. This is equivalent to transforming the pixel intensities, k, of f by the function

$$T(k) = \text{floor}((L-1)\sum_{n=0}^{k} p_n).$$

Histogram Equalization

- Motivation

The motivation for this transformation comes from thinking of the intensities of f and g as continuous random variables X, Y on $[0, L-1]$ with Y defined by

$$Y = T(X) = (L-1)\int_0^X p_X(x)dx, \tag{2}$$

where p_X is the probability density function of f. T is the cumulative distributive function of X multiplied by $(L-1)$. Assume for simplicity that T is differentiable and invertible. It can then be shown that Y defined by $T(X)$ is uniformly distributed on $[0, L-1]$, namely that $p_Y(y) = \frac{1}{L-1}$.

$$\begin{aligned}\int_0^y p_Y(z)dz &= \text{probability that } 0 \le Y \le y \\ &= \text{probability that } 0 \le X \le T^{-1}(y) \\ &= \int_0^{T^{-1}(y)} p_X(w)dw\end{aligned}$$

$$\frac{d}{dy}\left(\int_0^y p_Y(z)dz\right) = p_Y(y) = p_X(T^{-1}(y))\frac{d}{dy}(T^{-1}(y)).$$

Note that $\frac{d}{dy}T(T^{-1}(y)) = \frac{d}{dy}y = 1$, so

$$\frac{dT}{dx}\Big|_{x=T^{-1}(y)}\frac{d}{dy}(T^{-1}(y)) = (L-1)p_X(T^{-1}(y))\frac{d}{dy}(T^{-1}(y)) = 1,$$

which means $p_Y(y) = \frac{1}{L-1}$.

变上、下限积分求导公式

$$\frac{d}{dx}\int_a^{\varphi(x)} f(t)dt = f(\varphi(x))\cdot\varphi'(x)$$

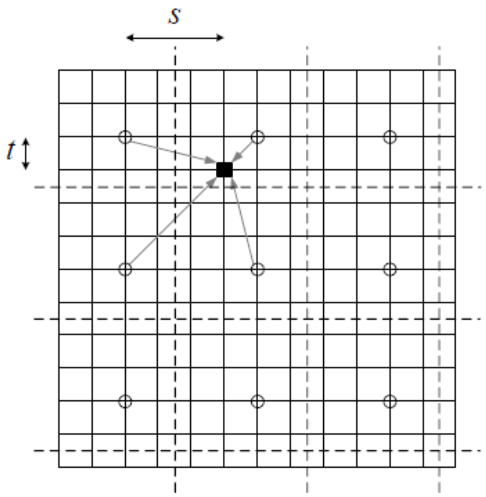
$$\frac{d}{dx}\int_{\phi(x)}^{b} f(t)dt = -f(\phi(x))\cdot\phi'(x)$$

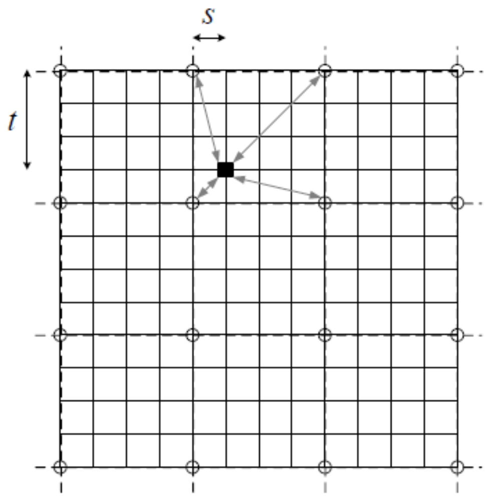
$$\frac{d}{dx}\int_{\phi(x)}^{\varphi(x)} f(t)dt = f(\varphi(x))\cdot\varphi'(x) - f(\phi(x))\cdot\phi'(x)$$

$$[f^{-1}(x)]' = \frac{1}{f'[f^{-1}(x)]}$$

Locally Adaptive Histogram Equalization

- Subdivide the image into blocks and perform separate histogram equalization in each sub-block
- Re-compute the histogram for every block centered at each pixel
- Adaptive histogram equalization:
  - Compute non-overlapped block-based equalization functions
  - Smoothly interpolate the transfer functions





