Chapter 6 Neighborhood Operators

6.1 Introduction

neighborhood operator: workhorse of low-level vision
neighborhood operator: performs conditioning, labeling, grouping

The output of a neighborhood operator at a given pixel position is a function of the position, of the input pixel value at the position, of the values of the input pixels in some neighborhood around the given input position, and possibly of some values of previously generated output pixels.
numeric domain: arithmetic operations, +, -, min, max
symbolic domain: Boolean operations, AND, OR, NOT, table-look-up
nonrecursive neighborhood operators: output is function of input
recursive neighborhood operators: output depends partly on previous output
neighborhood might be small and asymmetric or large
=====Fig. 6.1=====

$N(r,c)$: set of neighboring pixel positions around position (r,c)
general nonrecursive neighborhood operator ϕ : input f, output g

$$g(r,c) = \phi[r,c,f(r',c') : (r',c') \in N(r,c)]$$

linear operator: one common nonrecursive neighborhood operator
output: possibly position-dependent linear combination of inputs

$$g(r,c) = \sum_{(r',c') \in N(r,c)} f(r',c') w(r',c',r,c)$$

shift-invariant: position invariant: action same regardless of position
composition of shift-invariant operators: shift-invariant

cross-correlation of f with w: $g = f \otimes w$

$$g(r,c) = \sum_{(r',c') \in W, (r+r',c+c') \in F} f(r+r', c+c') w(r',c')$$

w: weight function: kernel or mask of weights
W: domain of w
common 3×3 masks for noise cleaning, (a) box filter
=====Fig. 6.2=====
common 5×5 masks for noise cleaning
=====Fig. 6.3=====
application of mask with weights to image
=====Fig. 6.5=====

convolution of f with w: $g = f * w$

$$g(r,c) = \sum_{(r',c') \in W, (r-r', c-c') \in F} f(r-r', c-c') w(r', c')$$

convolution: close relative to cross-correlation
convolution: linear shift-invariant
if mask symmetric, convolution and correlation the same

6.2 Symbolic Neighborhood Operators

indexing of neighborhoods
=====Fig. 6.7=====

6.2.1 Region-Growing Operator

h: projection, outputs first or second argument

$$h(c,d) = \begin{cases} d & \text{if } c = g \\ c & \text{if } c \neq g \end{cases}$$

g: background
background pixel labeled first nonbackground label

4-connected: $a_0 = x_0$

$$a_n = h(a_{n-1}, x_n), \quad n = 1, ..., 4$$

output $y = a_4$

8-connected: $a_0 = x_0$

$$a_n = h(a_{n-1}, x_n), \quad n = 1, ..., 8$$

output $y = a_8$
=====Fig. 6.8=====

6.2.2 Nearest Neighbor Sets and Influence Zones

influence zones: nearest neighbor sets
influence zones: iteratively region-growing
4-neighborhood for city-block distance
8-neighborhood for max distance (of horizontal and vertical distances)
alternate 4, 8-neighborhood for Euclidean distance
=====joke=====

6.2.3 Region-Shrinking Operator

region-shrinking: changes all border pixels to background
region-shrinking: can change connectivity
region-shrinking: can entirely delete region if repeatedly applied

h: whether or not arguments identical

$$h(c,d) = \begin{cases} c & \text{if } c = d \\ g & \text{if } c \neq d \end{cases}$$

g: background
border: has different neighbor and becomes background

4-connected: $a_0 = x_0$

$$a_n = h(a_{n-1}, x_n), \quad n = 1, ..., 4$$

output $y = a_4$

8-connected: $a_0 = x_0$

$$a_n = h(a_{n-1}, x_n), \quad n = 1, ..., 8$$

output $y = a_8$
region shrinking: related to binary erosion except on labeled region
=====Fig. 6.9=====

6.2.4 Mark-Interior/Border-Pixel Operator
mark-interior/border-pixel operator marks all interior pixels with the label i and all border pixels with the label b

h: whether or not arguments identical

$$h(c,d) = \begin{cases} c & \text{if } c = d \\ b & \text{if } c \neq d \end{cases}$$

f: recognizes whether or not its argument is symbol b

$$f(c) = \begin{cases} b & \text{if } c = b \\ i & \text{if } c \neq b \end{cases}$$

