

## תמונות בינאריות

תמונות בהן יש או 0 או 1 - כלומר או רקע או אובייקט. בד"כ נתייחס ל1 בתור אובייקט ו0 בתור רקע<sup>1</sup>. נרצה להשתמש בתמונות בינאריות כדי למצוא מזהים של האובייקט, למצוא רכיבי קשירות (עם שכנות 8, שכנות 4).

### מציאת רכיבים קשירות - tag connected components

עוברים על התמונה משמאל לימין, מלמעלה למטה. כאשר מגיעים לפיקסל שיש בו אובייקט, ומסתכלים על שכני הארבע שכבר עברנו עליהם - כלומר זה שמעליו וזה שמשמאלו - אם יש להם תוויות, לוקחים את התוויות שלהם. אם אין לו אף שכן עם תווית - מקצים תווית חדשה. אם יש שני שכנים עם אותה תווית אז אין בעיה, אבל אם יש לשניהם תוויות שונות צריך לבחור אחד מהם - נקבע למשל שבחרים את השמאלי - ומסמנים בטבלת השקילויות ששתי התוויות האלו הן למעשה אותה תווית. בסוף צריך להשתמש בטבלה כדי לאחד תוויות. איך עושים את זה? בונים מטריצת שכנויות (שאחר כך הופכים אותה למטריצת שקילויות):

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

את המטריצה מכפילים בעצמה:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 2 & 2 & 0 & 0 \\ 2 & 2 & 1 & 0 & 0 \\ 2 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 2 & 2 \end{bmatrix}$$

הסכום בכל תא הוא:

$$m^2(i, j) = \sum_k m_{ik} \cdot m_{kj}$$

כלומר,  $m^2(i, j) > 0 \iff$  יש קודקוד שהוא שכן (או שהוא בעצמו) גם של  $i$  וגם של  $j$ , ובעצם  $m^2(i, j)$  הוא מספר הדרכים להגיע מ  $i$  ל  $j$ . אחרי זה עושים  $\min(m, 1)$ :

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

אבל זה נותן לנו רק מסלולי 2 - אז צריך לעשות  $m^N$  (שזה  $\log N$  מכפלות) המטריצה נותנת לנו רכיבים קשירות, ומהם יוצרים וקטור המרה. מאתחלים אותו בשורה הראשונה:

$$[1 \ 1 \ 1 \ 0 \ 0]$$

<sup>1</sup>שבתמונות מודפסות זה הפוך כי לא רוצים להדפיס את הרקע בשחור.

רצים עד שמגיעים לעמודה שהיא לא 0, ואז הולכים בחזרה לשורה במטריצה, מכפילים במספר הבא, ומוסיפים:

$$\begin{bmatrix} 1 & 1 & 1 & 2 & 2 \end{bmatrix}$$

וככה ממשיכים עד שממלאים את הוקטור - ואז משתמשים בו בשביל לעדכן את התגיות.

## הוצאת שלד מתמונה

כשאנחנו מדברים על אובייקטים שונים, אנחנו רוצים למצוא מאפיינים שתופסים פחות מקום כדי להשוות מאפיינים או לקבל מהם מידע על התמונה.

לדוגמה, אם אנחנו רוצים לעשות זיהוי פנים, אי אפשר להשוות פיקסל-פיקסל - כי כל שינוי קטן בצילום משנה משמעותית את הפיקסלים (בהשוואה של אחד לאחד). לכן נרצה למצוא מאפיינים, למשל למצוא את העיניים, לחשב את המרחק בין העיניים, ולמצוא את היחס בין זה לרוחב של הראש. זהו מספר בודד שמאפשר לעשות חיפוש יותר יעיל ומהיר ולאנדקס. באופן דומה מוצאים עוד מאפיינים. השלד של התמונה הוא משהו מרכזי שמאפיין את הצורה. למשל השלד של מלבן הוא פס ישר כאשר בקצוות יש קווים לפינות.

אלגוריתם:

- שלב 1: מתחילים עם תמונה בינארית (כל התאים 0 או 1), ועוברים עליה באיטרציות.
  - באיטרציה  $i$  כל תא שהוא ושכני ה-4 שלו שווים ל-1 (באיטרציה הקודמת<sup>2</sup>) מקדמים ל- $i + 1$ .
  - תנאי עצירה: איטרציה בה לא קידמנו אף תא.
  - תוצאה של השלב: טבלת מרחקים מהרקע.
- שלב 2: סמן כל תא שערכו גדול או שווה לשכניו.

אי אפשר לשחזר מהשלד את התמונה המקורית. לדוגמה - השלד של עיגול הוא נקודה בודדת - מרכז המעגל ולעניינים ברדיוסים שונים יהיה אותו שלד (נקודה בודדת). אבל אם בשלב 2 נשמור את המספרים במקום סתם לסמן ב-1, נוכל לשחזר את התמונה המקורית באמצעות ציור מחדש של ה"מעגלים" סביב הנקודות.

## Directional Smoothing

במקום לעשות ממוצע רגיל, אפשר לעשות ממוצע מכוון - פילטרים בכיוונים שונים. למשל ב- $3 \times 3$ , אפשר לעשות פילטרים כמו:

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(כמובן, כולם חלקי 3)

עם כל פילטר נקבל תמונה אחרת, ואפשר לבחור את הפילטר בכל אזור כדי לקבל כמה שיותר קרוב לערך המקורי.

בסופו של דבר יש כאן תהליך בחירה של קונבולוציות, ולא חובה שכולם יהיו בצורה הזאת - למשל יש אזורים בהם פילטר מלא מתאים יותר.

### בעייה - overfitting

אם נספק יותר מדי פילטרים נקבל תמונה שיותר מדי קרובה לתמונה המקורית - כולל הרעש! לכן צריך להגביל את כמות הפילטרים.

<sup>2</sup>לחלופין, אפשר לבדוק שהוא והשכנים גדולים או שווים ל-1.