Binary Image Analysis I

- In many applications, **binary images** can be used as the input to algorithms that perform useful tasks
  - Document analysis
  - Industrial inspection

Mar. 20, 1986 THE THURSDAY REPORT Page 3

Chinese script being computerized

By James Risdon

(The following is part of an on-going series about outstanding graduate students at Concordia.)

Yuan Yan Tang, a 42-year-old Concordia graduate student, is computerizing Chinese writing. And with all its squiggly lines and odd shapes, that's not something that's particularly easy to do. But he's managed so far to input over 2,000 characters in three different styles of Chinese writing into the central Concordia computer.

His program can recognize a Chinese character 98.7 percent of the time. But, more importantly for the advance of computer technology, is that Tang's program can do the work fast — four times faster than another program on the market today.

And that makes his work desirable to people all over the world.

"The commercial aspect of this work is due to the sophistication of our program," says Dr. Ching Y. Suen, the professor of computer science whose work in pattern recognition first attracted Tang to Concordia.

"The patterns could be anything that can be seen, heard or felt: cancer cells, chromosones, x-rays, Landsat photos, or voice patterns," said Suen.

Typical uses of this work could be the early diagnosis of cancers or broken bones; a fingerprint or eye scan identification for instant bank tellers; or an identification system of voice patterns that could allow business people to complete transactions entirely over the phone.

Tang, who is the recipient of a Concordia Fellowship, earned his Master's in Electrical Engineering in 1982 and has written books about computers.

Student co-op opens

By Simon Twiston Davies

On the fourth floor in the EN Annex at 2040 Mackay Street you can get some pretty good deals if you're searching for some of those academic peripherals so often overpriced.

The first student co-operative at an anglophone university, Co-operative de l'Université Concordia, can supply you with a computer, typewriter, calculator or even felt-tip pens and folders — all at rock bottom prices.

Alain Lévesque, the Secretary, explains that the co-operative was formed last September solely to service Engineering students. However, it soon became obvious that there was a need for such an organization for the whole University.

Membership costs a mere $11 and is more or less for life, says Lévesque. "When you pay your $11, you become a shareholder in the Co-op. Nine dollars is refundable when you graduate from the University, but a lot of people continue to use a co-op long after graduating," he adds.

Membership entitles you to a reduction of about 20% on everything in the co-op's catalogue. If you aren't a member, you can still buy at the co-op but must pay the full price. Membership is open to all Concordia students, faculty and staff.

"We haven't had that much publicity yet," says Lévesque, "so that, so far, our membership isn't too big."

At the present time, only seven volunteers are running

(See "Co-op" on page 4)

Shakespeare's Italy is topic

The Department of English will present a public lecture by Dr. Maurice Charney, Distinguished Professor of English at Rutgers University, on Wed., Mar. 26. The topic will be "Shakespeare's Italy and the Venice of Shylock and Othello." The lecture will be given at 8:30 p.m. in Room 435 of the Hall Building, 1455 de Maisonneuve Boulevard West. A reception will follow in the Faculty Club.

Charney is the author of a number of books and several dozen articles on Shakespeare and related subjects. He is a consultant to the National Endowment of the Humanities in Washington, D.C., and to several publishers, including the University of Toronto Press. In 1981, he taught a course on "Shakespeare and his Contemporaries" in Concordia's Summer institute.

Charney will be in Montreal to conduct a seminar at the annual meeting of the Shakespeare Association of America, which will take place at the Ritz Carlton Hotel from March 27-30.

AT A GLANCE

Assoc. Prof. Harold M. Angell, Political Science, will deliver a paper entitled "Québec Language Minorities and the Law" at an international symposium on "Minorities and the Law from 1867 to the Present," planned by Dawson College and the Law Faculty of E.L.T.E. (Budapest) for May 2-4...Recently Mariela Gutiérrez, a Sessional Lecturer from Modern Languages and Linguistics, gave a lecture at the Department of Hispanic Studies of McGill Unviersity, dealing with Afro-American culture and literature...

CUNASA, the Concordia Unviersity Non-Academic Staff Association, is looking for nominations for positions on the executive and classification council for the 1986-87 term. The deadline is 4p.m. Frid., Apr. 4. Send nominations to P. Verret, Chief Returning Officer, SGW C-521...

Rector Patrick Kenniff and Mrs. Kenniff were the guests of honor at the 152nd ball of the St. Patrick's Society last Friday night at the Château Champlain. More than 480 guests attended the event, the proceeds going to the St. Patrick's Society charities, which include St. Mary's Hospital, Montreal Convalescent Hospital, Spera Foundation, Birthright of Montreal, Dawson Boys' and Girls' Club and the Bishop Leonard J. Crowley Fund.

A 24-hour dance-a-thon starts tomorrow night in the Hingston Hall cafeteria, west-end campus, at 8 p.m. and continues until Saturday night, with the proceeds slated for the Quebec Heart Foundation and the Canadian Cancer Society, reports organizer Bonnie Ste-Croix. Dancers, sponsors and spectators are all welcome as well as food donations to keep the dancers going — whether for an hour or 24 hours. Sign-up sheets are available in room 156 at Hingston Hall.

If you're looking for Registrar's Service at the west-end campus, you'll find them at AD 211 instead of their former location, CC 214...

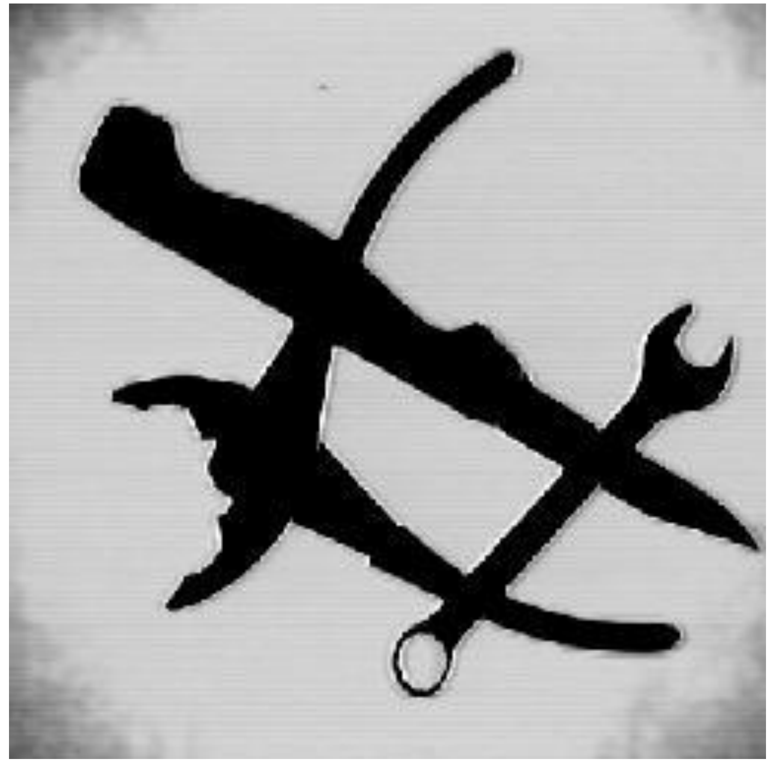
Nancy Allison, who teaches voice, singing and speech for the Theatre Department will be guest soprano soloist at a concert by the Shouer de Montréal and the Metropolitain Orchestra at 8 p.m. on Mar. 23 and 26 in Eglise St. Viateur in Outremont. A concert will also be given on Mar. 19 in Lachine. For more information, call 381-5502.