4-connected: $a_0 = x_0$

$$a_n = h(a_{n-1}, x_n), \quad n = 1, ..., 4$$

output $y = f(a_4)$

8-connected: $a_0 = x_0$

$$a_n = h(a_{n-1}, x_n), \quad n = 1, ..., 8$$

output $y = f(a_8)$

6.2.5 Connectivity Number Operator

connectivity number: nonrecursive and symbolic data domain
connectivity number: classify the way pixel connected to neighbors
six values of connectivity: five for border, one for interior

border: isolated, edge, connected, branching, crossing
=====Fig. 6.10=====
corner neighborhood
=====Fig. 6.11=====

Yokoi Connectivity Number
4-connectivity

$$h(b,c,d,e) = \begin{cases} q & \text{if } b = c \text{ and } (d \neq b \text{ or } e \neq b) \\ r & \text{if } b = c \text{ and } (d = b \text{ and } e = b) \\ s & \text{if } b \neq c \end{cases}$$

q: corner $1 \longrightarrow 0$ transition
r: corner all 1, no transition
s: center 1, neighbor 0, nothing will happen

$$f(a_1, a_2, a_3, a_4) = \begin{cases} 5 & \text{if } a_1 = a_2 = a_3 = a_4 = r \\ n & \text{where } n = \#\{a_k | a_k = q\}, \text{ otherwise} \end{cases}$$

5: no transition, all 8 neighbors 1, thus interior
n: 1 transition generates one connected component if center removed
connectivity number $y = f(a_1, a_2, a_3, a_4)$

$$a_1 = h(x_0, x_1, x_6, x_2)$$

$$a_2 = h(x_0, x_2, x_7, x_3)$$
$$a_3 = h(x_0, x_3, x_8, x_4)$$
$$a_4 = h(x_0, x_4, x_5, x_1)$$

=====Fig. 6.12=====

=====lena.64x64=====

=====lena.yokoi=====

8-connectivity, only h slightly different

$$h(b,c,d,e) = \begin{cases} q & \text{if } b \neq c \text{ and } (d = b \text{ or } e = b) \\ r & \text{if } b = c \text{ and } (d = b \text{ and } e = b) \\ t & \text{if } b = c \text{ and } (d \neq b \text{ and } e = b) \\ s & \text{otherwise} \end{cases}$$

q: center 1 and corner $0 \rightarrow 1$ transition

r: corner no transition, all 1

t: itself and two 4-neighbors 1, corner 0

$$f(a_1, a_2, a_3, a_4) = \begin{cases} 5 & \text{if } a_1 = a_2 = a_3 = a_4 = r \\ n & \text{where } n = \#\{a_k | a_k = q\} \text{ and } n \neq 0 \\ 1 & n = 0 \text{ and } m \neq 0 \text{ where } m = \#\{a_k | a_k = t\} \\ 0 & n = 0 \text{ and } m = 0 \end{cases}$$

Rutovitz Connectivity Number

Rutovitz connectivity: number of transitions from one symbol to another
Rutovitz connectivity number: sometimes called crossing number

6.2.6 Connected Shrink Operator

connected shrink: recursive operator, symbolic data domain
connected shrink: deletes border pixels without disconnecting regions
top-down, left-right scan: deletes edge pixels not right-boundary
=====Fig. 6.13=====

h: determines whether corner connected

$$h(b,c,d,e) = \begin{cases} 1 & \text{if } b = c \text{ and } (d \neq b \text{ or } e \neq b) \\ 0 & \text{otherwise} \end{cases}$$

$$f(a_1,a_2,a_3,a_4,x) = \begin{cases} g & \text{if exactly one of } a_1,a_2,a_3,a_4 = 1 \\ x & \text{otherwise} \end{cases}$$

g: background
output symbol $y = f(a_1,a_2,a_3,a_4,x_0)$

$$a_1 = h(x_0,x_1,x_6,x_2)$$

$$a_2 = h(x_0,x_2,x_7,x_3)$$

$$a_3 = h(x_0, x_3, x_8, x_4)$$
$$a_4 = h(x_0, x_4, x_5, x_1)$$

=====Fig. 6.14=====

6.2.7 Pair Relationship Operator

pair relationship: nonrecursive operator, symbolic data domain

h: determines whether first argument equals label 1

$$h(a, 1) = \begin{cases} 1 & \text{if } a = 1 \\ 0 & \text{otherwise} \end{cases}$$

