

עיבוד תמונה

הרצאה מס' 4



עיבוד תמונות בינאריות (binary images)

מוטיבציה

- יעיל יותר (מבחינת זיכרון, זמני ריצה וכו') לעבוד על תמונה בינארית מאשר על תמונת אפור (המכילה באופן טיפוסי 256 רמות).
- אינפורמציה בינארית מספיקה במקרים רבים לזיהוי עצמים (object recognition) ולו באופן חלקי.
- רלבנטי במיוחד לאפליקציות ספציפיות (כגון פיקוח תעשייתי, OCR וכו'), בהן קל יחסית להפריד את העצמים מן הרקע (background).

נוטציה:

פיקסלי אובייקט יהיו בעלי ערך 1 לוגי ("שחור") ופיקסלי רקע יהיו בעלי ערך 0 לוגי ("לבן").

הפעלת סף (Thresholding)

- אחת הבעיות המרכזיות בעיבוד תמונה ו/או במערכת ראייה ממוחשבת הינה לגלות תת-תמונות המייצגות אובייקטים. (קל לבני אנוש, קשה למחשב!)
- חלוקת התמונה לאזורים (regions) לקראת סגמנטציה (segmentation) ; זוהי אופרציה מאוד חשובה בהבנת התמונה.
- ההנחה הבסיסית הינה שכל הפיקסלים השייכים ל- regions או segment מסוים הם בעלי תכונה משותפת ; למשל, בעלי תחום עוצמות דומה.

Thresholding (Cont'd)

- הנחת עבודה: נתון אזור בעל עניין (RoI), בו האובייקט מיוצג ע"י פיקסלים באינטרבל עוצמות מסוים, והרקע מכיל פיקסלים שרמות האפור שלהם מחוץ לאינטרבל.
- אזי ניתן לקבל תמונה בינארית מתאימה ע"י הפעלת thresholding. (כל הפיקסלים באינטרבל יקבלו ערך 1 ושאר הפיקסלים יקבלו 0).
- במקרה של תמונה בינארית, פעולת הסגמנטציה שקולה לפעולת ה-thresholding.

Thresholding (Cont'd)

$$B[i, j] = f_T[i, j]$$

where

$$f_T[i, j] = \begin{cases} 1 & \text{if } f[i, j] \leq T \\ 0 & \text{o/w} \end{cases}$$

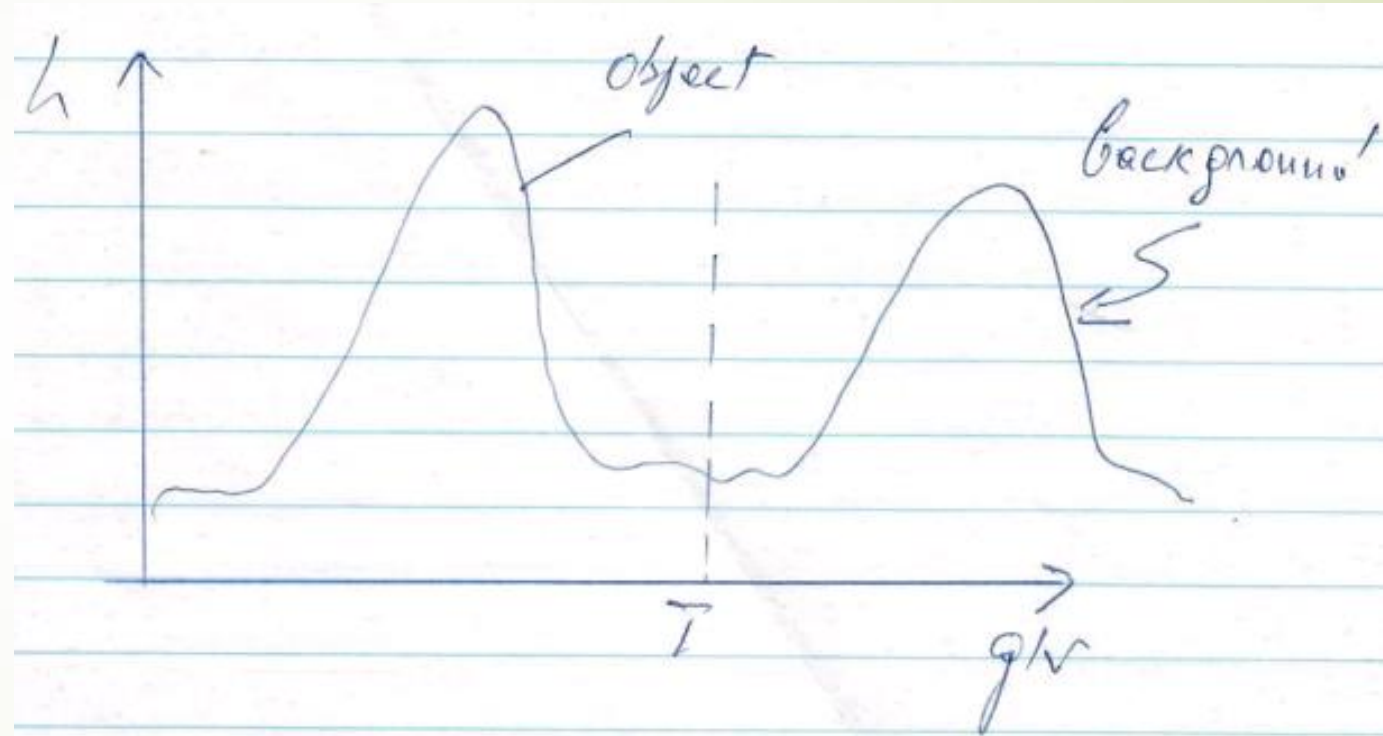
or

$$f_T[i, j] = \begin{cases} 1 & \text{if } T_1 \leq f[i, j] \leq T_2 \\ 0 & \text{o/w} \end{cases}$$

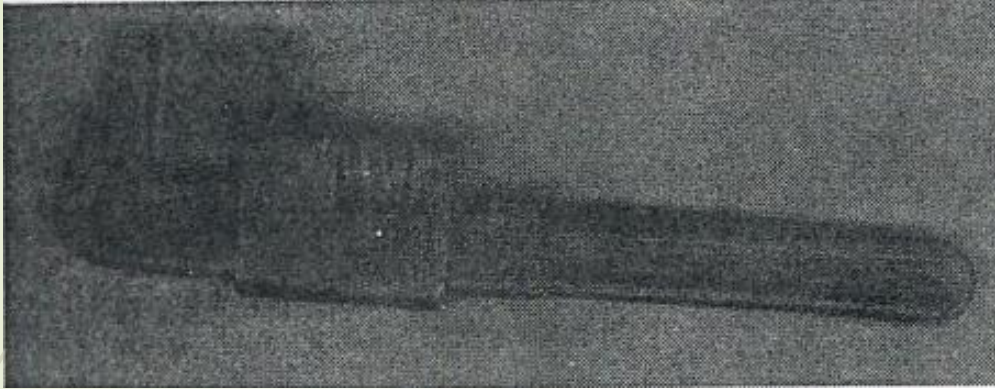
or

$$f_T[i, j] = \begin{cases} 1 & \text{if } f[i, j] \in Z \\ 0 & \text{o/w} \end{cases}$$

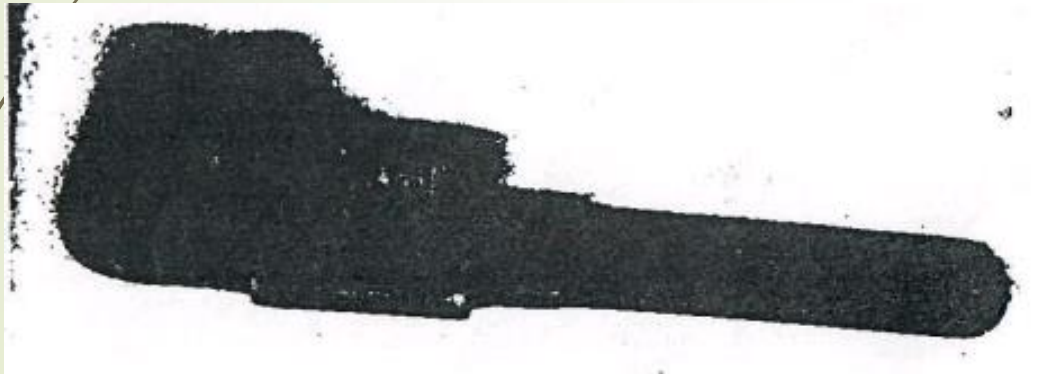
where Z consists of all non-contiguous subintervals containing object gray levels.



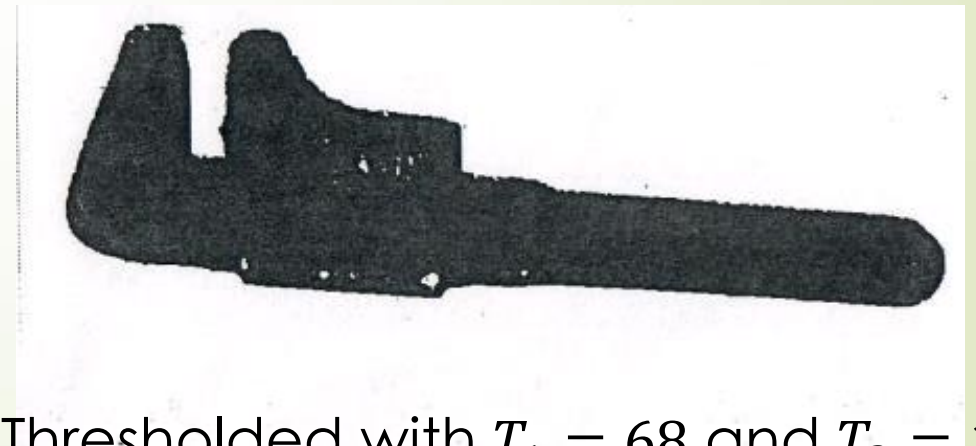
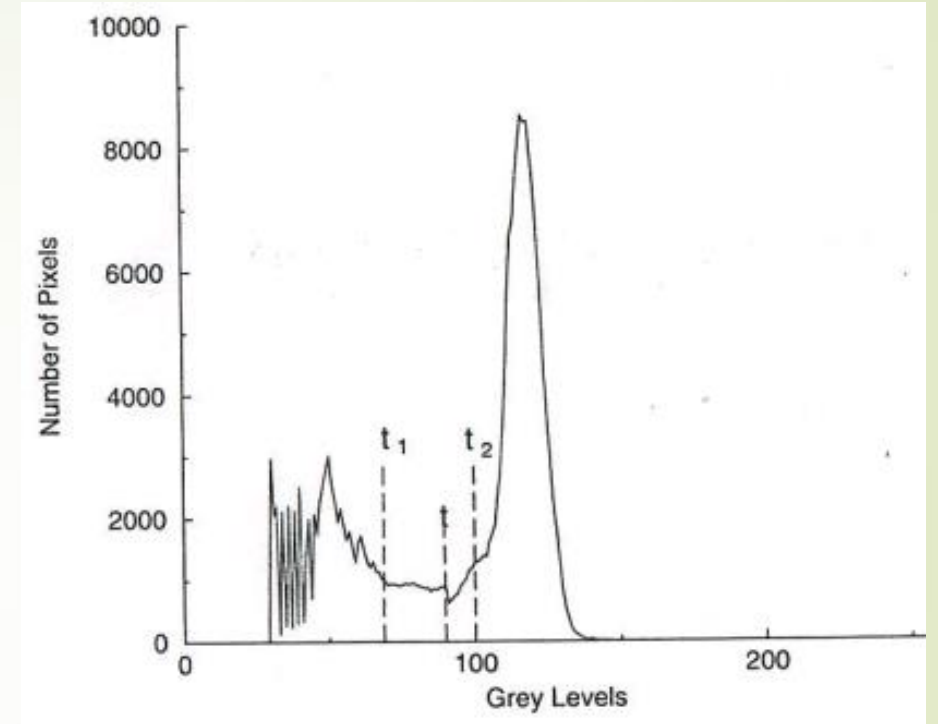
Thresholding (Example1)