...The Canadian Society for the Prevention of Cruelty to Animals (SPCA) will have information booths at Concordia on Mar. 24 and 25 at the main entrance of the Hall Building and the Campus Centre at the west-end campus.

Women's research institute announces grants, conference

The Canadian Research Institute for the Advancement of Women (CRIAW) announces:

1) Research Grants-in-Aid: A small number of grants are being offered for projects that promote the advancement of women ($2,000 each).
- The project must make a significant contribution to feminist research;
- be non-sexist in methodology and language;
- take place in Canada or should concern Canada;
- the research design and content must meet appropriate standards.
Deadline is August 31, 1986
2) The 10th CRIAW Conference, University of Moncton, November 7-9, 1986. Call for Papers.
Deadline for receipt of abstracts March 28, 1986
Address:
Isabelle McKee-Allain
Département de sociologie
Centre universitaire de Moncton
Moncton, Nouveau-Brunswick
E1A 3E9



- These algorithms can handle a wide range of basic tasks in image analysis, including:
  - Simple counting tasks
  - Complex tasks :
    - recognition
    - localization
    - Inspection

- By studying binary image analysis before going on to gray-tone and color images, one can gain insight into the entire image analysis process

Binary Image Processing -Advantages

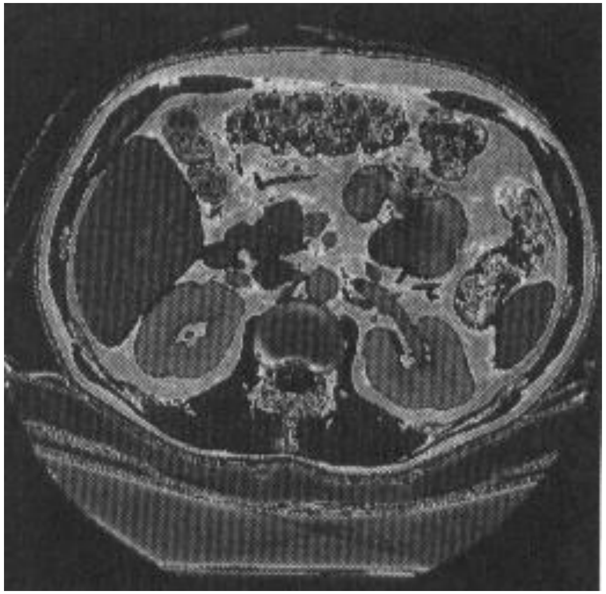
- Easy to acquire
- Low memory requirement
- Simple processing

Binary Image Processing-disadvantages

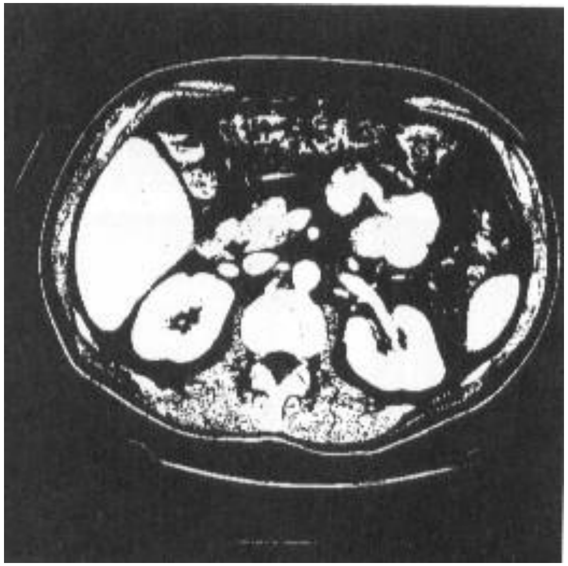
- Limited application
- Cannot extend to 3D
- Specialized lighting is required

1. Pixels and Neighborhoods

- A binary image **B** can be obtained from a gray-scale or color image **I** through an operation that selects a subset of the image pixels as ***foreground*** pixels, the pixels of interest in an image analysis task, leaving the rest as ***background*** pixels to be ignored.



Thresholding



- The pixels of a binary image **B** are **0**s and **1**s; The **1**s will be used to denote **foreground pixels**, and the **0**s **background pixels**.

- In many algorithms, not only the value of a particular pixel, but also the values of its **neighbors** are used when processing that pixel.
  - ***4-neighbors***
  - ***8-neighbors*** of a pixel

	N	
W	*	**E**
	S	

4-neighbors

NW	**N**	**NE**
W	*	**E**
SW	**S**	**SE**

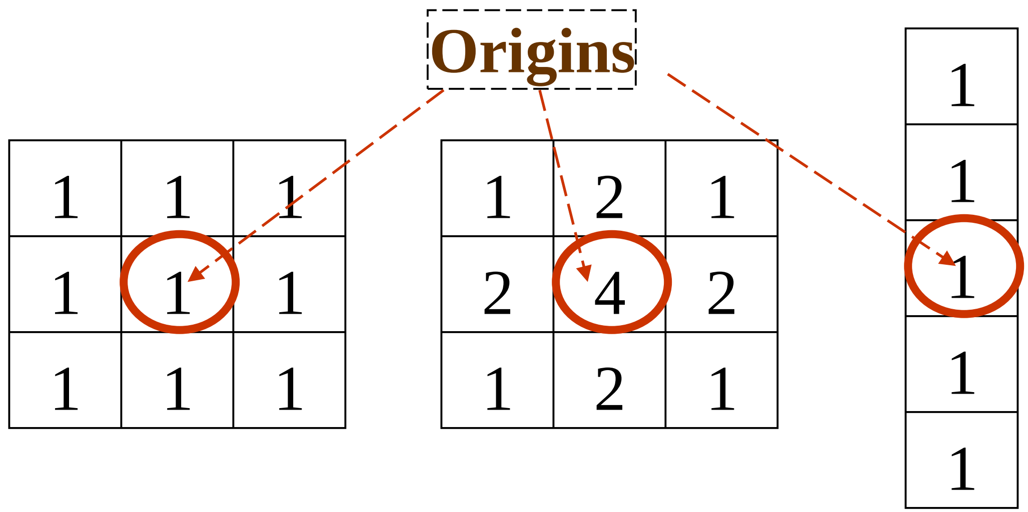
8-neighbors

- The **4-neighborhood N4[r, c]** of pixel [r, c] includes pixels [r - 1, c], [r + 1, c], [r, c -1], and [r, c + 1], which are often referred to, as its **north, south, west,** and **east** neighbors, respectively.

- The **8-neighborhood N8 [r, c]** of pixel [r, c] includes each pixel of the **4-neighborhood plus** the **diagonal neighbor pixels** [r - 1, c -1], [r - 1, c + 1], [r + 1, c - 1], and [r + 1, c + 1], which can be referred to as its **northwest, northeast, southwest,** and **southeast** neighbors, respectively.