4-connected mode, output y

$$y = \begin{cases} q & \text{if } \Sigma_{n=1}^{4} h(x_n, 1) < 1 \text{ or } x_0 \neq 1 \\ p & \text{if } \Sigma_{n=1}^{4} h(x_n, 1) \geq 1 \text{ and } x_0 = 1 \end{cases}$$

q: not deletable if Yokoi number $\neq 1$ or no neighbor 1

p: possibly deletable if Yokoi number 1 and some neighbor 1

6.2.8 Thinning Operator

thinning operator is composition of three operators: Yokoi connectivity, pair relationship, connected shrink

=====Fig. 6.15=====

=====lena.thin=====
=====joke=====

6.2.9 Distance Transformation Operator
distance transformation: produces distance to closest border pixel
nonrecursive, iterative, start with

g	g	g	g	g	g	g
g	g	0	0	0	0	g
g	g	0	i	i	0	g

...
g: background
0: border, because distance to border is 0
i: interior pixels
=====Fig. 6.16=====
nth iteration: label with n all i with neighbor $n-1$

equivalent algorithm: nonrecursive, iterative

$$h(a_0,...,a_N) = \begin{cases} i & \text{if } a_n = i, n = 0,...,N \\ \min & \{b | \text{for some } n \le N, a_n \ne i, b = a_n + 1\} \\ & \text{if } a_0 = i \text{ and there exists } n \text{ such that } a_n \ne i \\ a_0 & \text{if } a_0 \ne i \end{cases}$$

i: if no neighbor labeled, still interior, leave it alone
min: $a_0 = x_0$ self interior but neighbor labeled
a_0: already labeled, won't change value since later route farther
4-connected: output $y = h(x_0, x_1, x_2, x_3, x_4)$

distance transformation: produces distance to closest background
recursive, two-pass
first pass: left-right, top-bottom

$$h(a_1,...,a_N; d) = \begin{cases} 0 & \text{if } a_N = 0 \\ \min\{a_1,...,a_{N-1}\} + d & \text{otherwise} \end{cases}$$

0: if background still background, since $a_N = x_0$
min: min of upper and left neighbors and add $d = 1$
4-connected: output $y = h(x_2, x_3, x_0; 1)$
second pass: right-left, bottom-up

$$g(a_1,...,a_N; d) = \min\{a_1,...,a_N\} + d$$

4-connected: output $y = \min\{x_0, g(x_1, x_4; 1)\}$
after pass 1

0	0	0	0	0	0	0
0	0	1	1	1	1	0
0	0	1	2	2	2	0
0	1	2	3	3	3	0
0	0	1	2	3	4	0
0	0	1	2	3	4	0
0	0	0	0	0	0	0

=====Fig. 6.16=====

6.2.10 Radius of Fusion

The radius of fusion for any connected component of binary image I is the radius ρ of a disk satisfying the condition that if the binary image I is morphologically closed with a disk of radius ρ, then the given connected region will fuse with some other connected region

6.2.11 Number of Shortest Paths

number of shortest paths for each 0-pixel: to binary-1 pixel set given binary image $p^0(r,c)$

$$p^m(r,c) = \begin{cases} p^{m-1}(r,c) & \text{if } p^{m-1}(r,c) \neq 0 \\ \Sigma_{(r',c')\in N}\, p^{m-1}(r-r', c-c') & \text{if } p^{m-1}(r,c) = 0 \end{cases}$$

N: can be 4-neighborhood or 8-neighborhood
$p^{m-1}(r,c)$: nonzero, stays unchanged since shortest paths counted
Σ: if zero then sum of neighboring shortest paths
=====Fig. 6.17=====

6.3 Extremum-Related Neighborhood Operators

6.3.1 Non-Minima-Maxima Operator

non-minima-maxima: nonrecursive operator, symbolic output
a pixel can be neighborhood maximum not relative maximum
=====Fig. 6.18=====
4-connected: $a_0 = b_0 = x_0$

$$a_n = \min\{a_{n-1}, x_n\} \quad n = 1,2,3,4$$

$$b_n = \max\{b_{n-1}, x_n\} \quad n = 1,2,3,4$$

output pixel

$$l = \begin{cases} 0 \text{ (flat)} & \text{if } a_4 = x_0 = b_4 \\ 1 \text{ (nonmaximum)} & \text{if } a_4 = x_0 < b_4 \\ 2 \text{ (nonminimum)} & \text{if } a_4 < x_0 = b_4 \\ 3 \text{ (transition)} & \text{if } a_4 < x_0 < b_4 \end{cases}$$