Original image

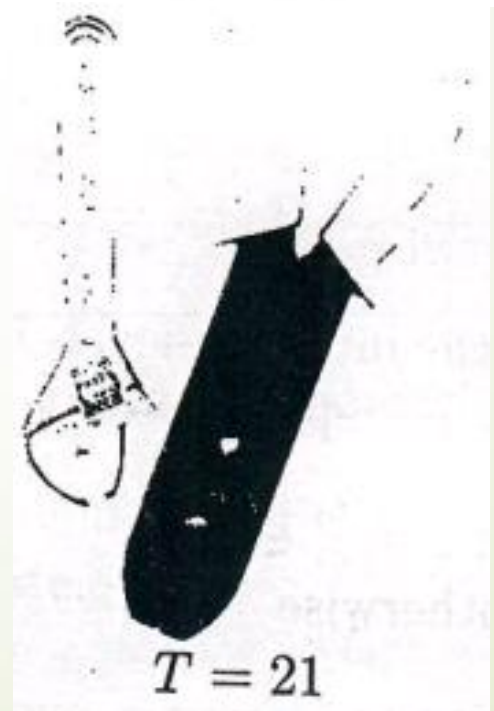


Thresholded with $T = 91$



Thresholded with $T_1 = 68$ and $T_2 = 100$

Thresholding (Example2)



תכונות גיאומטריות

חישוב מספר מאפיינים (קרי, פרמטרים) גיאומטריים עבור פיקסלים השייכים לאובייקט מסוים הינו חיוני לצורך זיהוי (ומיקום) האובייקט.

$$A = \text{area} = \sum_{i=1}^n \sum_{j=1}^m B[i, j]$$

גודל (size):

נקרא גם מומנט מסדר 0.

מאפיינים גיאומטריים (המשך)

➤ קואורדינטות מיקום (מרכז כובד), מומנטים מסדר ראשון:

$$\bar{x} = \frac{\sum_{i=1}^n \sum_{j=1}^m j \cdot B[i, j]}{\sum_{i=1}^n \sum_{j=1}^m B[i, j]} = \frac{\sum_{i=1}^n \sum_{j=1}^m j \cdot B[i, j]}{A}$$

$$\bar{y} = \frac{\sum_{i=1}^n \sum_{j=1}^m i \cdot B[i, j]}{\sum_{i=1}^n \sum_{j=1}^m B[i, j]} = \frac{\sum_{i=1}^n \sum_{j=1}^m i \cdot B[i, j]}{A}$$

➤ כללית, מומנטים מסדר $p + q$:

$$m_{pq} = \frac{\sum_{i=1}^n \sum_{j=1}^m i^p \cdot j^q B[i, j]}{A}$$

מאפיינים גיאומטריים (המשך)

➤ אוריינטציה (orientation): רלבנטי במיוחד עבור צורות/אובייקטים מוארכים (elongated).

➤ ניתן לאמוד זאת ע"י התאמת ישר לפיקסלי האובייקט. למשל, ע"י הפעלת קריטריון השגיאה הריבועית המינימלית (ordinary least squares),

$$\chi^2 = \sum_{i=1}^n \sum_{j=1}^m r_{i,j}^2 B[i, j]$$

באשר r_{ij} מבטא את השגיאה בין נקודת אובייקט $[i, j]$ לישר.

מאפיינים גיאומטריים (המשך)

➤ קומפקטיות (compactness):

$$\frac{p^2}{A} \geq 4 \cdot \pi \quad \text{נמדדת באמצעות היחס}$$

באשר P הינו היקף האובייקט ו- A הוא שטחו.

במקרה המעגלי היחס הוא 4π , כלומר המינימלי האפשרי. כאשר המעגל מוטה, מתקבלת למעשה אליפסה בעלת היקף דומה ששטחה קטן יותר (כלומר, היחס גדל).

במקרה הקיצוני (קו ישר), היחס הינו ∞ .

נראה בהמשך כיצד למצוא היקף אובייקט בתמונה בינארית.

ייצוג תמונה בינארית

- קידוד ע"י "אורכי רצף" (run-length encoding).
• התחלה + אורכי רצף פיקסלי "1" בכל שורה.
• אורכי רצף בכל שורה (כשמתחילים תמיד עם רצף פיקסלי "1").

Binary image:

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

Start and length of 1 runs: (1, 3) (7, 2) (12, 4) (17, 2) (20, 3)
(5, 13) (19, 4)
(1, 3) (17, 6)

Length of 1 and 0 runs: 3, 3, 2, 3, 4, 1, 2, 1, 3
0, 4, 13, 1, 4
3, 13, 6

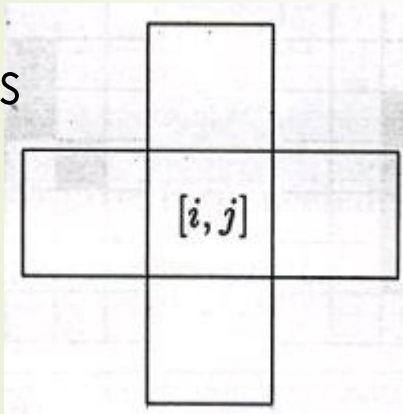
אלגוריתמים "בינאריים"

➤ המטרה: לקבץ (grouping) יחד פיקסלי אובייקטים לתמונת אובייקטים.
ההנחה הבסיסית היא שפיקסלי אובייקט מסוים נמצאים בסמיכות מרחבית (spatial proximity)

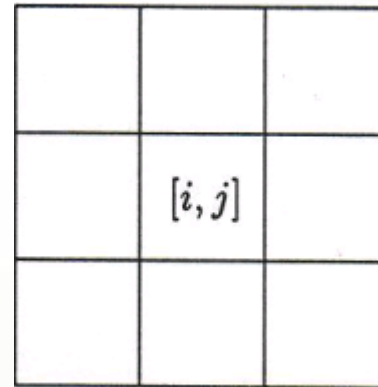
➤ הגדרות ומושגי יסוד:

- שכנות: מטיפוס 4 ומטיפוס 8.

4-neighbors



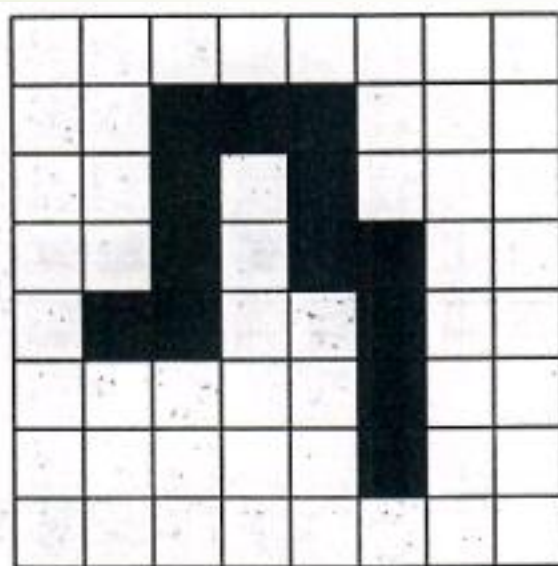
8-neighbors



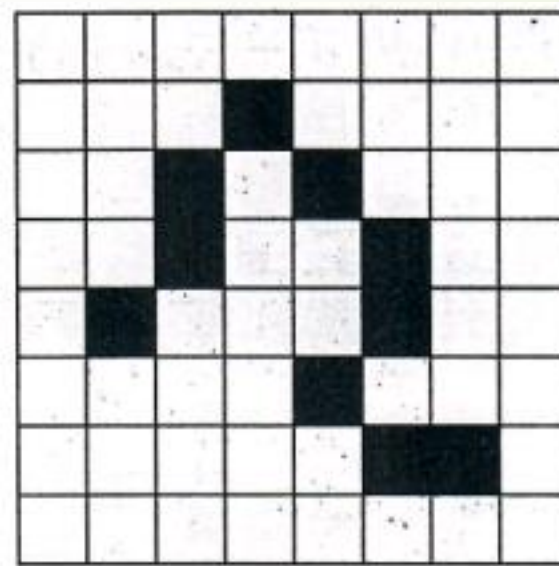
- מסלול (path): סדרת פיקסלים $[i_0, j_0], [i_1, j_1], \dots, [i_n, j_n]$ כך ש- $[i_k, j_k]$ הינו שכן של $[i_{k+1}, j_{k+1}]$ עבור $0 \leq k \leq n - 1$

- foreground: סט כל פיקסלי "1", יסומן ע"י S .