2. Applying Masks to Images (filtering)

- ***Mask:*** A mask is a small matrix whose values are called ***weights***
- Each mask has an ***origin***, which is usually one of its positions.
  - The origins of symmetric masks are usually their center pixel position.
  - For non-symmetric masks, any pixel location may be chosen as the origin (depending on the intended use)

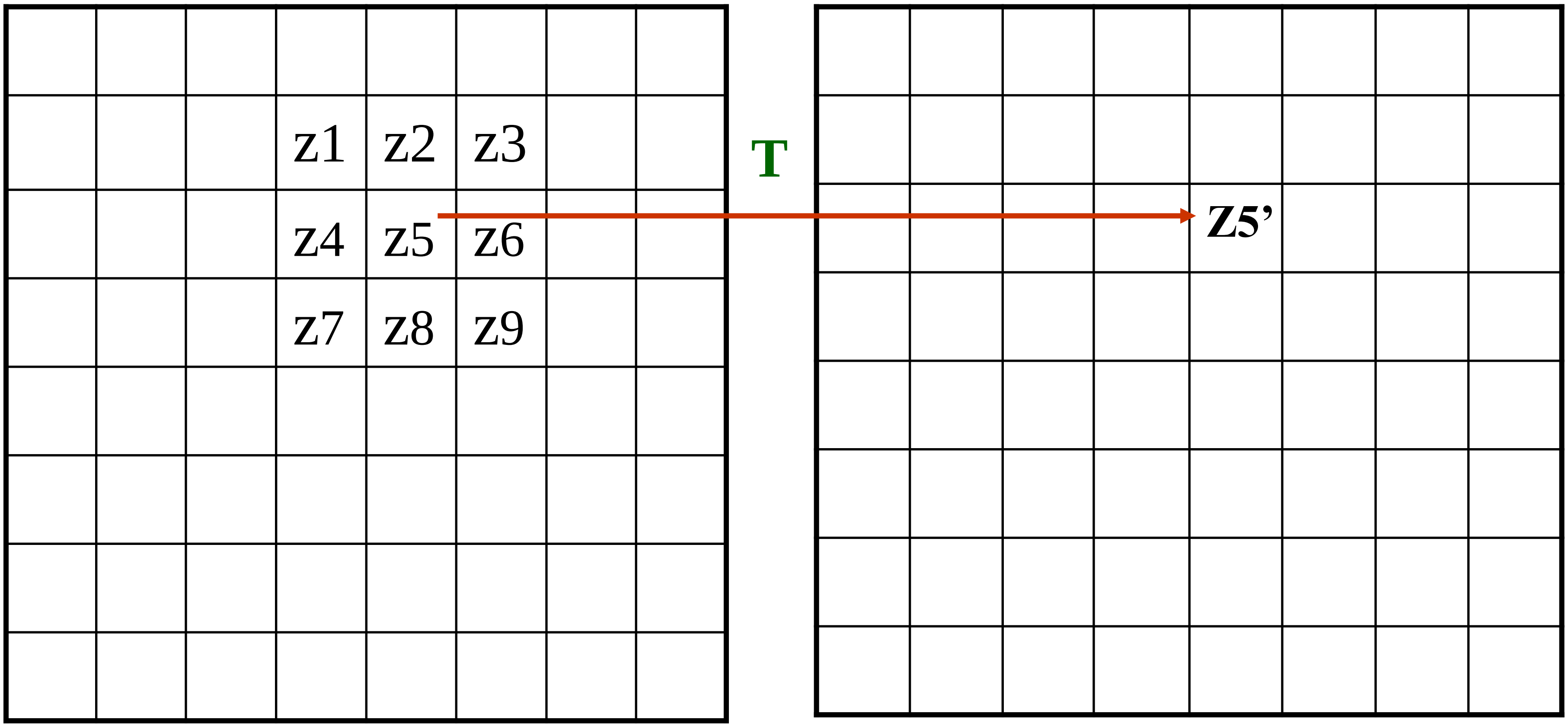

Origins
1 1 1
1 1 1
1 1 1
1 2 1
2 4 2
1 2 1
1
1
1
1
1

- Applying a mask to **smooth** an image
- The application of a mask to an input image produces an output image of the same size as the input.
  - For each pixel in the input image, the mask is conceptually placed on top of the image with its origin lying on that pixel
  - The values of each input image pixel under the mask is multiplied by the weight of the corresponding mask pixel

- The results are summed together to yield a single output value that is placed in the output image at the location of the pixel being processed on the input

$h(x,y) = T[f(x,y)]$

T operates on a neighborhood of pixels



w1	w2	w3
w4	w5	w6
w7	w8	w9

$$z5' = R = w1z1 + w2z2 + ... + z9w9$$

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

$f[.,.]$

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								

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

1	1	1
1	1	1
1	1	1

$f[.,.]$

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							

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

$f[.,.]$

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						

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

$f[.,.]$

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					

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

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$f[.,.]$

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					

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

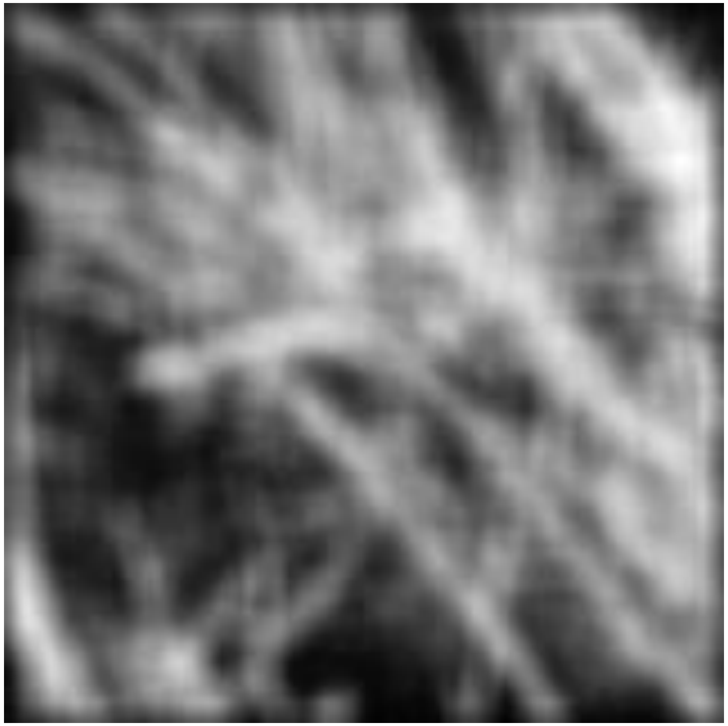
$f[.,.]$

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	

- What does it do?
  - Replaces each pixel with an average of its neighborhood
  - Achieve smoothing effect (remove sharp features)



Practice with linear filters

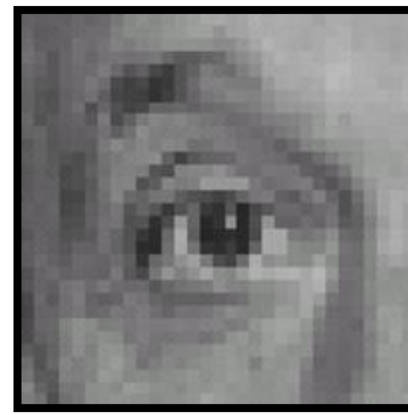
Original

0	0	0
0	1	0
0	0	0

?