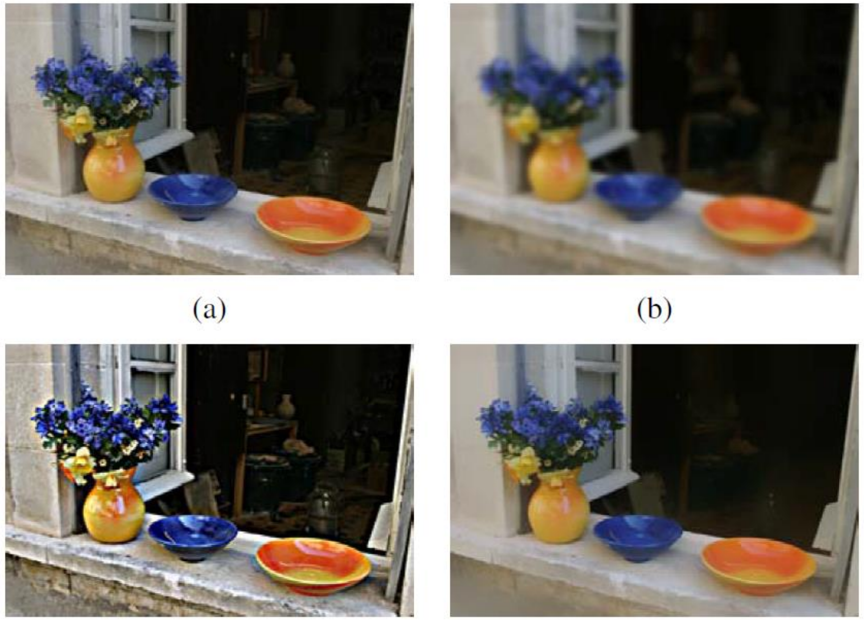
$$f_{s,t}(I) = (1-s)(1-t)f_{00}(I) + s(1-t)f_{10}(I) + (1-s)tf_{01}(I) + stf_{11}(I)$$

Distribute each input pixel into four adjacent lookup tables during the histogram accumulation phase

Linear Filtering

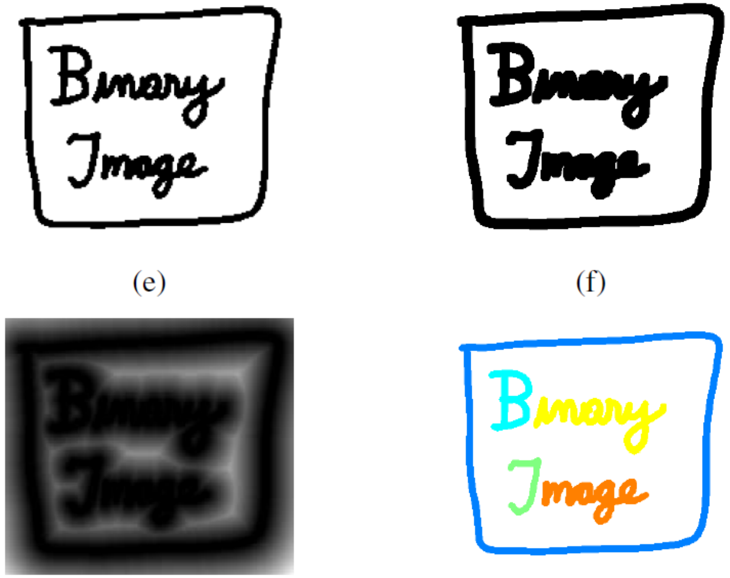
Linear Filtering

- (a) original image
- --(b) blurred
- --(c) sharpened
- --(d) smoothed with edge-preserving filter



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

- (e) binary image
- --(f) dilated
- --(g) distance transform
- --(h) connected components



(e) (f) (g) (h)

Linear Filtering

- Linear filter: correlation operator
  - ➢ The entries in the weight kernel or mask are the filter coefficients

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

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

$f(x,y)$

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$h(x,y)$

=

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

$g(x,y)$

Linear Filtering

- Linear filter: convolution operator

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

$$g=f*h$$

- Additional nice properties, e.g., it is both commutative and associative
- The Fourier transform of two convolved images is the product of their individual Fourier transforms
- The continuous version of convolution

$$g(\boldsymbol{x})=\int f(\boldsymbol{x}-\boldsymbol{u})h(\boldsymbol{u})d\boldsymbol{u}.$$

Linear Filtering

- Linear shift-invariant (LSI) operators
  - Correlation operator
  - Convolution operator
- The superposition principle $h \circ (f_0 + f_1) = h \circ f_0 + h \circ f_1$
- Shift invariance principle

$$g(i,j) = f(i+k, j+l) \quad \Leftrightarrow \quad (h \circ g)(i,j) = (h \circ f)(i+k, j+l)$$

- Matrix-vector multiply (remember cross product?)