דוגמאות למסלולים מטיפוס 4 ומטיפוס 8



(a) 4-path



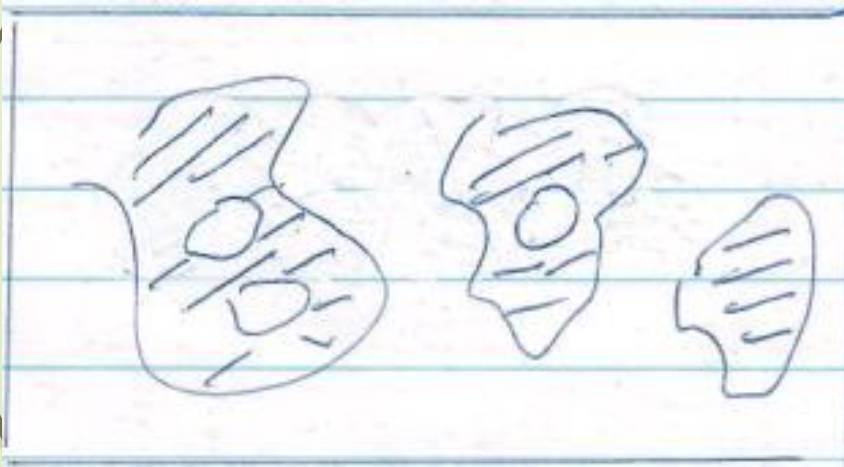
(b) 8-path

מושגי יסוד נוספים בתמונות בינאריות

- קשירות (connectivity)
פיקסל $p \in S$ קשור ל- $q \in S$ אם קיים מסלול מ- p ל- q המכיל אך ורק פיקסלי S .
קשירות מקיימת יחס שקילות (הוכח!)

- מרכיב קשיר (connected component)
סט של פיקסלים בו כל פיקסל קשור לשאר הפיקסלים הינו מרכיב קשיר.

- רקע (background)
סט כל המרכיבים הקשירים ב- \bar{S} (המשלים של S) הכוללים פיקסלי קצוות של התמונה.

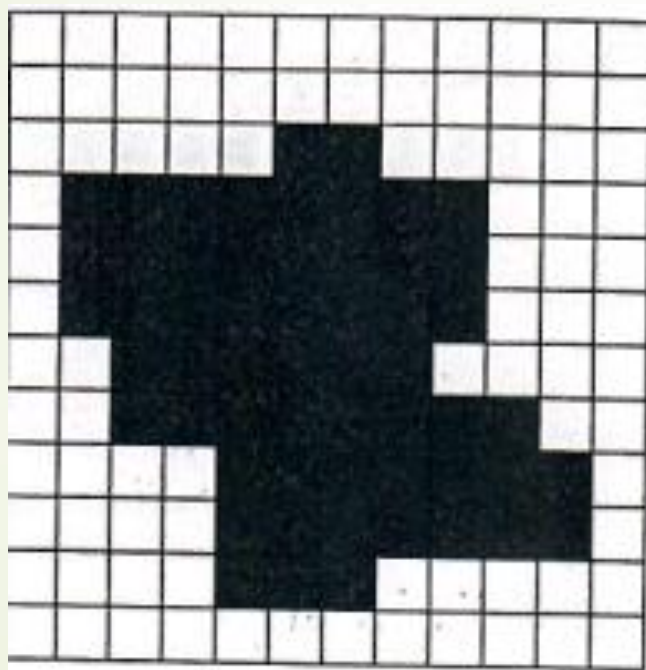


- שאר המרכיבים של \bar{S} נקראים חורים (holes).
איור הדוגמא מכיל 3 מרכיבים קשירים, רקע ו-3 חורים.

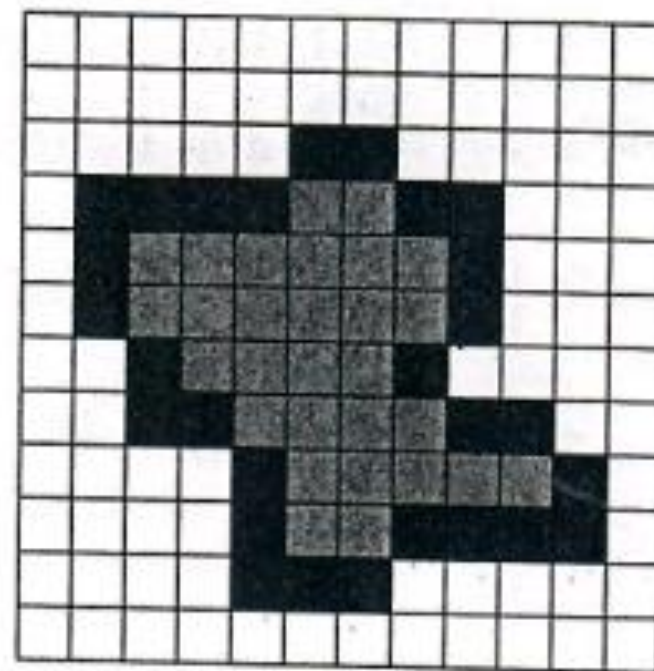
פיקסלי גבול (boundary)

➤ הגדרה:

הגבול של קבוצת פיקסלים S כולל את אותם פיקסלים ב- S שלהם שכני-4 ב- \bar{S} .
נסמן את הגבול ב- S' .



(a) Original image



- (b)
- Boundary pixels
 - Interior pixels
 - Surrounds pixels

אלגוריתם למציאת פיקסלי הגבול/קונטור וההיקף

1. מצא את נקודת ההתחלה $s \in S$ ע"י סריקה שיטתית של התמונה (שמאל ימין ומעלה מטה).
2. יהא פיקסל השפה הנוכחי c ; הצב $c = s$. נסמן ב- b את שכן-4 (ממערב) של s ; $(b \in \bar{S})$
3. יהיו n_1, n_2, \dots, n_8 שמונת שכני-8 של c (עם כיוון השעון), באשר $n_1 = b$; מצא את ה- i הראשון עבורו $n_i \in S$.
4. הצב $b = n_{i-1}, c = n_i$.
5. חזור על צעדים 3 ו-4, עד אשר $c = s$.

אלגוריתם למציאת פיקסלי הגבול/קונטור (דוגמא)

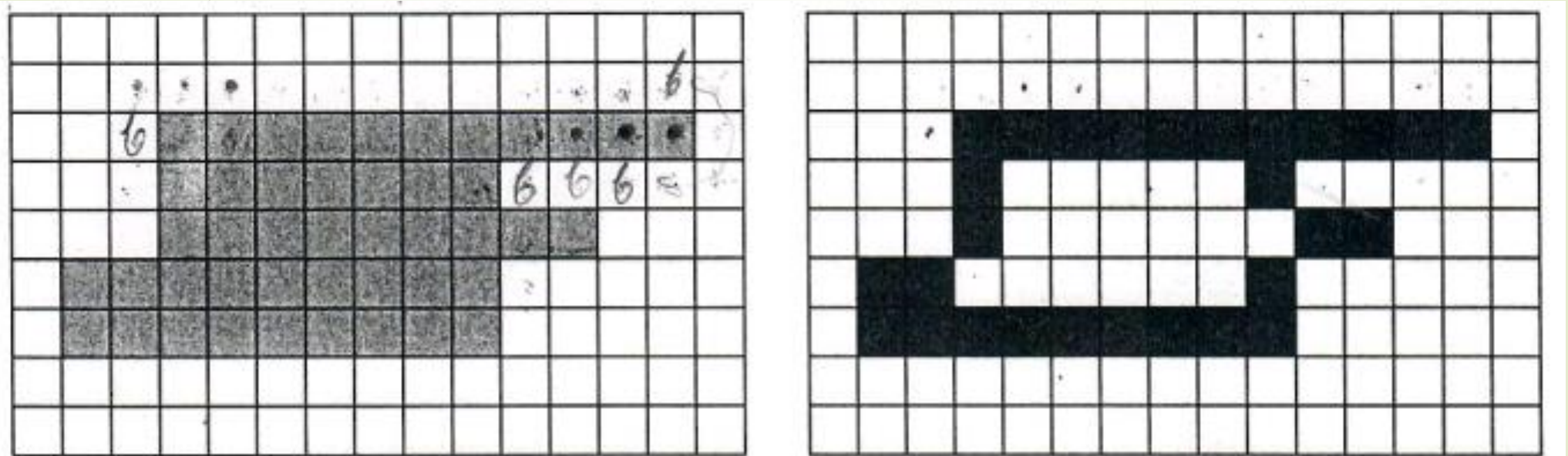


Figure 2.16: Results of a boundary-following algorithm. *Left*: Original binary object. *Right*: Calculated boundary.

➤ ייצוג פיקסלי הגבול/קונטור יכול להעשות ע"י קוד השרשרת (chain code)

אלגוריתם למציאת מרכיבים קשירים (connected component labeling)

➤ המטרה:

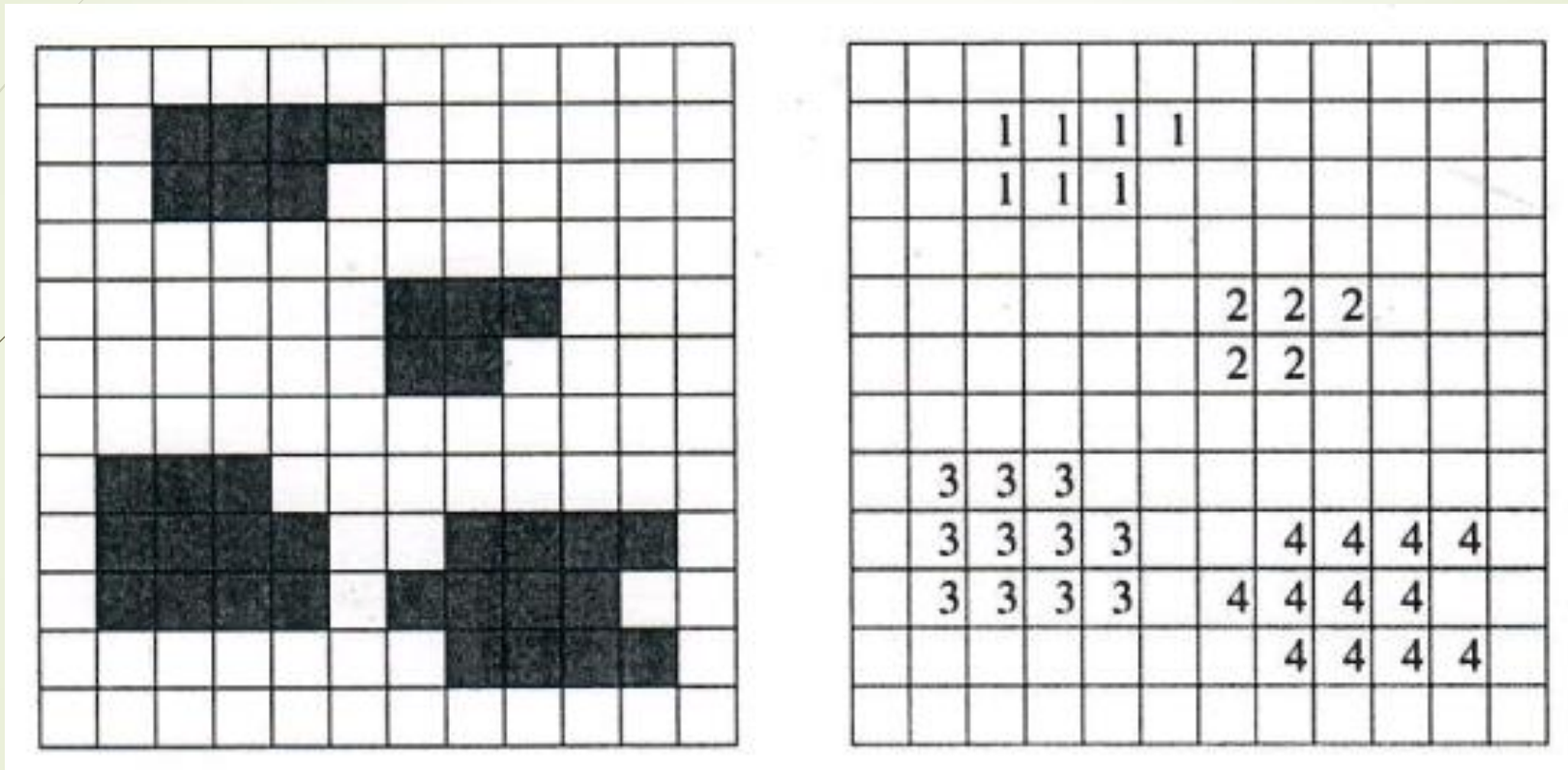
מציאת כל המרכיבים הקשירים בתמונה נתונה, והקניית תווית (label) בלעדית לכל הפיקסלים השייכים לאותו מרכיב קשיר.