Original

0	0	0
0	1	0
0	0	0



Filtered
(no change)

Original

0	0	0
0	0	1
0	0	0

?

Original

0	0	0
0	0	1
0	0	0

Shifted left
By 1 pixel

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

?

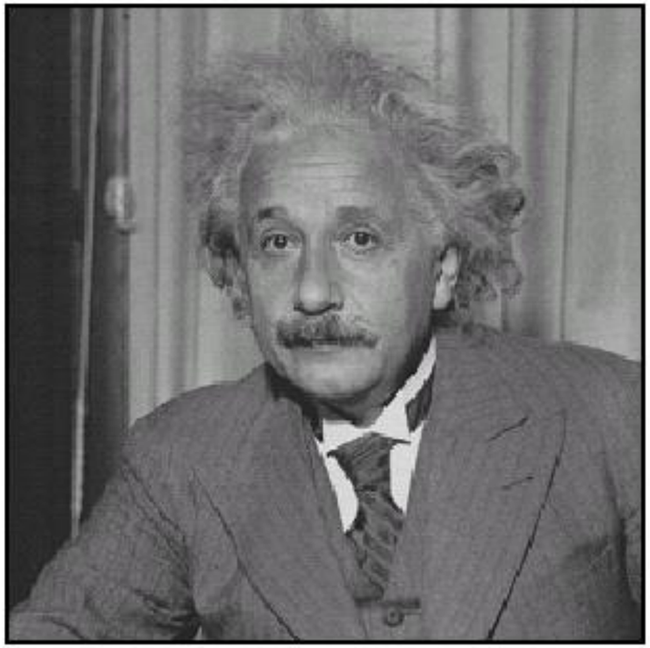
Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