$$\boldsymbol{g} = \boldsymbol{H} \boldsymbol{f}$$

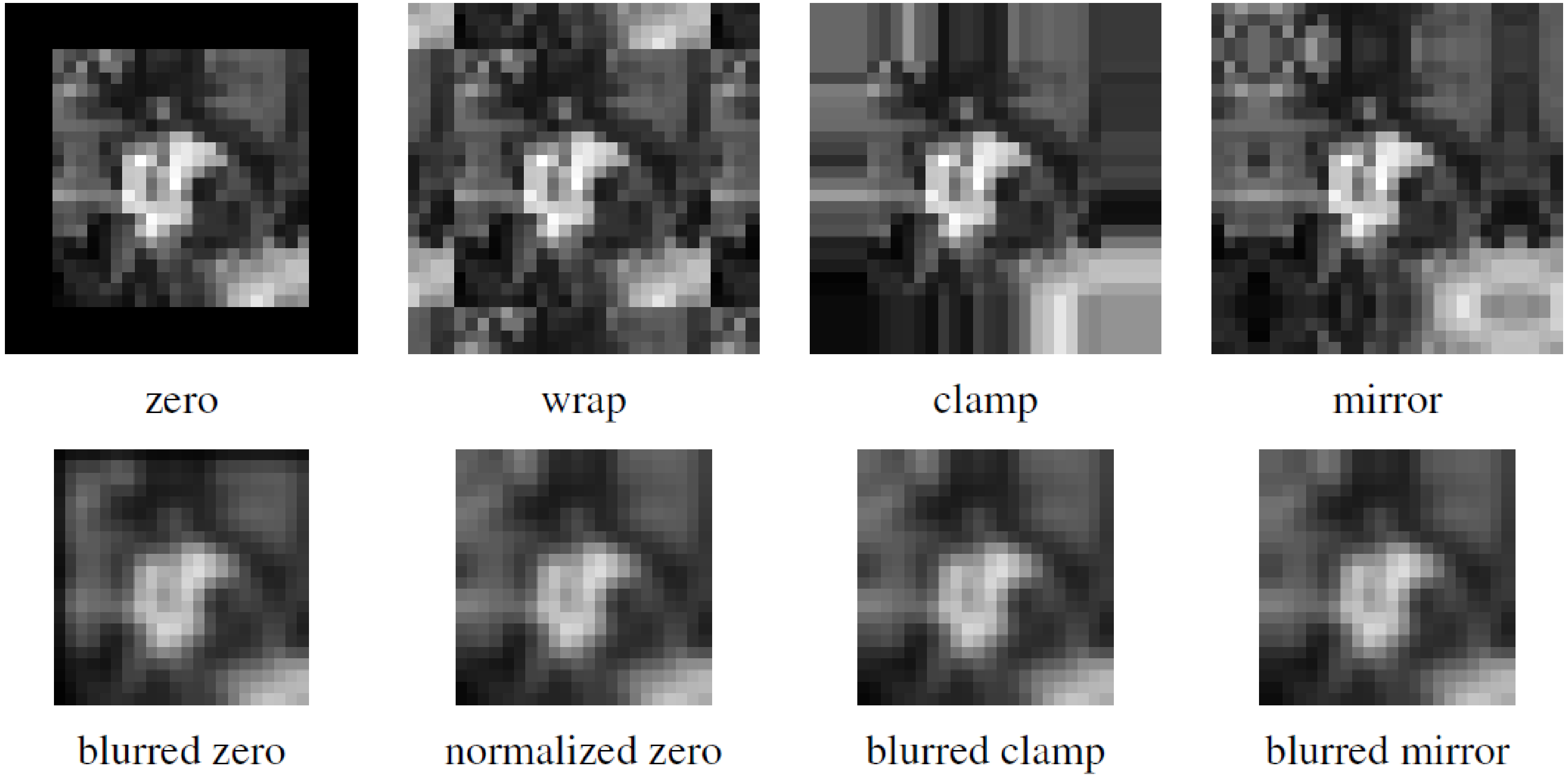
$$\boxed{72}\boxed{88}\boxed{62}\boxed{52}\boxed{37} * \boxed{1/4}\boxed{1/2}\boxed{1/4} \Leftrightarrow \frac{1}{4}\begin{bmatrix} 2 & 1 & . & . & . \\ 1 & 2 & 1 & . & . \\ . & 1 & 2 & 1 & . \\ . & . & 1 & 2 & 1 \\ . & . & . & 1 & 2 \end{bmatrix} \begin{bmatrix} 72 \\ 88 \\ 62 \\ 52 \\ 37 \end{bmatrix}$$

Padding (border effects)

- How to keep the same size?
  - zero: set all pixels outside the source image to 0
  - constant (border color): set all pixels to a specified value;
  - clamp: 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.