➤ המוטיבציה היא לקבל ע"י כך אזורים (regions) בתמונה שעשויים לייצג אובייקטים שונים.

פסבדו-קוד של האלגוריתם (דו-שלבי)

1. סרוק משמאל לימין ומלמעלה למטה.
2. אם הפיקסל הוא "1" (לוגי), אזי:
 - א. אם רק לשכן השמאלי או לשכן שמעל יש תווית, העתק אותה.
 - ב. אם לשני השכנים הנ"ל אותה תווית, העתק אותה.
 - ג. אם לשניהם תוויות שונות, העתק את זו של השכן מעל וסמן את התוויות כשקולות בטבלת השקילות.
 - ד. אחרת, תן תווית חדשה לפיקסל והכנס אותה לטבלת השקילות.
3. חזור ל-2 אם טרם עברת כל הפיקסלים.
4. מצא את התווית בעלת הערך קטן ביותר לכל קבוצת שקילות.
5. סרוק שוב, החלף כל תווית בתווית השקולה שנמצאה בסעיף הקודם.

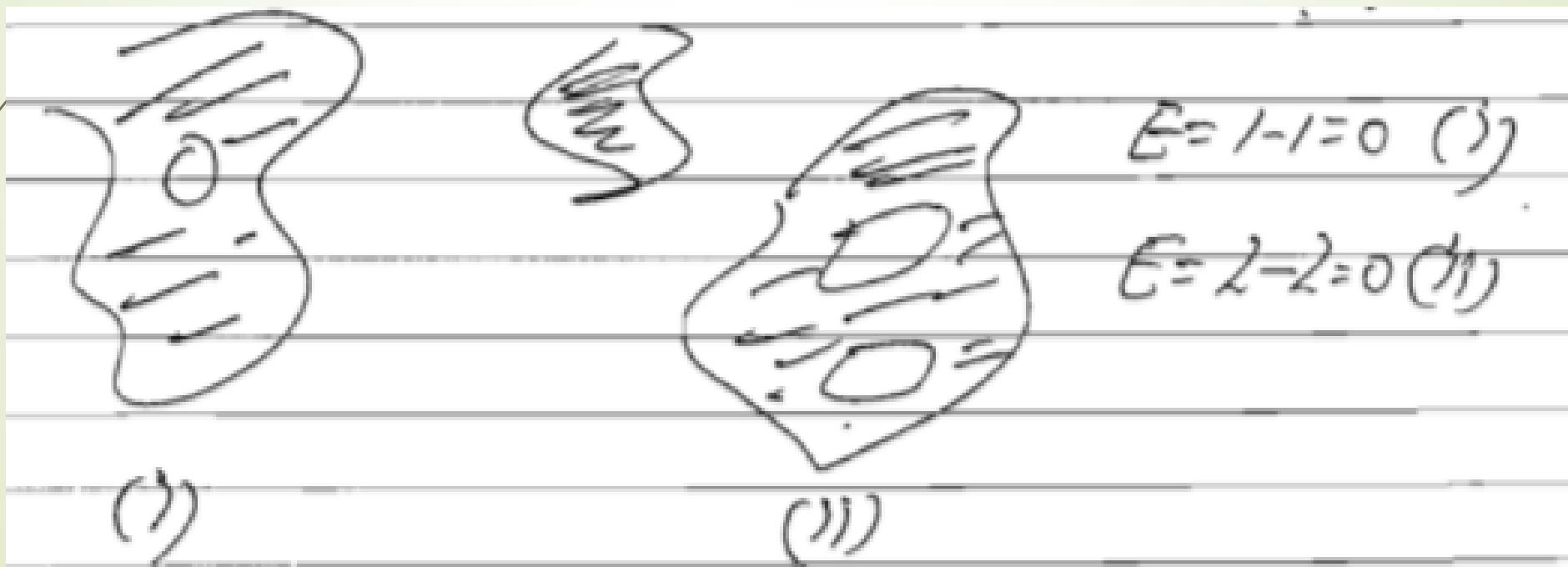
Connected Component Labeling (Example)



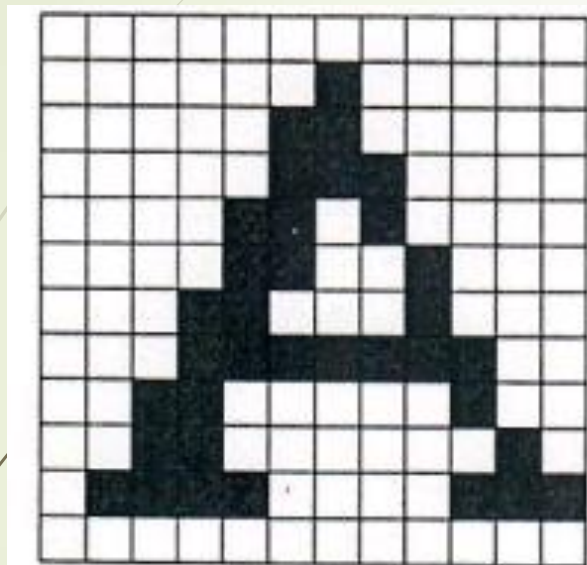
A binary image (left) and its labeled connected components (right).

מאפיין טופולוגי: מספר Euler

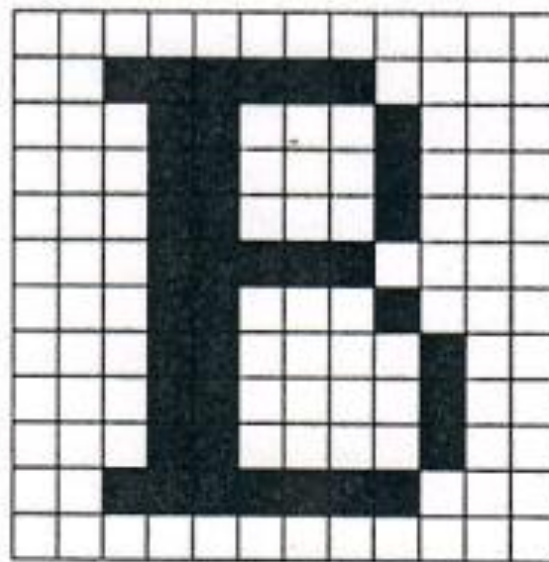
- מספר אוילר (Euler number): פרמטר המשמש לאפיון צורה, $E \triangleq C - H$, כאשר C הינו מס' המרכיבים ו- H שווה למס' החורים.
- שים לב! צורה אינה מתאפיינת בהכרח (באופן חד-חד ערכי) ע"י מספר אוילר.



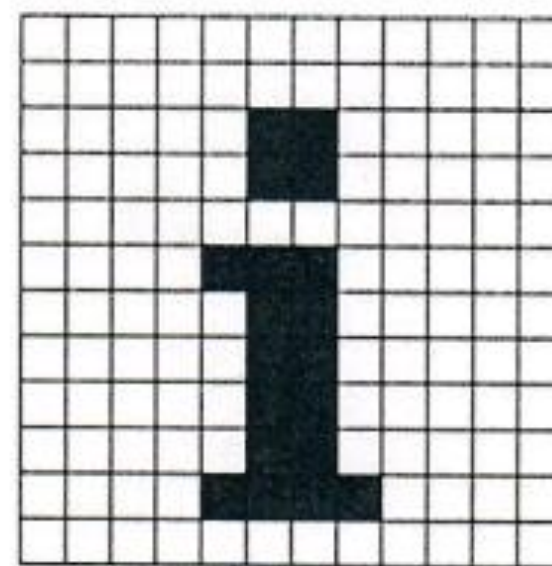
מספר אוילר (דוגמא)