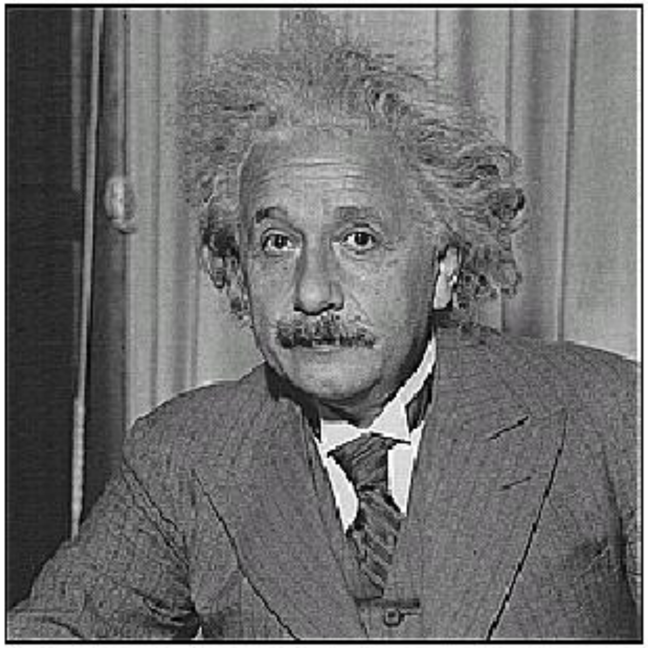
Sharpening filter

- Accentuates differences with local average

Sharpening

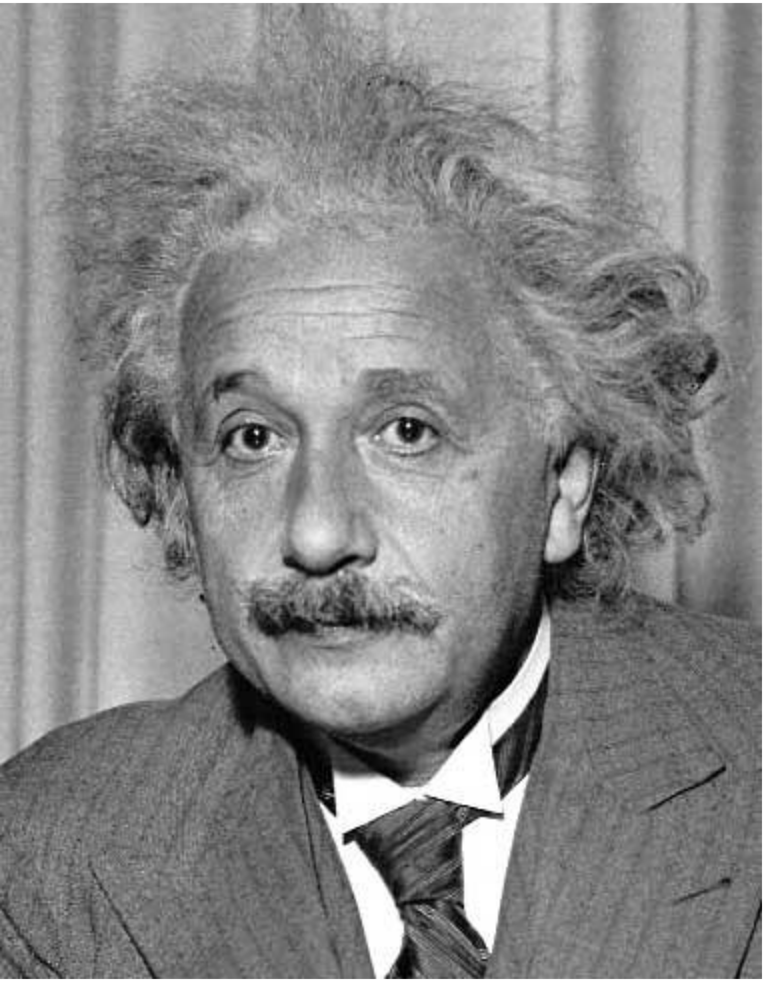


before



after

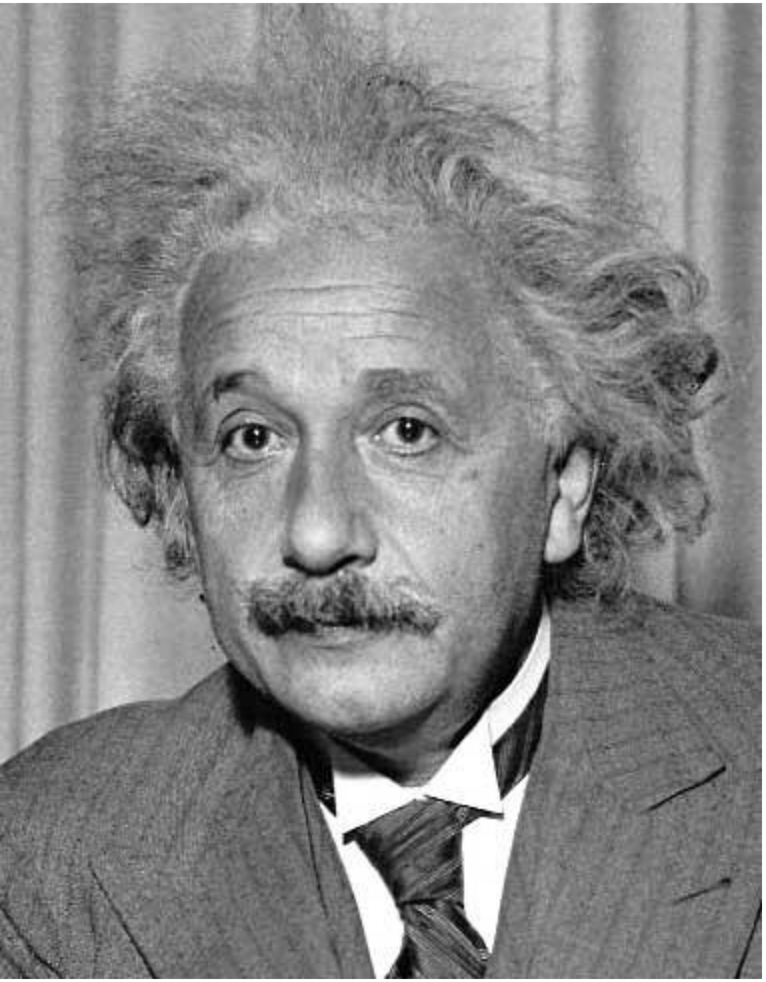
Other filters



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

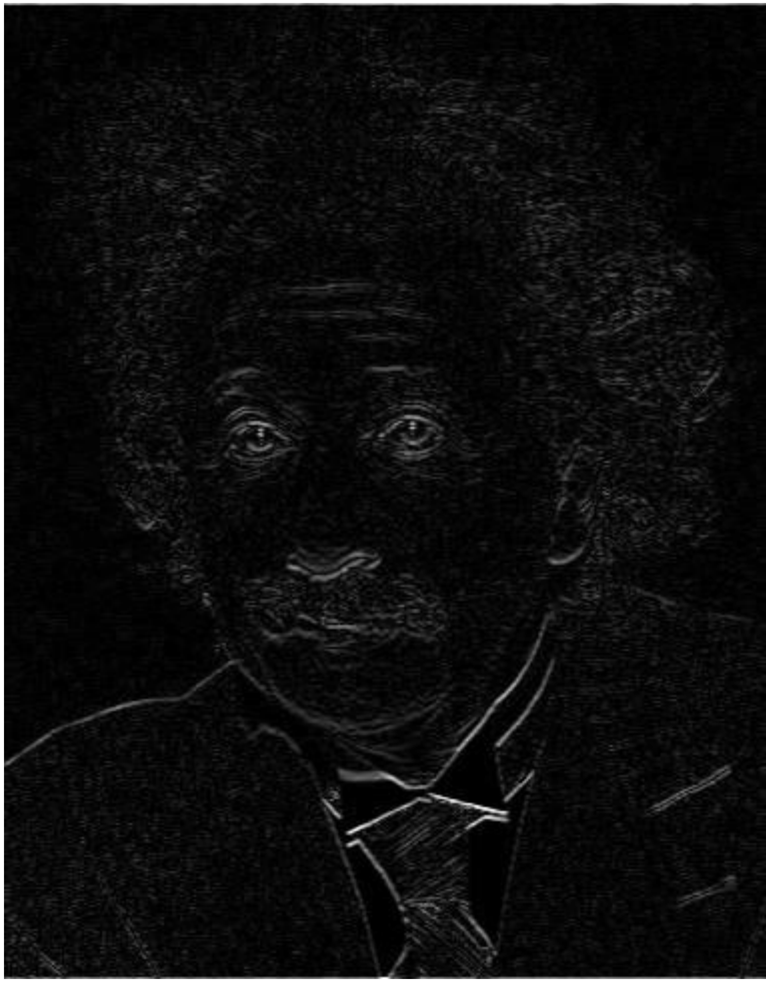
Sobel

Vertical Edge
(absolute value)



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

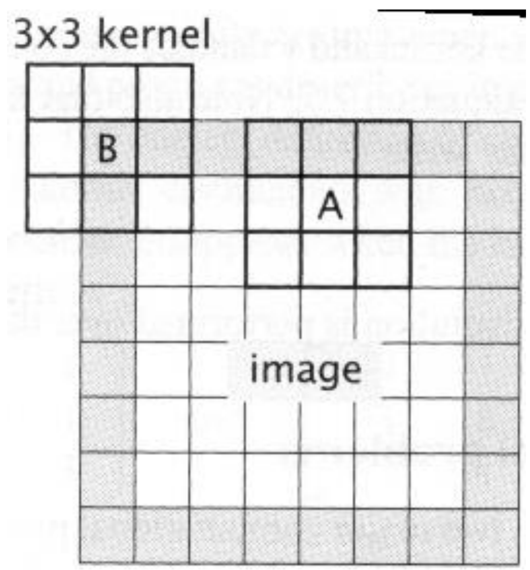
Sobel



Horizontal Edge
(absolute value)

How to treat the image borders?

- **Step 1:** In order to make the output image come out the same size as the input image, we must add some virtual rows and columns to the input image around the edges.



40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80

- **Step 2:** The values in these virtual rows and columns can be set arbitrarily to zero or some other constant or, as has been done here, they can merely duplicate the closest row (or column) to them.

40 1	40 2	40 1	80	80	80	80
40 2	40	40	80	80	80	80
40 1	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80

1	2	1
2	4	2
1	2	1

$1 \times 40 + 2 \times 40 + 1 \times 40 +$
$2 \times 40 + 4 \times 40 + 2 \times 40 +$
$1 \times 40 + 2 \times 40 + 1 \times 40$

40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80
40	40	40	80	80	80	80

1	2	1
2	4	2
1	2	1

Image smooth

640	800	1120	1280	1280
640	800	1120	1280	1280
640	800	1120	1280	1280
640	800	1120	1280	1280
640	800	1120	1280	1280

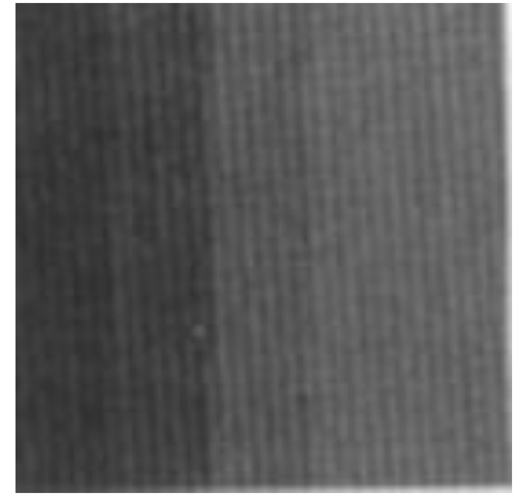
- **Step3:** To normalize, we can divide the value obtained for each pixel by the sum of the weights in the mask, in this case 16, obtaining the final image

1	2	1
2	4	2
1	2	1

16

640	800	1120	1280	1280
640	800	1120	1280	1280
640	800	1120	1280	1280
640	800	1120	1280	1280
640	800	1120	1280	1280