Padding (border effects)

- The effects of padding an image
- Padding is an important step to design a NN
  - ➢ Padding
  - ➢ Size of filter
  - ➢ Stride



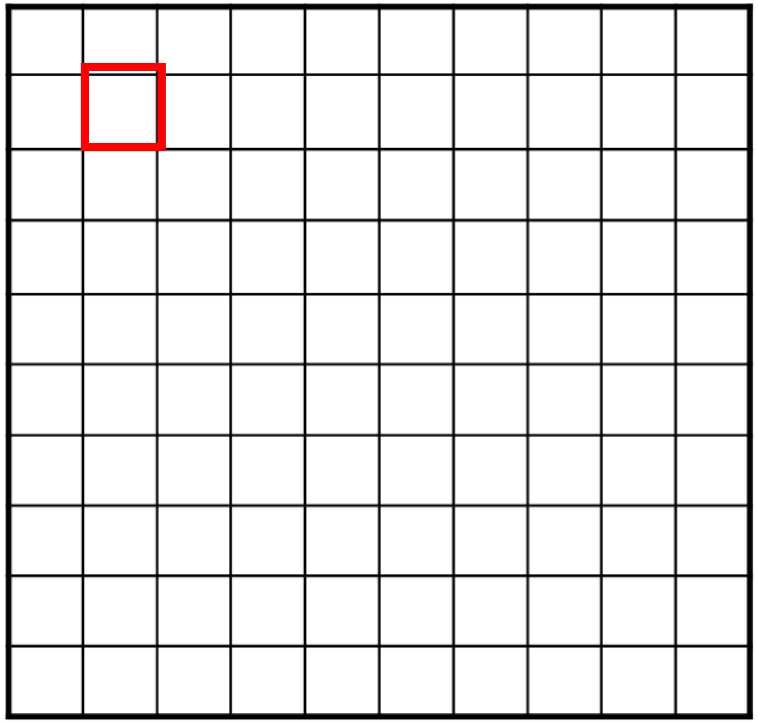
The normalized zero image is the result of dividing (normalizing) the blurred zero padded RGBA image by its corresponding soft alpha value.

Linear Filtering: An Example

$f[.,.]$

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

$h[.,.]$



$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Linear Filtering: An Example

$f[.,.]$

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

$h[.,.]$

	0	10							

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Linear Filtering: An Example

$f[.,.]$

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

$h[.,.]$

	0	10	20						

$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Linear Filtering: An Example

$f[.,.]$

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

$h[.,.]$

	0	10	20	30					

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Linear Filtering: An Example

$f[.,.]$

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

$h[.,.]$

	0	10	20	30	30				

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

$$h[m,n]=\sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Linear Filtering: An Example

$f[.,.]$

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

$h[.,.]$

	0	10	20	30	30				
				?					

$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Linear Filtering: An Example

$f[.,.]$

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

$h[.,.]$

	0	10	20	30	30				
						?			
				50					

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$$h[m,n]=\sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Linear Filtering: An Example

$f[.,.]$

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

$h[.,.]$

	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	

$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Separable Filtering

- Problem
  - Not efficient: performing a convolution requires K^2 (multiply-add) operations per pixel
- Solution: separable filtering
  - First performing a one-dimensional horizontal convolution
  - Then being followed by a one-dimensional vertical convolution
- Why it works: outer product of the two kernels
  - $2K$ operations per pixel

$$K = vh^T \qquad K = \sum_i \sigma_i u_i v_i^T$$

Separable Filtering

- Examples

$$\frac{1}{K^2}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array}$$

$$\frac{1}{K}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline \end{array}$$

(a) box, $K = 5$

$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$\frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array}$$

(b) bilinear

$$\frac{1}{256}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 6 & 24 & 36 & 24 & 6 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array}$$

$$\frac{1}{16}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array}$$

(c) "Gaussian"

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

$$\frac{1}{2}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array}$$

(d) Sobel

First derivative image

$$\frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline -2 & 4 & -2 \\ \hline 1 & -2 & 1 \\ \hline \end{array}$$

$$\frac{1}{2}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline \end{array}$$

(e) corner

Band-Pass

- Gaussian filter

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

- Second derivative of a two-dimensional image

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- Discrete Laplacian filters
  - ➢ Used to find areas of rapid change (edges) in images
  - ➢ Finite-difference method or by the finite-element method

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

(a) (b) (c)

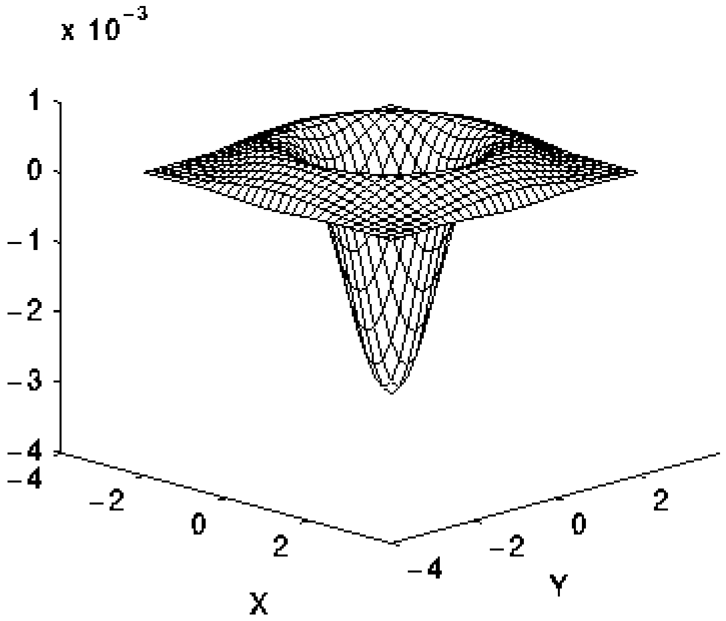
(a) original image of Einstein;
(b) orientation map computed from the second-order oriented energy;
(c) Original image with oriented structures enhanced.