$E = 0$



$E = -1$



$E = 2$

➤ למרות שאינו מאפיין חד-חד ערכי, עשוי לשמש בסיווג תווים בהקשר של OCR

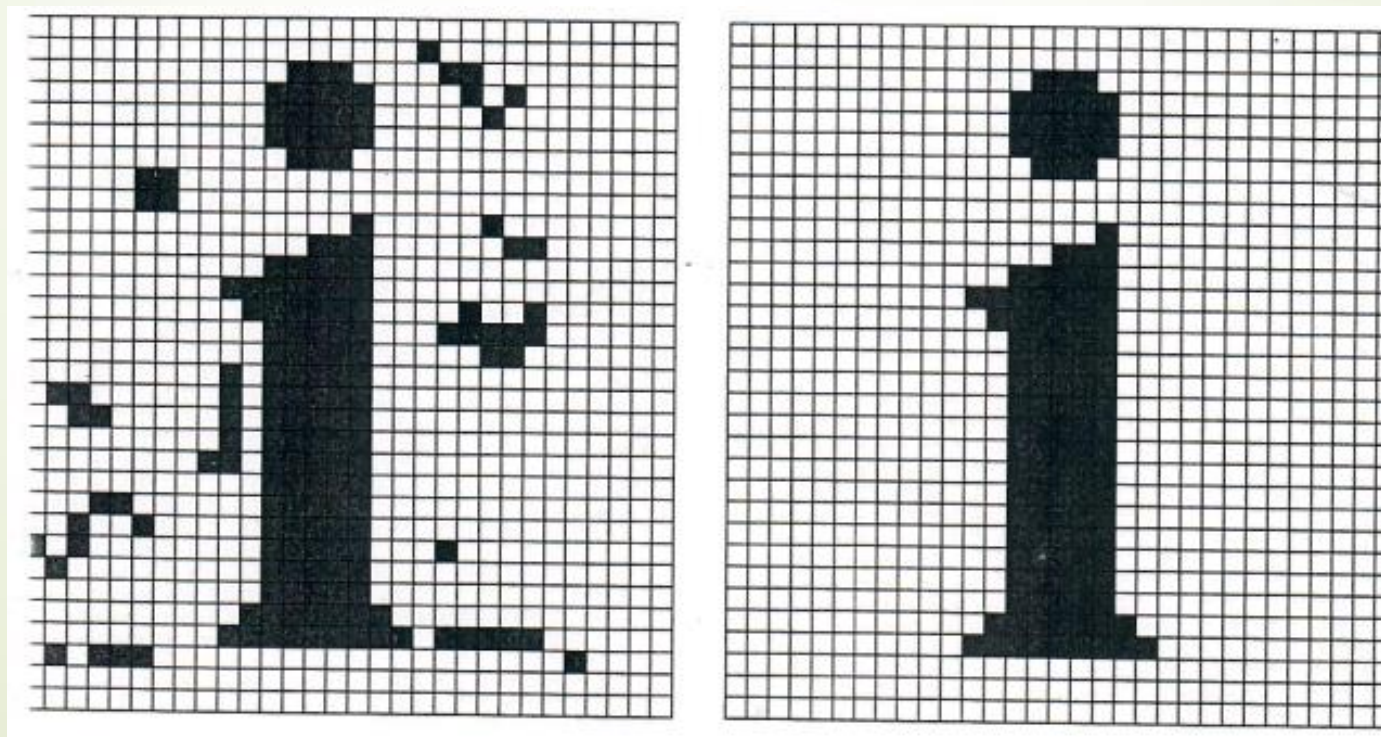
➤ רגיש לרעש!

שימוש במסנן מידה (size filter)

דוגמא:

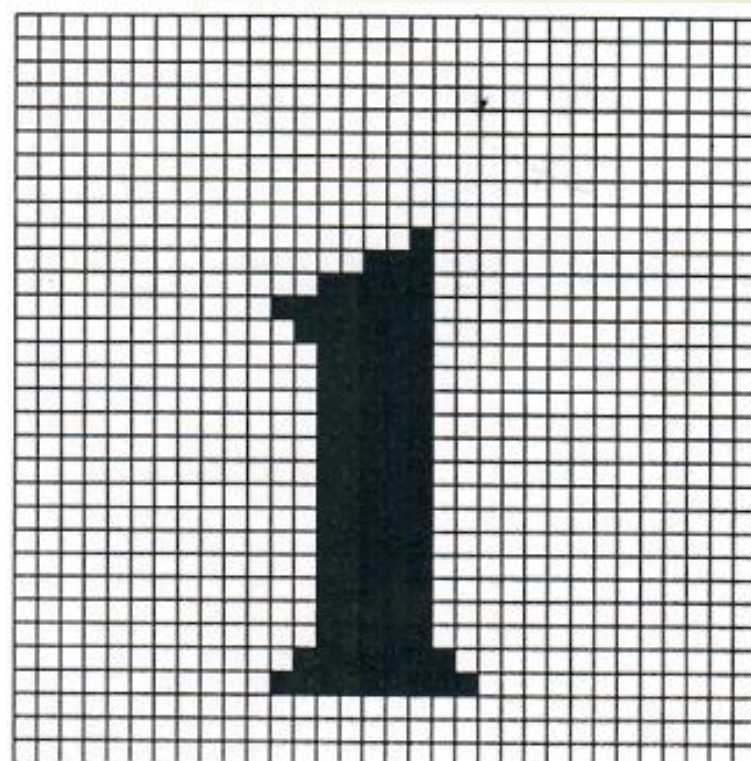
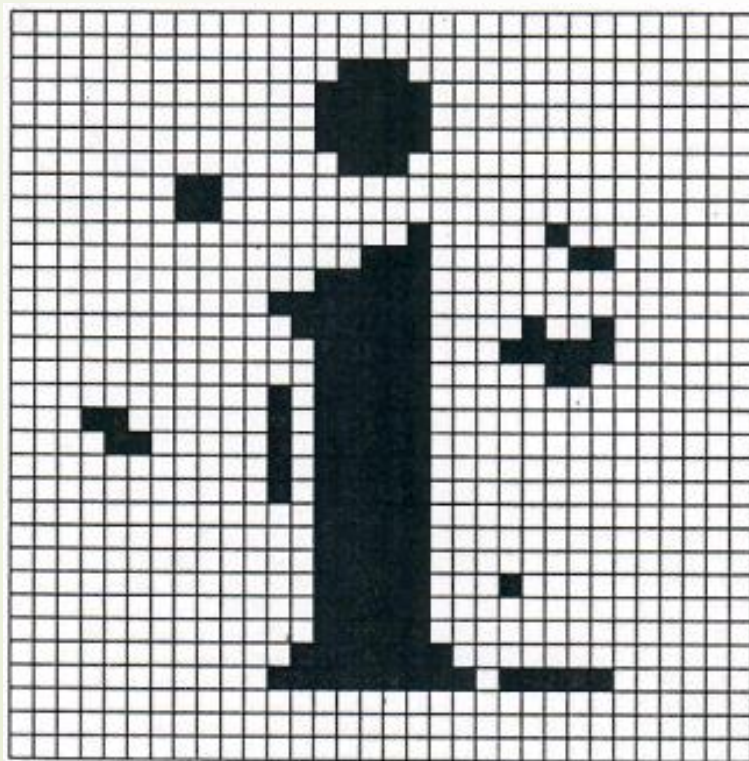
עקרונית, חשוב לבחור T גדול מספיק על מנת לנקות את הרעש, אולם אין לקחת T גבוה מידי שעלול "למחוק" חלק מהתמונה.

עבור $T = 10$, יסוננו רכיבים קשירים שגודלם פחות מ-10 פיקסלים.



מסנן מידה, דוגמא (המשך)

דוגמא: עבור סף גדול מידי ($T = 25$), מתבצע סינון לא רצוי (מתבטא בקבלת הספרה "1" במקום האות "i")



מרחקים בין פיקסלים (distance measures)

➤ בהקשר של תמונה בינארית, ניתן להגדיר מרחק בכמה אופנים. בכל מקרה, חייבות להישמר התכונות של מטריקה.

➤ תכונות של מטריקה:

(i) $d(p, q) \geq 0$; $d(p, q) = 0 \Leftrightarrow p = q$

(ii) $d(p, q) = d(q, p)$

(iii) $d(p, r) \leq d(p, q) + d(q, r)$

סוגי מרחקים בין פיקסלי תמונה

➤ מרחק אוקלידי (Euclidean) או מרחק קו אווירי.

$$d([i_1, j_1], [i_2, j_2]) = \sqrt{(i_1 - i_2)^2 + (j_1 - j_2)^2}$$

לדוגמא:

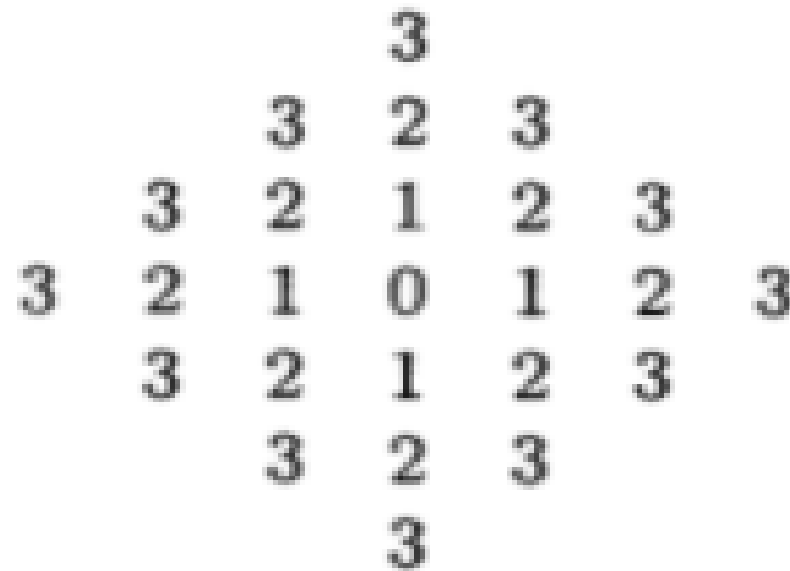
			3			
	$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$	$\sqrt{8}$	
	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{5}$	
3	2	1	0	1	2	3
	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{5}$	
	$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$	$\sqrt{8}$	
			3			

סוגי מרחקים בין פיקסלים (המשך)

- מרחק מנהטן (city block): אינו מרחק קו אווירי, אלא כמו המרחק על-פני רחובות ושדרות במנהטן, קרי ההעתק האופקי + ההעתק האנכי.

$$d = |i_1 - i_2| + |j_1 - j_2|$$

דוגמא



סוגי מרחקים בין פיקסלים (המשך)

➤ מרחק שח (chessboard): המקסימום בין ההעתק האופקי וההעתק האנכי. כמו מסעי מלך על לוח השח; יכול לנוע לכל משבצת סמוכה (גם אלכסונית), כך שהמרחק שעובר הוא הגדול מבין העתק האופקי וההעתק האנכי.

$$d = \max(|i_1 - i_2|, |j_1 - j_2|)$$

דוגמא

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

טרנספורמציות מרחקים (distance transformations)

➤ המרחק בין פיקסלי אובייקט (או מרכיב של אובייקט) לרקע יכול לשמש במספר אפליקציות (למשל OCR).

➤ בהינתן סט S , נרצה לחשב עבור כל $p \in S$ את המרחק של p לרקע \bar{S} .

➤ חישוב המרחקים המתאימים ניתן להיעשות באופן איטרטיבי, תוך הסתמכות על המשוואות הבאות [RK '82]:

$$f^\circ[i, j] = f[i, j]$$
$$f^m[i, j] = f^\circ[i, j] + \min(f^{m-1}[u, v])$$

באשר m מס' האיטרציה עבור כל הפיקסלים $[u, v]$ כך ש- $d([u, v], [i, j]) = 1$ (הכוונה רק לשכני-4 של $[i, j]$).