$640 \div 16 = 40$



40	50	70	80	80
40	50	70	80	80
40	50	70	80	80
40	50	70	80	80
40	50	70	80	80

3. Counting the Objects in an Images

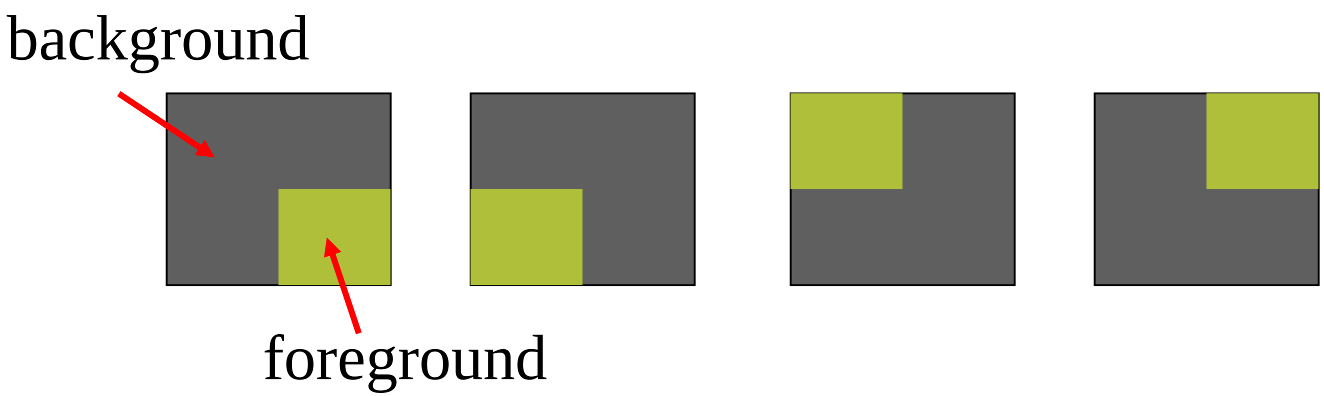
- Counting the number of background objects is an equivalent problem that can be performed with the same algorithm by merely swapping the roles of the two sets: **E** and **I**.
- **E** --- the external corner patterns are 2×2 masks that have three 0s and one 1-pixel.

0	0
0	1

0	0
1	0

1	0
0	0

0	1
0	0



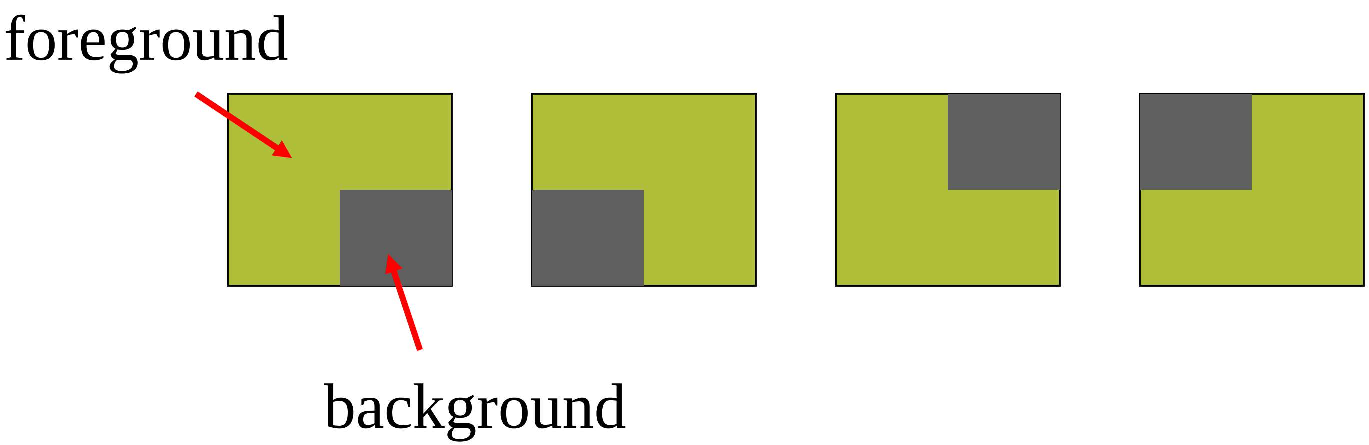
- **I** --- the internal corner patterns are 2×2 masks that have three 1s and one 0-pixel.

1	1
1	0

1	1
0	1

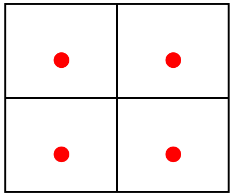
1	0
1	1

0	1
1	1



Algorithm: Counting foreground objects

- **Compute the number of foreground objects of binary image B.**
- Objects are 4-connected and simply connected.



- **Step1**: Counting the number of external corners (E) in region of 2×2 sub-images
- **Step2**: Counting the number of internal corners (I) in region of 2×2 sub-images
- **Step3**: Counting the number of foreground objects in whole image:

$$\text{The number of foreground objects} = \left| \frac{I - E}{4} \right|$$

Algorithm: Counting foreground objects

```
procedure count-objects(B);
{       E := 0;
        I := 0;
        for L:= 0 to MaxRow - 1
           for P:= 0 to MaxCol - 1
           {
             if external-match(L, P) then E:= E + 1;
             if internal-match(L, P) then I:= I + 1;
           };
        return ((I - E) / 4);
}
```

Function external-match(L, P)

- It sequences through the four external masks and returns *true* if the sub-image with top left pixel [L, P] matches one of them, *false* otherwise.

0	0
0	1

0	0
1	0

1	0
0	0

0	1
0	0

Function internal-match(L, P)

- Similarly, the function internal-match(L, P) returns *true* if the subimage with top left pixel [L, P] matches one of the four internal masks, and *false* otherwise.