6.3.2 Relative Extrema Operator

relative extrema operators: relative maximum and minimum operators
relative extrema: recursive operator, numeric data domain
relative extrema: input not changed, output successively modified
top-down, left-right scan, then bottom-up, right-left scan until no change
output: value of highest extrema reachable by monotonic path
relative extrema pixels: output same as input pixels
pixel designations for the normal and reverse scans
=====Fig. 6.19=====
two primitive functions h and max

$$h(b,c,d,e) = \begin{cases} \max\{d,e\} & \text{if } c \geq b \\ d & \text{if } c < b \end{cases}$$

$\max\{d,e\}$: use new maximum value when ascending
d: keep original maximum when descending

4-connected, output $l = a_2$

$$a_0 = l_0$$

$$a_n = h(x_0, x_n, a_{n-1}, l_n), \quad n = 1 \text{ and } 2$$

=====Fig. 6.20=====

6.3.3 Reachability Operator

successively propagating labels that can reach by monotonic paths

descending reachability operator employs h

$$h(a, b, x, y) = \begin{cases} a & \text{if } (b = g \text{ or } a = b) \text{ and } x < y \\ b & \text{if } a = g \text{ and } x < y \\ c & \text{if } a \neq g, b \neq g, a \neq b, \text{ and } x < y \\ a & \text{if } x > y \end{cases}$$

g: pixels that are not relative extrema labeled background

a: unchanged when neighbor is g or the same with neighbor

b: become first propagated label if originally g

c: common region: more than one extremum can reach it by monotonic path

c: already labeled and neighbor has different label, then two extrema reach

a: propagate from extremum

4-connected, output $l = a_2$

$$a_0 = l_0$$

$$a_n = h(a_{n-1}, l_n, x_0, x_n), \quad n = 1 \text{ and } 2$$

$$\begin{matrix} 9 & 2 & 1 & 2 & 3 & 3 & 1 \\ 2 & 1 & 2 & 3 & 3 & 1 & 2 \\ 1 & 2 & 3 & 3 & 1 & 2 & 3 \\ 2 & 3 & 3 & 1 & 2 & 3 & 9 \end{matrix} \overset{\text{max}}{\Rightarrow} \begin{matrix} 9 & 9 & 9 & 3 & 3 & 3 & 9 \\ 9 & 9 & 3 & 3 & 3 & 9 & 9 \\ 9 & 3 & 3 & 3 & 9 & 9 & 9 \\ 3 & 3 & 3 & 9 & 9 & 9 & 9 \end{matrix} \overset{\text{region}}{\Rightarrow} \begin{matrix} 1 & 1 & 1 & 2 & 2 & 2 & 3 \\ 1 & 1 & 2 & 2 & 2 & 3 & 3 \\ 1 & 2 & 2 & 2 & 3 & 3 & 3 \\ 2 & 2 & 2 & 3 & 3 & 3 & 3 \end{matrix}$$

$$\overset{g}{\Rightarrow} \begin{matrix} 1 & g & g & g & 2 & 2 & g \\ g & g & g & 2 & 2 & g & g \\ g & g & 2 & 2 & g & g & g \\ g & 2 & 2 & g & g & g & 3 \end{matrix} \overset{\text{reachable}}{\Rightarrow} \begin{matrix} 1 & 1 & c & 2 & 2 & 2 & c \\ 1 & c & 2 & 2 & 2 & c & 3 \\ c & 2 & 2 & 2 & c & 3 & 3 \\ 2 & 2 & 2 & c & 3 & 3 & 3 \end{matrix}$$

=====joke=====

6.4 Linear Shift-Invariant Neighborhood Operators

convolution: commutative, associative, distributor over sums, homogeneous

6.4.1 Convolution and Correlation

convolution of an image f with kernel w

$$(f * u)(r, c) = \sum_{(r',c') \in W, (r-r', c-c') \in F} f(r - r', c - c') w(r', c')$$

=====Fig. 6.21=====

w: impulse response function: point spread function

6.4.2 Separability

straightforward computation: $(2M+1) \times (2N+1)$ multiplications, additions
decomposition: $(2M+1) + (2N+1)$ multiplications, additions
=====Fig. 6.22=====

Project due Nov. 16

Write a program to generate Yokoi connectivity number

Project due Nov. 30

Write a program to generate thinned image

Midterm Nov. 23

extent: whatever covered up to Nov. 16 (inclusive)