טרנספורמצית מרחקים (דוגמא)

$$f^0[i, j] = f[i, j]$$

$$f^m[i, j] = f^0[i, j] + \min(f^{m-1}[u, v])$$

1	1	1	1	1	1		1	1	1	1	1	1		1	1	1	1	1	1
1	1	1	1	1	1		1	2	2	2	2	1		1	2	2	2	2	1
1	1	1	1	1	1	→	1	2	2	2	2	1	→	1	2	3	3	2	1
1	1	1	1	1	1		1	2	2	2	2	1		1	2	2	2	2	1
1	1	1	1	1	1		1	1	1	1	1	1		1	1	1	1	1	1

Medial Axis Transform (MAT)

➤ נועד לייצוג קומפקטי של אובייקטים. כמו-כן, מאפשר מענה לשאלות בעלות עניין כגון האם פיקסל נתון נמצא באזור כלשהו.

➤ הגדרה: סט הפיקסלים ב- S שמרחקיהם מ- \bar{S} מהווים מקסימלי לוקאלי הקרוי skeleton, או MAT (ה-MAT יצוין ע"י S^*):

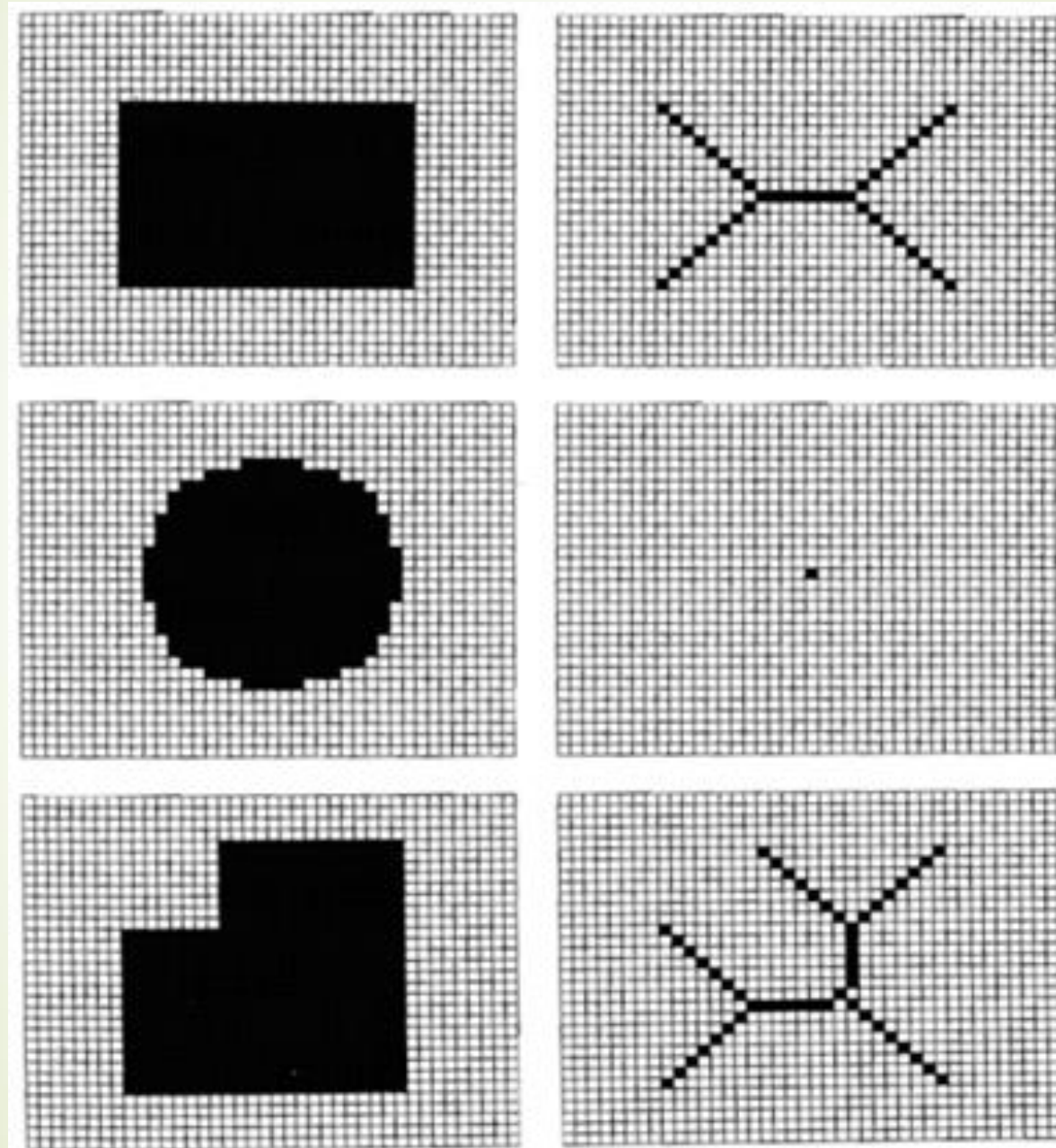
$$d([i, j], \bar{S}) \geq d([u, v], \bar{S})$$

לכל הפיקסלים $[u, v]$ שבסביבת $[i, j]$.

➤ מתאים במיוחד לצורות מוארכות.

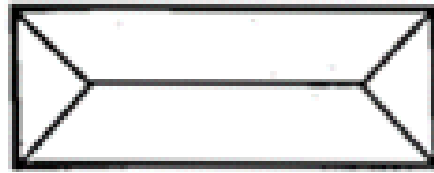
➤ S^* מהווה ייצוג קומפקטי של S . (S ניתן לשחזור באמצעות S^* והמרחקים ל- \bar{S} של כל פיקסל ב- S^*).

MAT Examples (Nonrepresentative)



MAT Examples (Cont'd)

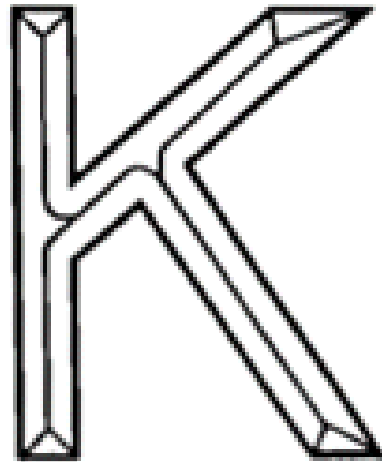
a



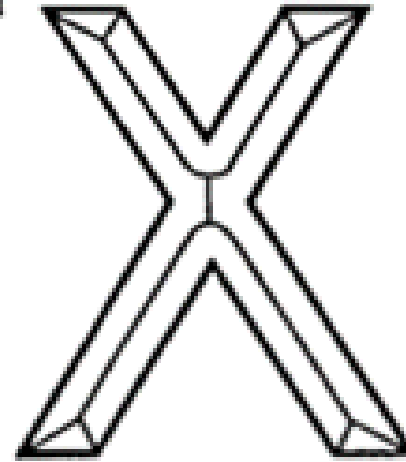
b



c



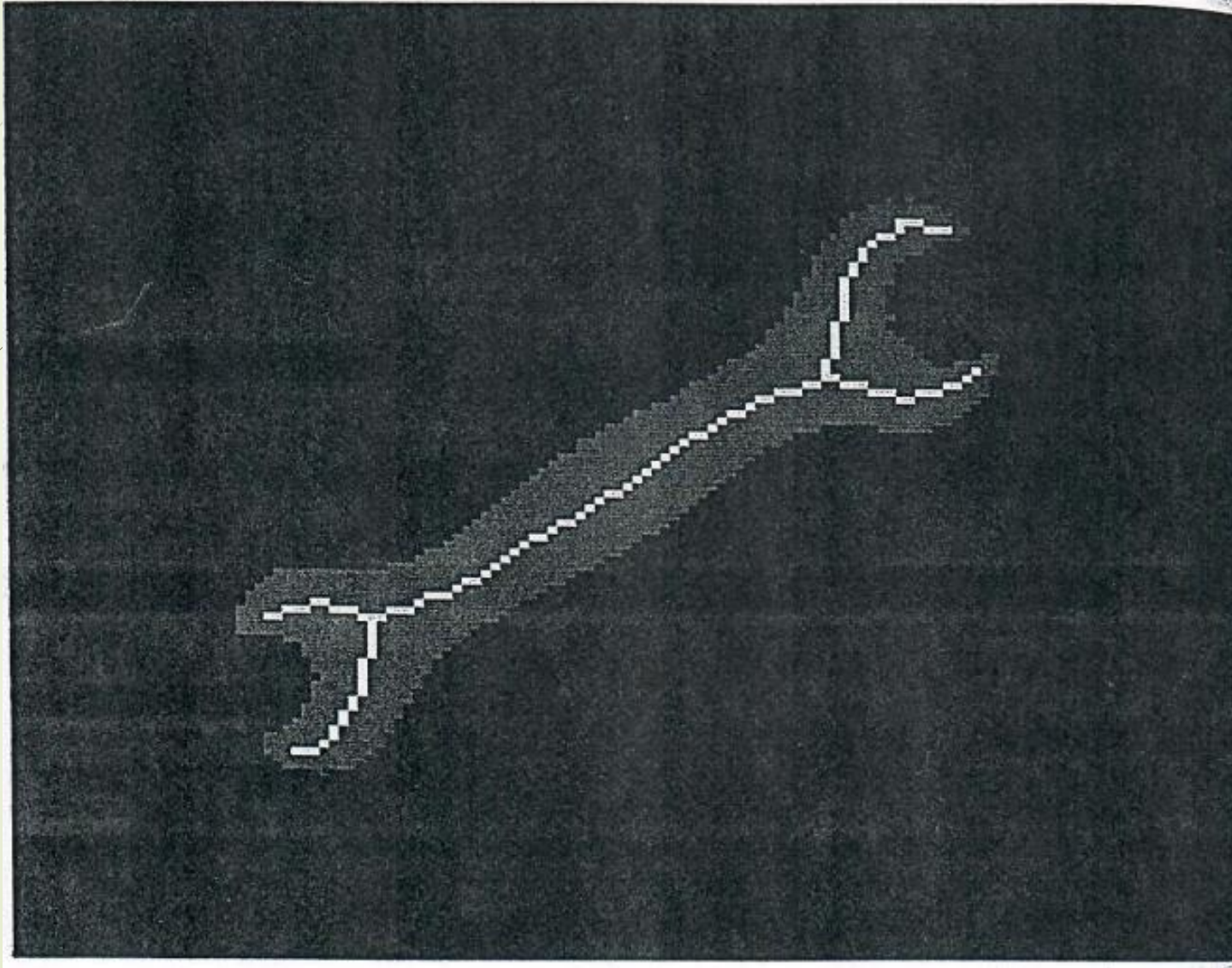
d



אופרטור דילול (thinning)