Band-Pass

- Problem
  - ➢ Laplacian filter is sensitive to noise
  - ➢ Common to smooth the image (e.g., using a Gaussian filter) before applying the Laplacian
- This two-step process
  - ➢ Laplacian of Gaussian (LoG) filter

$$\nabla^2 G(x, y; \sigma) = \left(\frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2} \right) G(x, y; \sigma)$$

$$\begin{pmatrix} 0 & 0 & 3 & 2 & 2 & 2 & 3 & 0 & 0 \\ 0 & 2 & 3 & 5 & 5 & 5 & 3 & 2 & 0 \\ 3 & 3 & 5 & 3 & 0 & 3 & 5 & 3 & 3 \\ 2 & 5 & 3 & -12 & -23 & -12 & 3 & 5 & 2 \\ 2 & 5 & 0 & -23 & -40 & -23 & 0 & 5 & 2 \\ 2 & 5 & 3 & -12 & -23 & -12 & 3 & 5 & 2 \\ 3 & 3 & 5 & 3 & 0 & 3 & 5 & 3 & 3 \\ 0 & 2 & 3 & 5 & 5 & 5 & 3 & 2 & 0 \\ 0 & 0 & 3 & 2 & 2 & 2 & 3 & 0 & 0 \end{pmatrix}$$



Steerable Filters

- Sobel operator

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \qquad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A}$$

$$\mathbf{G}_x = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * ([-1 \quad 0 \quad +1] * \mathbf{A}) \quad \mathbf{G}_y = \begin{bmatrix} -1 \\ 0 \\ +1 \end{bmatrix} * ([1 \quad 2 \quad 1] * \mathbf{A})$$

- Directional or oriented filter
  - ➢ Smoothing with a Gaussian
  - ➢ Taking a directional derivative

$$\hat{\boldsymbol{u}} \cdot \nabla(G * f) = \nabla_{\hat{\boldsymbol{u}}}(G * f) = (\nabla_{\hat{\boldsymbol{u}}} G) * f \qquad \hat{\boldsymbol{u}} = (\cos\theta, \sin\theta)$$

  - ➢ The smoothed directional derivative filter

$$G_{\hat{\boldsymbol{u}}} = uG_x + vG_y = u\frac{\partial G}{\partial x} + v\frac{\partial G}{\partial y}$$

a steerable filter

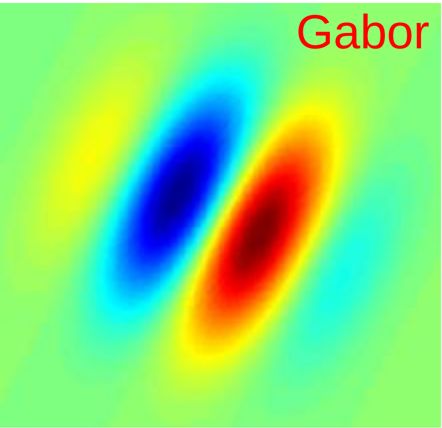
Steerable Filters

- Steps
  - First convolving with the pair of filters
  - Steering the filter by multiplying this gradient with a unit vector
- Advantage
  - A whole family of filters can be evaluated with very little cost
- A directional second derivative filter

$$G_{\hat{\mathbf{u}}\hat{\mathbf{u}}} = u^2 G_{xx} + 2uv G_{xy} + v^2 G_{yy}$$

- Steerable filters are often used
  - Feature descriptors and edge detectors

Example of a two-dimensional Gabor filter



Summed Area Table: Integral Image

- The running sum of all the pixel values from the origin

$$s(i,j) = \sum_{k=0}^{i}\sum_{l=0}^{j} f(k,l) \qquad s(i,j) = s(i-1,j) + s(i,j-1) - s(i-1,j-1) + f(i,j)$$

$$S(i_0 \ldots i_1, j_0 \ldots j_1) = s(i_1,j_1) - s(i_1, j_0 - 1) - s(i_0 - 1, j_1) + s(i_0 - 1, j_0 - 1)$$

3	2	7	2	3
1	5	1	3	4
5	1	**3**	5	1
4	3	2	1	6
2	4	1	4	8

(a) S = 24

3	5	12	14	17
4	*11*	**19**	24	31
9	**17**	28	38	46
13	24	37	48	62
15	30	44	59	81

(b) s = 28

3	5	12	*14*	17
4	11	19	24	31
9	17	28	38	46
13	24	37	**48**	62
15	30	44	59	81

(c) S = 24

More Neighborhood Operators (Non-linear)