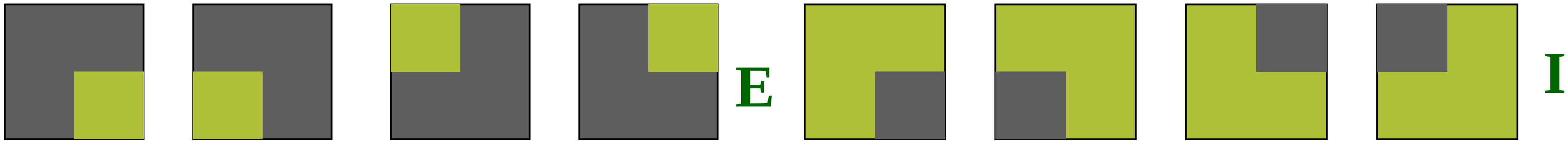
1	1
1	0

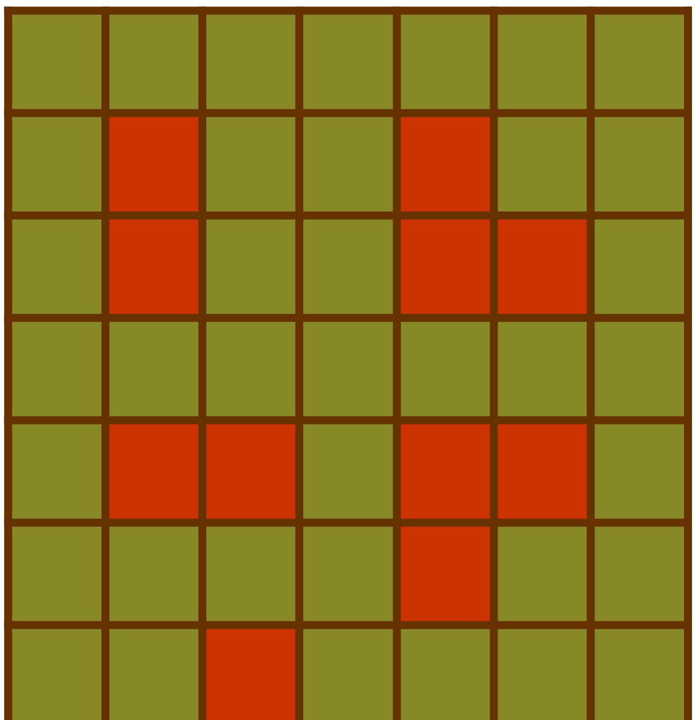
1	1
0	1

1	0
1	1

0	1
1	1

- Example





E = 4

E = 1, I = 1

E = 4

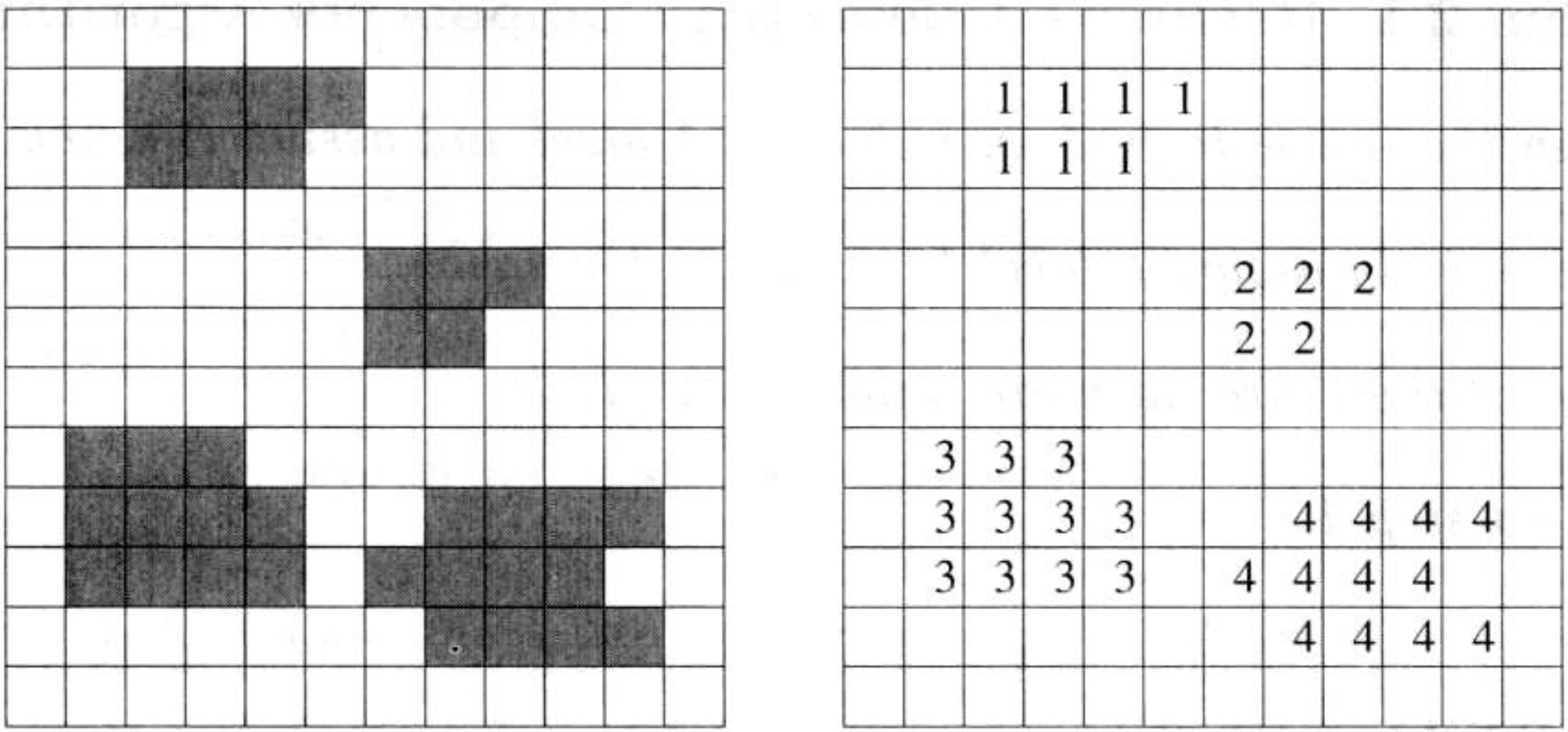
E = 4

E = 3, I = 1

E = 4

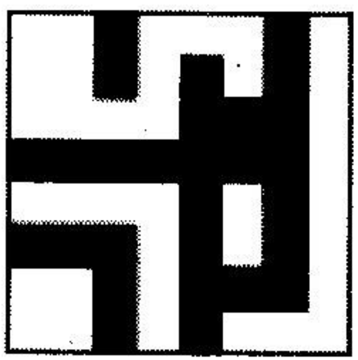
$|(I - E) / 4| = |(2 - 20) / 4| \approx 4$ objects

4. Connected Components Labeling



(a) (b)

- **Definition** **---** A ***connected components labeling*** of a binary image **B** is a labeled image **LB** in which the value of each pixel is the label of its connected component
- A label is a symbol that uniquely names an entity. While character labels are possible, positive integers are more convenient and are most often used to label the connected components.



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

1	1	0	1	1	1	0	2
1	1	0	1	0	1	0	2
1	1	1	1	0	0	0	2
0	0	0	0	0	0	0	2
3	3	3	3	0	4	0	2
0	0	0	3	0	4	0	2
5	5	0	3	0	0	0	2
5	5	0	3	0	2	2	2

- There are a number of different algorithms for the connected components labeling operation.
- Some algorithms assume that the entire image can fit in memory and employ a simple, ***recursive algorithm*** that works on one component at a time, but can move all over the image while doing so.

- Other algorithms were designed for larger images that may not fit in memory and work on only two rows of the image at a time.

- Two different algorithms are commonly used:
  - The ***recursive*** search algorithm
  - A ***row-by-row*** algorithm that uses a special union-find data structure to keep track of components

Recursive Labeling Algorithm

- Suppose that **B** is a binary image with MaxRow + 1 rows and MaxCol + 1 columns.
- We wish to find the connected components of the 1-pixels and produce a labeled output image **LB** in which every pixel is assigned the label of its connected component

1: To negate the binary image, so that all the “1” pixels become “-1”s.

- This is needed to distinguish unprocessed pixels (-1) from those of component label 1.
- This is accomplished with a function called *negate* that inputs the binary image **B** and outputs the negated image **LB**, which will become the labeled image.

2: Finding the connected components becomes one of

- finding a pixel whose value is **-1** in **LB**,
- assigning it a new label **1**, and
- calling **procedure *search*** to find its neighbors that have value -1 and recursively repeat the process for these neighbors

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

B: Binary image

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

LB: Negated image

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

- **Recursive Algorithm**

  1.Scan the image to find an unlabeled -1 pixel and assign it a new label L.