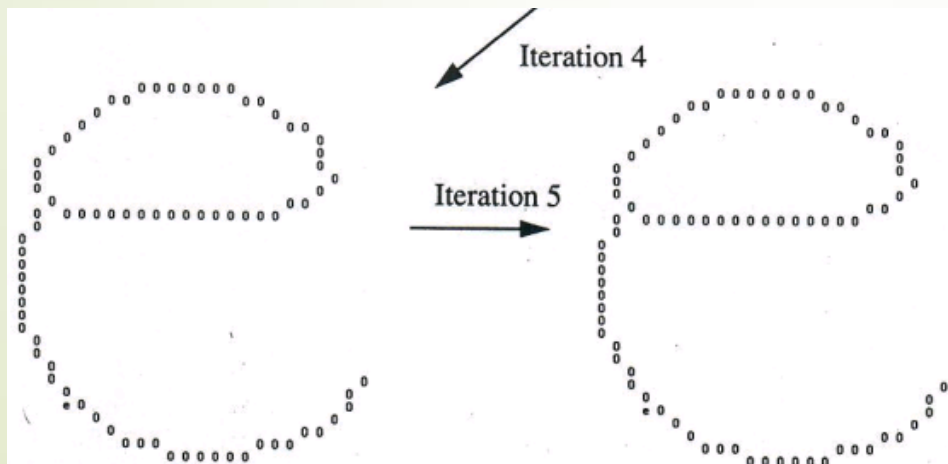
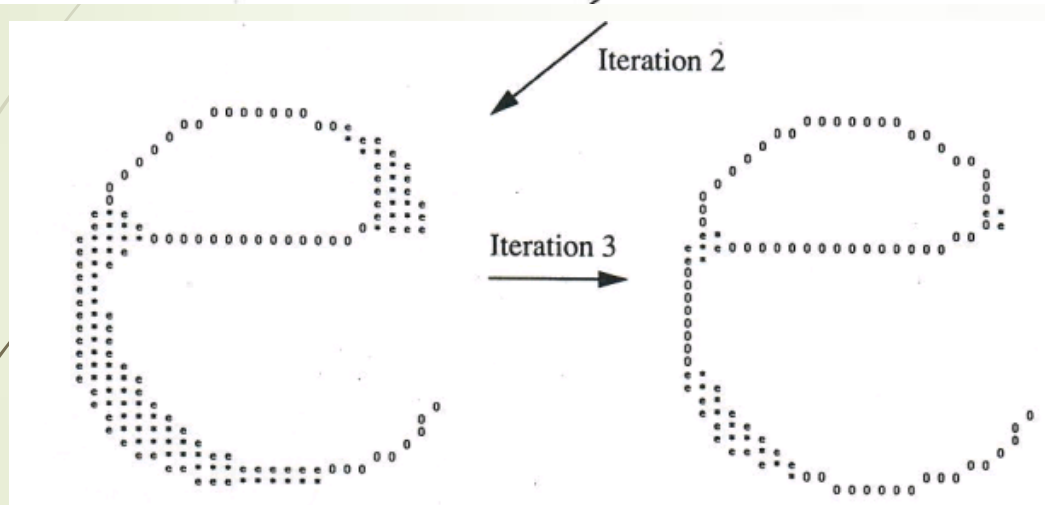
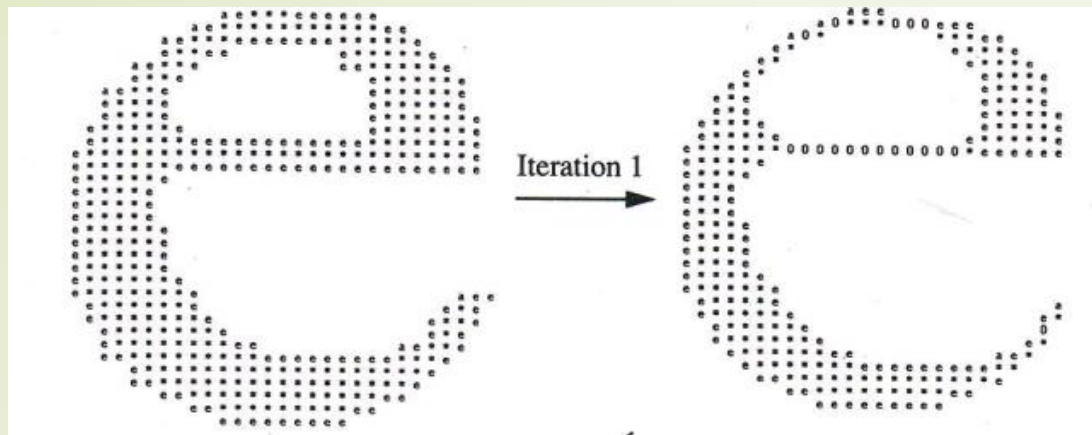
- אופרציה לקבלת קטעים קווים (lines) המהווים אומדן לקווים מרכזיים (MAT) של קבוצות פיקסלים בתמונה בינארית.
 - תוצאת האופרטור: סט דחוס (קומפקטי) של פיקסלים קשירים כך שנקודות קצה (endpoints) נשמרות.
- ** ראה [RK'82] עמ' 232-237 (מצ"ב שקפים אחרונים).

Thinning Operator (Example)



Specifications of Thinning Operator

- Connected image regions must thin to connected line structures
- The thinned result should be minimally 8-connected, i.e., the resulting line should always contain a minimal number of pixels maintaining 8-connectedness.
- Approximate endlines should be maintained
- The thinning results should approximate the medial lines (MAT)
- Extraneous short branches (caused by thinning) should be minimized, i.e., noise reduction



Example of 5-iteration thinning of the letter “e”; at each iteration a layer of the boundary is peeled off. The procedure terminates on iteration 5, as there was no change.

a. Thinning

def Our thinning algorithm is a specialized shrinking process which deletes from S , at each iteration, border points whose removal does not locally disconnect their neighborhoods; it can be shown [54] that this guarantees that the connectedness properties of S do not change, even if all such points are deleted simultaneously. To prevent an already thin arc from shrinking at its ends, we further stipulate that points having only one neighbor in S are not deleted.

OK

Unfortunately, if we delete all such border points from S , and S is only two points thick, e.g.,

```

1 1 1 1 1
1 1 1 1 1

```

✓ then S will vanish completely. We could avoid this by using an algorithm that examines more than just the immediate neighbors of a point, but such an algorithm would have to be quite complicated. Instead, we delete only the border points that lie on a given side of S , i.e. that have a specific neighbor (north, east, south, or west) in S , at a given iteration. To insure that the skeleton is as close to the "middle" of S as possible, we use opposite sides alternately, e.g., north, south, east, west (It is possible to devise algorithms that remove border points from two adjacent sides at once, e.g., north and east, then west and south; but this approach is somewhat more complicated and will not be described here in detail. Another possibility is to check the neighbors of a point on two sides to determine whether they too will be deleted, and if so, not to delete the given point.)

— In order to state the algorithm more precisely, we must give the exact conditions under which a border point can be removed. The border point P of S is called simple if the set of 8-neighbors of P that lie in S has exactly one 8-component that is adjacent to P . This last clause means that if we are using 4-connectedness for S , we care only about components that are 4-adjacent to P . If we are using 8-connectedness, the last clause can be omitted. For example, P is 4-simple if its neighborhood is

```

0 1 0
0 P 0
1 0 0

```

1-Connected

since in this case only one 4-component of 1's is 4-adjacent to P ; but P is not 4-simple if its neighborhood is

```

0 1 1
0 P 0
0 1 0

```

or

```

0 1 0
0 P 1
0 0 0

```

8-Connected

On the other hand, P is 8-simple in the third case, but not in the first two cases. \checkmark

\checkmark It is easily seen that deleting a simple point from S does not change the connectedness properties of either S or \bar{S} ; $S - \{P\}$ has the same components as S , except that one of them now lacks the point P , and $\bar{S} \cup \{P\}$ has the same components as \bar{S} , except that P is now in one of them. Note that an isolated point (having no neighbor in S) is not simple, and that an end point (having exactly one neighbor in S) is automatically simple. \checkmark } !

\checkmark Our thinning algorithm can now be stated as follows: Delete all border points from a given side of S , provided they are simple and not end points. Do this successively from the north, south, east, west, north, ... sides of S until no further change takes place. A simple example of the operation of this algorithm is shown in Fig. 14. \checkmark The algorithm

The deletion of border points from a given side of S should be done "in parallel," i.e. the conditions for deletion of a point should be checked before any other points are deleted. (Suppose we did not do this, but simply performed the deletion row by row. When we deleted north border points, we would strip away layer after layer from the top of S , and the resulting skeleton would not be symmetric; e.g., if S were an upright rectangle, nothing would be left but its bottom row after the first operation.) Thus each iteration of the algorithm can be done as a simple parallel operation on a cellular array computer.