Median Filtering

- Select the median from neighborhood
  - Filter away shot noise
  - It is not as efficient (rank) as Guassian
- Alpha-trimmed mean
  - Average all of the pixels except for the alpha-fraction that are the smallest and the largest
- Weighted median
  - Each pixel is used a number of times depending on its distance from the center
  - Equivalent to minimizing the weighted objective function

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

(a) median = 4

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

(b) α-mean= 4.6

$$\sum_{k,l} w(k,l)|f(i+k,j+l)-g(i,j)|^p$$

Bilateral Filtering

- A bilateral filter is a non-linear, edge-preserving, and noise-reducing smoothing filter for images
- It replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels.

$$g(i,j) = \frac{\sum_{k,l} f(k,l) w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

$$w(i,j,k,l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2} - \frac{\|f(i,j) - f(k,l)\|^2}{2\sigma_r^2}\right)$$

$$d(i,j,k,l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2}\right)$$

$$r(i,j,k,l) = \exp\left(-\frac{\|f(i,j) - f(k,l)\|^2}{2\sigma_r^2}\right)$$

product of a domain kernel data-dependent range kernel

Bilateral Filtering

- This weight can be based on a Gaussian distribution. Crucially, the weights depend not only on Euclidean distance of pixels, but also on the radiometric differences (e.g., range differences, such as color intensity, depth distance, etc.). This preserves sharp edges.

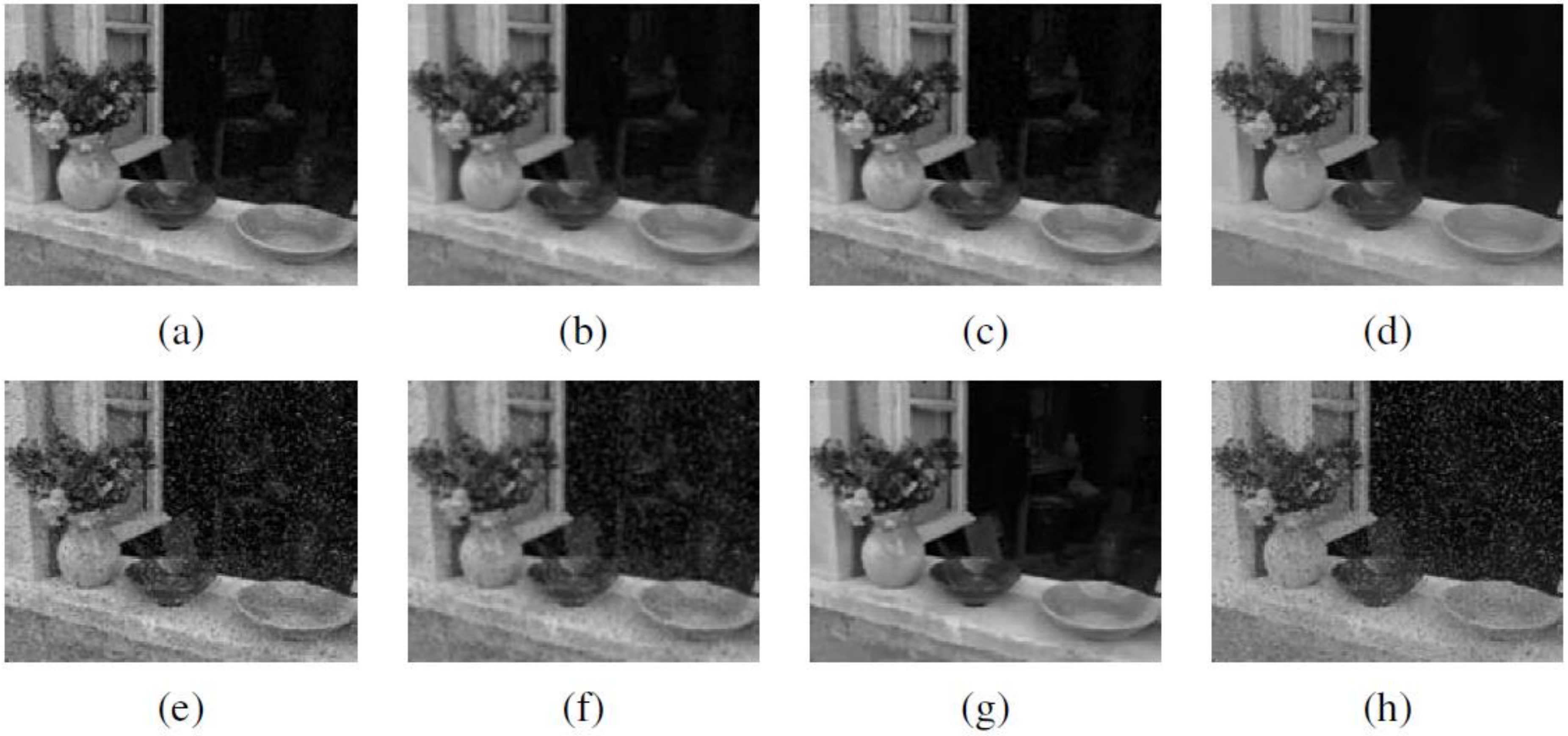
	2	1	0	1	2
2	0.1	0.3	0.4	0.3	0.1
1	0.3	0.6	0.8	0.6	0.3
0	0.4	0.8	1.0	0.8	0.4
1	0.3	0.6	0.8	0.6	0.3
2	0.1	0.3	0.4	0.3	0.1

