

## עיבוד תמונות בינאריות (binary images)

תמונה בינארית היא תמונה "שחור-לבן" - כל פיקסל יכול להיות מיוצג על ידי 2 ערכים בלבד - 0 או 1. שחור באופן מוחלט או לבן באופן מוחלט. למה שבכלל נרצה דבר כזה? יש מספיק אפליקציות בהן לא חייבות להיות רמות אפור. למשל - טקסט. יותר יעיל ויותר נוח לעבוד עם תמונות בינאריות. קל מאוד - מבחינה טכנית - לעבור מתמונת רמות-אפור לתמונה בינארית.

### הנחת עבודה:

- כל פיקסלי ה-1 הלוגיים יהיו "שחורים" (יש אובייקט)
- כל פיקסלי ה-0 הלוגיים יהיו "לבנים" (אין אובייקט - רקע)  
נשים לב ש-0 ו-1 לוגיים הפוכים מ-0 ו-1 "הרגילים"

## הפעלת סף - Thresholding

הנחת עבודה יש אובייקטים עם רמות אפור טיפוסיות, ויש רקע עם רמות אפור אחרות, ורוצים להפריד בין האובייקט לרקע. נתון אזור בעל עניין<sup>1</sup>, ואנחנו רוצים להפריד בין פיקסלים בטווח מסויים לפיקסלים בטווח אחר. כשמציירים את ההיסטוגרמה אפשר לבחור איך לחלק את האינטרוולים.

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

כאשר:

$B[i, j]$  התמונה הבינארית

$f_T[i, j]$  טרנספורמציה  $T$  על התמונה המקורית  $f$

ו:

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

או - אם רוצים לבחור טווח מסויים ולא רק לחלק את ההיסטוגרמה ל-2:

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

או - הגדרה יותר כללית:

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

כאשר  $Z$  הוא סט של אינטרוולים. הפעולה היא קלה טכנית, אבל קביעת הספים אינה טריוויאלית.

יש לנו תמונה בינארית  $B[i, j]$  - מה אפשר לעשות איתה?

## תכונות גיאומטריות

- גודל (size) - שטח התמונה, מספר הפיקסלים השייכים לאובייקט (1 לוגי)

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

- קואורדינטית מיקום - מרכז המסה:

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

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

זה נקרא "מומנט מסדר ראשון". שטח הוא מומנט מסדר 0, ובאופן כללי, מומנטים מסדר  $p + q$ :

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

ככל שהצורה מורכבת יותר, אפשר לתאר אותה עם מאפיינים מהסוג הזה (עם ערכי  $p$  ו  $q$  יותר ויותר גבוהים). אנחנו רוצים לבנות "וקטור מאפיינים". אנחנו לא יכולים לזהות. אנחנו לא מסתכלים על כל פיקסל ופיקסל בתמונה כדי לזהות מה יש שם - יש איזה משהו אחר שמסתכלים עליו, וקטור מאפיינים שמייצג את התמונה באופן יותר קומפקטי (לא צריך "לסחוב" אחרינו המון פיקסלים). לכן יש עניין איזה מאפיינים לבחור. היום רשתות נייורוניות לומדות את וקטור המאפיינים בלי שחוקר אנושי יצטרך לשבור את הראש. המטרה היא לחלץ "תעודת זהות" של אובייקט נתון - רשימה של פרמטרים שהם מספיק טובים כדי לזהות את האובייקט database.

- אוריינטציה - התאמה של ישר לאוסף הפיקסלים. רוצים למזער את קריטריון השגיאה:

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

כאשר  $r_{i,j}$  הוא המרחק בין הישר בנקודה ספציפית ביחד ל  $[i, j]$

- קומפקטיות - היחס בין ההיקף ( $P$ ) בריבוע לשטח:

$$\frac{P^2}{A} \geq 4\pi$$

המצב הקומפקטי ביותר הוא  $4\pi$  - ההיקף בריבוע של מעגל חלקי השטח. כאשר הצורה שטוחה השטח שואף לאינסוף.

## ייצוג תמונה בינארית

אפשר לקדד על ידי אורכי רצפים:

$$[1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1 \ \dots] \implies (1,3)(7,2)\dots$$

אפשר גם לקודד אורכי רצפים של 1ים ושל 0ים, לסירוגין:

$$[1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1] \implies 3,3,2,\dots$$

## אלגוריתמים בינאריים

רוצים לקבץ פיקסלים לאובייקטים. בשביל זה צריך למצוא:

שכנות: יש שתי הגדרות לשכנות: מטיפוס 4:  $\begin{bmatrix} n & p & n \\ n & p & n \\ n & n & n \end{bmatrix}$ . מטיפוס 8:  $\begin{bmatrix} n & n & n \\ n & p & n \\ n & n & n \end{bmatrix}$ . כל שכן מטיפוס 4 הוא גם שכן מטיפוס 8 - אבל לא להפך!

מסלול: אוסף של פיקסלים עם קשרי שכנויות אחד לשני (לא כולם לכולם, אבל כל פיקסל שכן לפיקסל אחר במסלול)

קשירות: פיקסל  $p \in S$  קשור ל- $q \in S$  אם קיים מסלול  $mp$  ל- $q$  המכיל אך ורק פיקסלי  $S$ .

מרכיב קשיר: סט של פיקסלים בו כל פיקסל קשור לכל שאר הפיקסלים. המרכיבים הקשירים אמור לייצג את האובייקטים.