The algorithm can be implemented on a conventional computer, using one scan of the picture for each iteration. On each row, points are marked for deletion, but are not deleted until the points on the following row have been marked. We can avoid repeated scanning of the entire picture by operating on the rows in sequence 1; 2, 1; 3, 2, 1; ... as in Section 11.2.1c. After k steps, where $2k + 1$ is the maximum width of S , no more thinning is needed on the first row, so it can be dropped; thus only about k rows need to be available at a time. \checkmark

Exercise 12 [3]. Prove that a north border point (=having north neighbor 0) is 8-simple iff

$$w\bar{s}e + \bar{w}(nw)\bar{n} + \bar{n}(ne)\bar{e} + \bar{e}(se)\bar{s} + \bar{s}(sw)\bar{w} = 0$$

where $n, e, s, w, (nw), (ne), (se), (sw)$ denote the north, south, east, west, northwest, northeast, southeast, and southwest neighbors of the point, respectively, and the overbars denote negation ($\bar{0} = 1, \bar{1} = 0$). Can you formulate an analogous Boolean condition for 4-simplicity? ■

\checkmark b. *Alternative thinning schemes*

Simplified approaches to thinning can sometimes be used. For example, if S has everywhere essentially constant thickness (see Section 11.3.2d), say

$2k + 1$, it can be thinned (at least roughly) by shrinking it k times. Note, however, that the resulting skeleton may occasionally be thick or broken. The thinning processes described in (a) will handle S 's of variable width. Another method [32] that can be used if S is an arc of constant thickness is to initiate two border-following processes at one end of S that traverse the opposite sides of S . (Analogously, if S is a closed curve of constant thickness, we initiate two border-following processes at points just across the width of S from each other, which traverse the two borders of S .) If the distance between the border followers gets significantly larger than the width, we stop one of them until the other one catches up; thus they always remain approximately

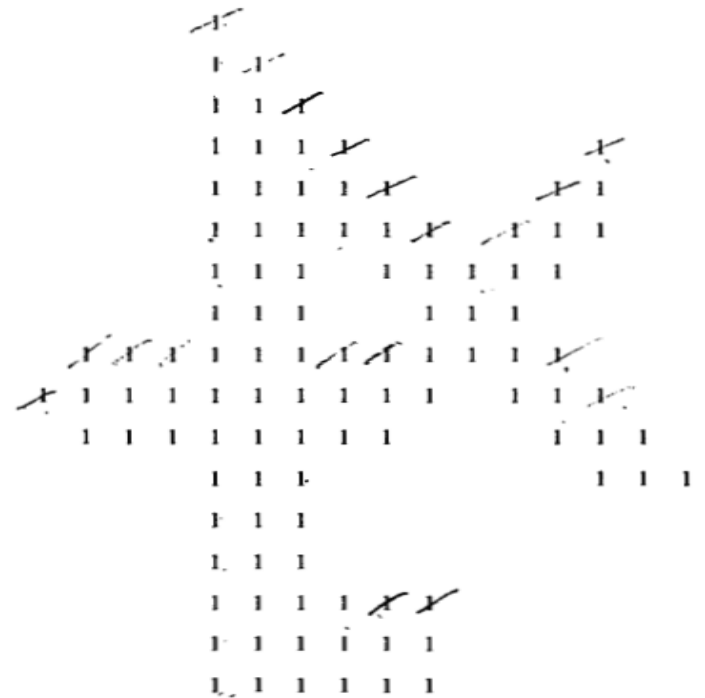
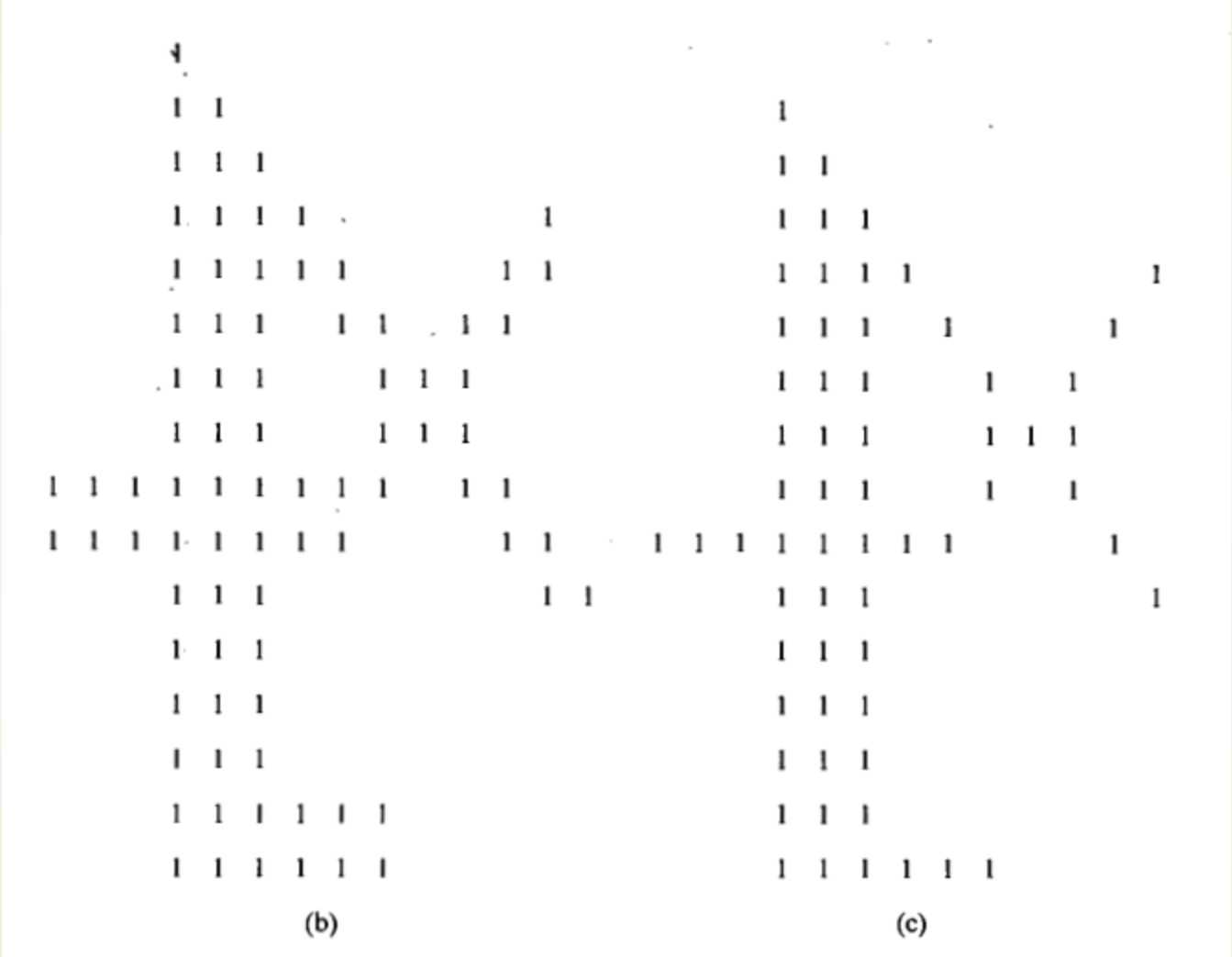
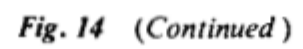


Fig. 14 Thinning. (a) Original S ; (b)–(e) results of thinning successively from north, south, east, and west, treating S as 8-connected. At the next north step, the uppermost point of (e) will be deleted; at the next south step, the leftmost of the lowermost points; and at the next west step, the leftmost of the uppermost points. There will be no other changes. Note how the angle at the upper left shrinks down to the same height as the diagonal line at the upper right. (b')–(e') Analogous results treating S as 4-connected, and defining end points as having only one 4-neighbor in S . One more point will be removed at the next north step; there will be no other changes.







```

      1
      1
      1 1
      1 1 1
      1 1 1 1 1
      1 1      1 1      1 1
      1 1      1 1 1
      1 1      1 1
1 1 1 1 1 1 1 1 1 1 1 1
      1 1      1 1
      1 1      1 1 1
      1 1
      1 1
      1 1
      1 1 1 1 1 1
      6'

```

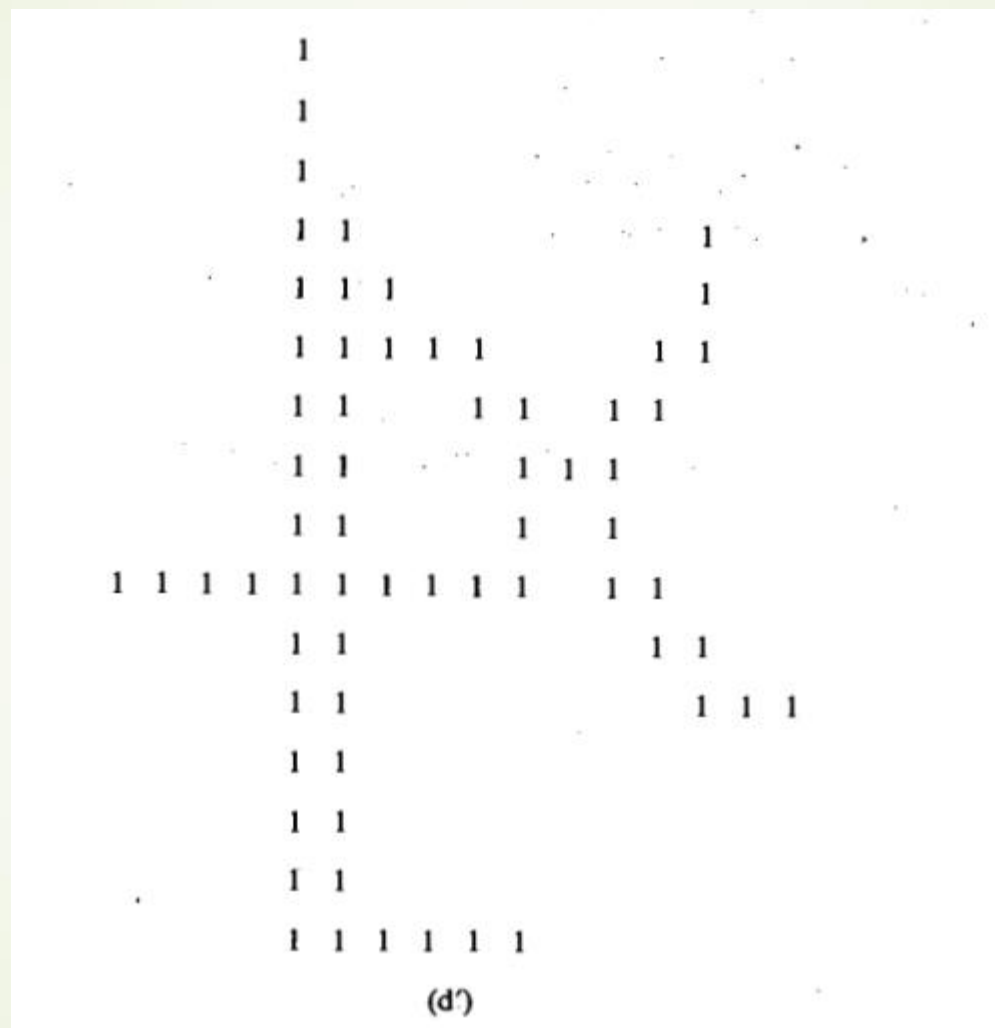
```


      1
      1
      1
      1 1              1
        1 1          1
        1 1 1 1      1 1
        1      1 1 1 1
        1      1 1 1
        1      1 1
1 1 1 1 1 1 1 1 1 1 1 1
      1          1 1
      1          1 1 1
      1
      1
      1
      1 1 1 1 1

```

(c) c'

Fig. 14 (Continued)





```

      1
      1
      1
    1 1                                1
        1 1                                1
          1 1 1 1                        1 1
            1      1 1      1 1
            1      1 1 1
            1      1 1
            1      1
1 1 1 1 1 1 1 1 1 1 1 1
          1      1 1
          1      1 1 1
          1
          1
          1
        1 1 1 1 1
  
```

(e')

Fig. 14 (Continued)