(c) domain filter

0.0	0.0	0.0	0.0	0.2
0.0	0.0	0.0	0.4	0.8
0.0	0.0	1.0	0.8	0.4
0.0	0.2	0.8	0.8	1.0
0.2	0.4	1.0	0.8	0.4

(d) range filter

An Example



Note that the bilateral filter fails to remove the shot noise because the noisy pixels are too different from their neighbors.

(a) original image with Gaussian noise; (b) Gaussian filtered; (c) median filtered; (d) bilaterally filtered;
(e) original image with shot noise; (f) Gaussian filtered; (g) median filtered; (h) bilaterally filtered

Morphology

- Binary images
  - First convolve the binary image with a binary structuring element
    - ✓ It can be any shape, from a simple 3 × 3 box filter s

      $$c = f \otimes s$$
  - Then select a binary output value depending on the thresholded result of the convolution

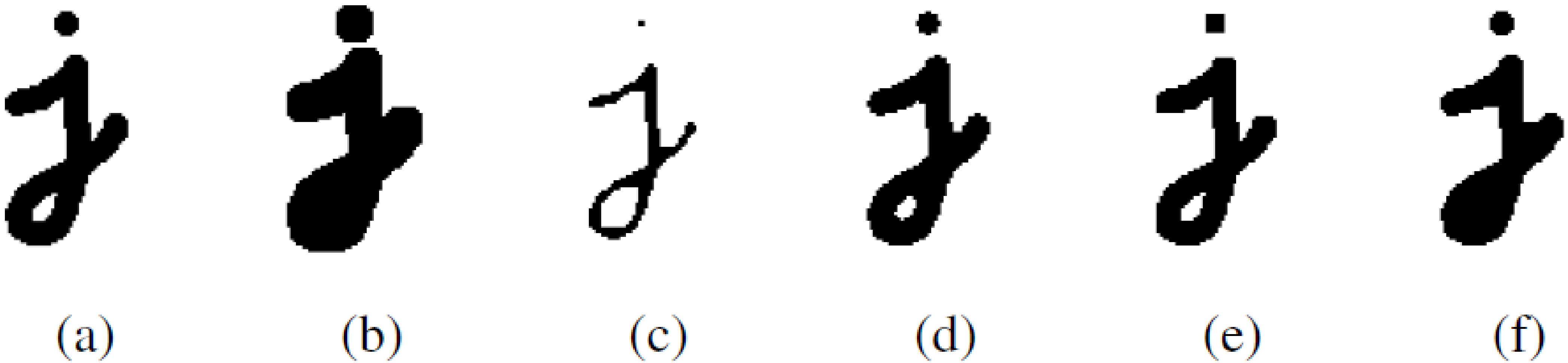
$$\theta(f,t) = \begin{cases} 1 & \text{if } f \geq t, \\ 0 & \text{else,} \end{cases}$$

- **dilation**: $\text{dilate}(f, s) = \theta(c, 1)$;
- **erosion**: $\text{erode}(f, s) = \theta(c, S)$;
- **majority**: $\text{maj}(f, s) = \theta(c, S/2)$;
- **opening**: $\text{open}(f, s) = \text{dilate}(\text{erode}(f, s), s)$;
- **closing**: $\text{close}(f, s) = \text{erode}(\text{dilate}(f, s), s)$

S number of pixels

Morphology

- The structuring element for all examples is a 5*5 square



(a) original image; (b) dilation; (c) erosion; (d)majority; (e) opening; (f) closing

Distance Transforms (Binary Image)

- The distance transform
  - ➢ Quickly pre-computing the distance to a curve or set of points
    - ✓ Manhattan distance $d_1(k,l) = |k| + |l|$
    - ✓ Euclidean distance $d_2(k,l) = \sqrt{k^2 + l^2}$
- How to calculate distances to the *nearest* background pixel

$$\boxed{D(i,j) = \min_{k,l:b(k,l)=0} d(i-k, j-l)}$$

  - ➢ Forward and backward pass of a simple raster-scan algorithm

0	0	0	0	1	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0
0	0	1	0	0	0	0
0	0	0	0	0	0	0