רקע: סט כל המרכיבים הקשירים ב- $\overline{S^3}$  הכוללים פיקסלי קצוות (מסגרת) של התמונה.

חורים: שאר המרכיבים הקשירים של  $\overline{S}$  שאינם רקע.

פיקסלי גבול: הגבול של קבוצת פיקסלים  $S$  כולל את הפיקסלים ב- $S$  שיש להם שכני-4 ב- $\overline{S}$ . נסמן את הגבול ב- $S'$ .

נשים  $\heartsuit$ : אם ההגדרה של הגבול הייתה שכני-8, היה יכול להיות לנו גבול עבה: למשל ב- $\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

הגבול היה  $\begin{bmatrix} b & b \\ b & b & b & b \end{bmatrix}$ .

## אלגוריתם למציאת פיקסלי הגבול/קונטור

- סורקים את התמונה משמאל לימין, מלמעלה למטה, עד שנתקלים בפיקסל  $s \in S$  הראשון.
- מסמנים את פיקסל השפה הנוכחי ב- $c$ . נציב  $c = s$ . נסמן ב- $b$  את שכן ה-4 (ממערב) של  $s$ . נשים לב ש- $b \in \overline{S}$ .
- יהיו  $n_1, n_2, \dots, n_8$  שמונת שכני-8 של  $c$  (עם כיוון השעון) כאשר  $n_1 = b$ . מוצאים את ה- $i$  הראשון עבורו  $n_i \in S$ .

4. נציב  $b = n_{i-1}, c = n_i$ .

5. חוזרים על צעדים 3 ו-4 עד ש  $c = s$ .

נשים ♡: האלגוריתם ימצא את הגבול של האובייקט הראשון - אם רוצים למצוא אובייקטים אחרים ו/או חורים צריך להמשיך את החיפוש.

### קוד השרשרת

לכל כיוון ניתן מספר - מתחילים במזרח=0 וממשיכים נגד כיוון השעון:

שכנות-4:  $\begin{bmatrix} 3 & 2 & 1 \\ 4 & & 0 \\ 5 & 6 & 7 \end{bmatrix}$

שכנות-8:  $\begin{bmatrix} & 1 & \\ 2 & & 0 \\ & 3 & \end{bmatrix}$

אפשר לתאר גבול באמצעות כיוונים. מתחילים מהפינה השמאלית העליונה, ורושמים את הכיוון לפיקסל הגבול הבא. למשל בייצוג-8:

$$\begin{bmatrix} b & b & b \\ b & & b \\ b & b & \end{bmatrix} \implies 0065422$$

### אלגוריתם למציאת מרכיבים קשירים - connected component labeling

המטרה: מציאת כל המרכיבים הקשירים בתמונה, והקניית תווית בלעדית לכל הפיקסלים השייכים לאותו מרכיב קשיר.

1. סורקים את התמונה משמאל-לימין, מלמעלה-למטה.

2. אם הפיקסל הוא 1 לוגי, אז:

( ) אם רק לשכן השמאלי או לשכן שמעל יש תווית - מעתיקים אותה.

( ) אם לשני השכנים הללו יש את אותה תווית - מעתיקים אותה.

( ) אם יש להם תוויות שונות - מעתיקים את זו של השכן מעל ומסמנים אותן כשקולות בטבלת השקילות.

( ) אם אין לאף אחד מהם תווית, נותנים תווית חדשה לפיקסל ומכניסים אותה לטבלת השקילות.

3. חוזרים על 2 עד שעוברים על כל הפיקסלים.

4. מוצאים את התווית בעלת הערך הקטן ביותר לכל קבוצת שקילות.

5. סורקים שוב, ומחליפים כל תווית בתווית השקולה שמצאנו בשלב 4.

## מאפיין טופולוגי נוסף: מספר אויילר Euler Number

מספר אויילר הוא פרמטר המשמש לאפיון צורה:

$$E \triangleq C - H$$

כאשר:

$C$  מספר מרכיבי הקשירות

$H$  מספר החורים

מספר אויילר אינו מאפיין חד-חד-ערכי של התמונה! תמונת עם אותו מספר אויילר יכולות להיות מאוד שונות. אז למה זה טוב? ב-OCR למשל הוא עוזר בסיווג אותיות.

## מסנן מידה size filter

רכיבי קשירות קטנים הם מבחינתנו רעש, ונרצה לסנן אותם - למחוק רכיבים שגודלם פחות מ- $T$  פיקסלים. חשוב לבחור  $T$  מספיק גדול כדי לנקות את הרעש, אבל מספיק קטן כדי לא למחוק חלקים מהתמונה עצמה. כמו בהפעלת סף - זה לא טריוויאלי לבחור את  $T$ . אפשר לעשות את זה עם ניסוי וטעייה, עם רשת ניורונית או עם כל מיני יוריסטיקות.

## מרחקים בין פיקסלים distance measures

ניתן להגדיר מרחק בכמה אופנים, אבל חייבות להישמר תכונות המטריקה:

1. אי-שליליות:  $d(p, q) \geq 0$ ;  $d(p, q) = 0 \iff p = q$

2. סימטריה:  $d(p, q) = d(q, p)$

3. אי שוויון המשולש:  $d(p, q) + d(q, r) \leq d(p, r)$

## סוגי מרחקים בין פיקסלים

- מרחק אוקלידי:

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

- מרחק מנהטן:

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

- מרחק שח (chessboard) - מספר צעדי מלך על לוח שח:

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