  2.Recursively assign a label L to all its -1 neighbors.

  3.Stop of there are no unlabeled -1 pixels

  4.Go to step 1

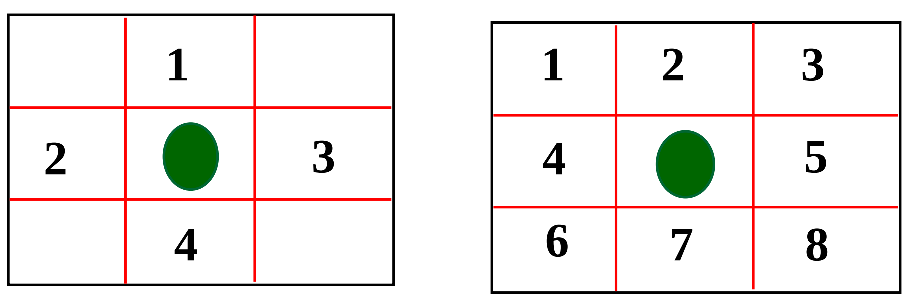
- Features

  -Inefficient for sequential processors/general purpose computers

  -Commonly used on parallel machines.

Function *neighbors* (L, P):

- The utility **function *neighbors* (L, P)** is given a pixel position defined by L and P.
- It returns the set of pixel positions of all of its neighbors, using either the 4-neighborhood or 8-neighborhood definition.
- Only neighbors that represent legal positions on the binary image are returned.
- The neighbors are returned in scan-line order



Attention

- The type of neighborhood you choose affects the number of objects found in an image and the boundaries of those objects. For this reason, the results of many morphology operations often differ depending upon the type of connectivity you specify.

- For example, if you specify a 4-connected neighborhood, this binary image contains two objects; if you specify an 8-connected neighborhood, the image has one object.

0	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**0**	**0**	**0**
0	**1**	**1**	**0**	**0**	**0**
0	**0**	**0**	**1**	**1**	**0**
0	**0**	**0**	**1**	**1**	**0**

- **Compute the connected components of a binary image.**
- This algorithm is a set of five procedures:

recursive-connected-components,

(1) negate,

(2) find-components,

(3) search,

(4) neighbors

(5) print

```
procedure recursive-connected-components (B, LB);
{
LB := negate (B);
label := 0;
find-components (LB, label);
print (LB);
}

procedure find-components (LB, label);
{
for L:= 0 to MaxRow
        for P:= 0 to MaxCol
                if LB[L,P] == -1 then
                {
                label:= label + 1;
                search (LB, label, L, P);
                }
```

```
procedure search (LB, label, L, P);
{
LB[L,P] := label;
Nset:= neighbors (L, P);
for each [L', P'] in Nset
            {
            if LB[L', P'] == -1
            then search (LB, label, L', P’);
            }
}
```

A Row-by-Row Labeling Algorithm

- The classical algorithm
- The algorithm makes ***two passes*** over the image:
  - One pass to record equivalences and assign temporary labels.
  - The second pass to replace each temporary label by the label of its equivalence class.

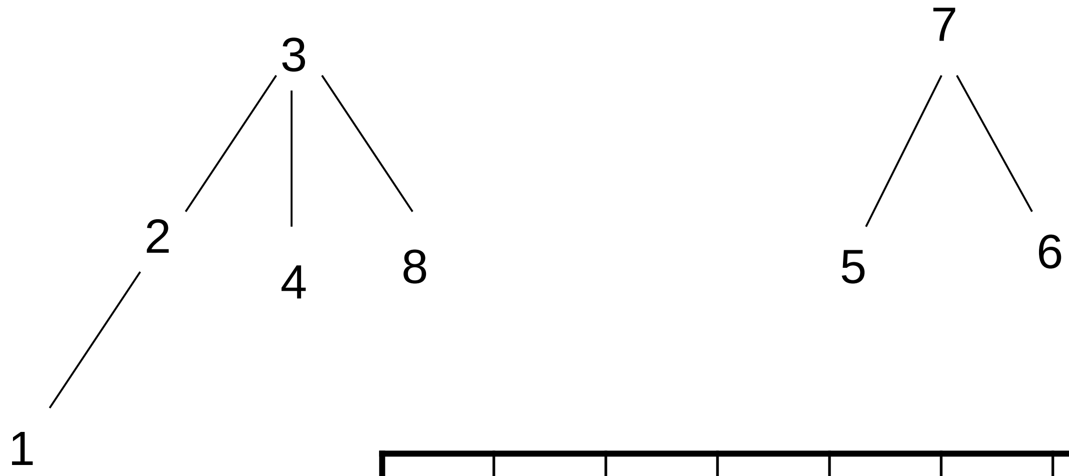
1. Scan the image from left to right, top to bottom; if the pixel is *1* then
    a) if only one of the upper *or* left pixels has a label, copy this label to current pixel
    b) if both have the same label, copy this label
    c) if they have different labels, copy one label and mark these two labels as equivalent
    d) if there are no labeled neighbors, assign it a new label
2. Scan the labeled image and replace all equivalent labels with a common label
3. If there are no neighbors, go to *1*

- Union-Find Structure

  Union: merging two sets into one

  Find: determining which set a particular element is in
- Each set is stored as a tree structure

The union-find data structure for two sets of labels. For each integer Label **i**, the value of **Parent[i]** is the label of the parent of **i** or zero if **i** is a root node and has no parent**.**



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

Parent

- If the roots are not the same, one label is made the parent of the other.
- It is also possible to keep track of the set sizes and to attach the smaller set to the root of the larger set. This has the effect of keeping the tree depths down.

1. The first pass of the algorithm performs label propagation (傳播) to propagate a pixel's label to its neighbors to the right and below it.
2. Whenever a situation arises in which two different labels can propagate to the same pixel, the smaller label propagate and each such equivalence found is entered in the union-find structure

3. A second pass through the image then performs a translation, assigning to each pixel the label of its equivalence class.

Example

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

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

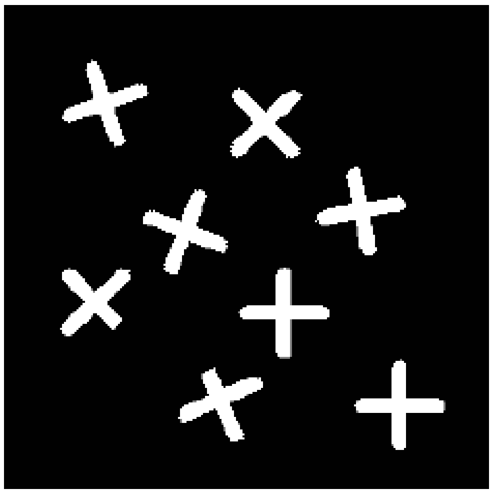
After Pass 1

1	1	0	1	1	1	0	3
1	1	0	1	0	1	0	3
1	1	1	1	0	0	0	3
0	0	0	0	0	0	0	3
4	4	4	4	0	5	0	3
0	0	0	4	0	5	0	3
6	6	0	4	0	0	0	3
6	6	0	4	0	3	3	3

After Pass 2

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

Parent



Original Image

Into Binary