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

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

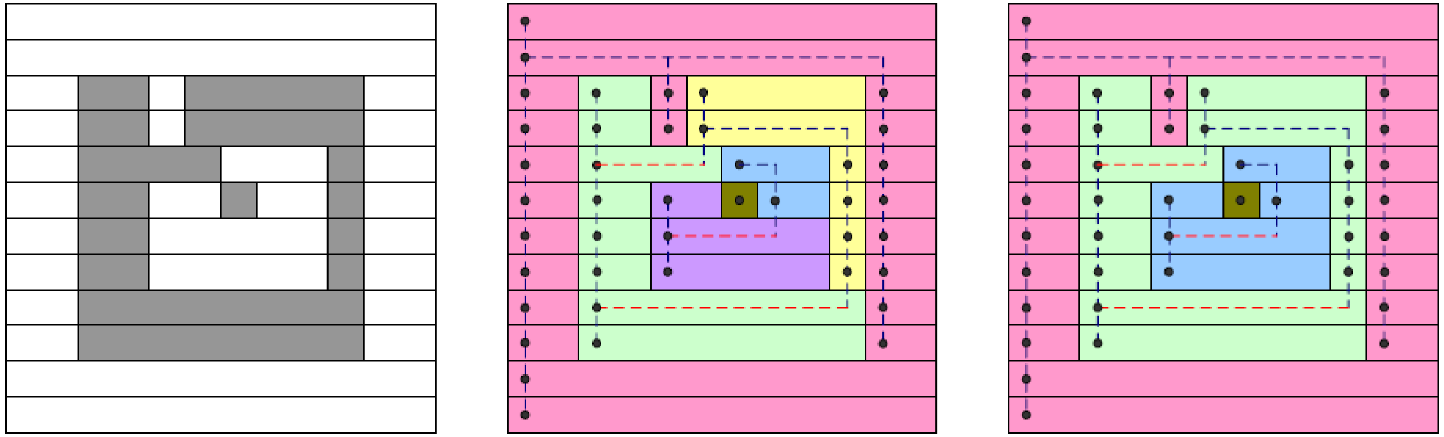
0	0	0	0	1	0	0
0	0	1	1	1	0	0
0	1	2	2	2	1	0
0	1	2	2	1	1	0
0	1	2	1	0	0	0
0	0	1	0	0	0	0
0	0	0	0	0	0	0

Connected Components

- Connected components
  - Define regions of adjacent pixels that have the same input value
  - Consider pixels to be adjacent if they are immediate N4 neighbors and they have the same input value
- Connected components can be used to
  - Finding individual letters
  - Finding objects
  - Compute their area statistics
    - ✓ The area (number of pixels)
    - ✓ The perimeter (number of boundary pixels)
    - ✓ The centroid (average x and y values)
    - ✓ The second moments

Connected Components

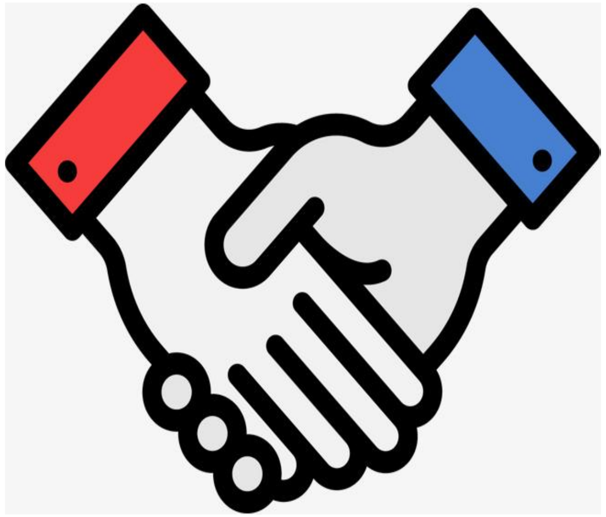
- First step
  - ➢ Split the image into horizontal runs of adjacent pixels
  - ➢ Then color the runs with unique labels
  - ➢ Re-use the labels of vertically adjacent runs whenever possible
- Second step
  - ➢ Adjacent runs of different colors are then merged



Conclusions

Conclusions

- Point operators
  - ➢ Include brightness and contrast adjustments as well as color correction and transformations
- Neighborhood operators: filter images (linear)
  - ➢ Add soft blur, sharpen details, accentuate edges, or remove noise
- More neighborhood operators (non-linear)
  - ➢ Edge preserving median, bilateral filters,
  - ➢ Morphological operators that operate on binary images,
  - ➢ Semi-global operators that compute distance transforms and find connected components in binary images



Thanks

zhengf@sustc.edu.cn