6. Algoritmul de urmărire a conturului

6.1. Obiective:

Obiectivele acestui laborator sunt următoarele:

- extragerea conturului obiectelor folosind algoritmul de urmărire a conturului;
- reprezentarea eficientă a conturului extras folosind coduri înlănțuite;
- exploatarea avantajelor utilizării codurilor înlănțuite în reprezentarea contururilor obiectelor (reconstrucția conturului, potrivire, unire, etc.).

6.2. Fundamente teoretice

6.2.1. Algoritmul de urmărire a conturului

Algoritmul de urmărire a conturului este folosit pentru extragerea conturului obiectelor dintr-o imagine. La aplicarea acestui algoritm presupunem că imaginea este binară sau că obiectele din imagine au fost etichetate în prealabil.

Paşii algoritmului:

1. Se scanează imaginea din colțul stânga sus până când se găseşte un pixel care aparține unei regiuni; acest pixel P_0 reprezintă pixelul de start al conturului regiunii. Se definește o variabilă *dir* în care se reține direcția mutării anterioare de-a lungul conturului de la elementul anterior spre elementul curent. Se inițializează:
   (a) *dir* = 0 dacă conturul este detectat folosind vecinătate de 4 (Fig. 6.1a)
   (b) *dir* = 7 dacă conturul este detectat folosind vecinătate de 8 (Fig. 6.1b)
2. Se parcurge vecinătatea de 3x3 a pixelului curent în sens invers acelor de ceasornic, începând cu pixelul corespunzător poziției:
   (a) (*dir* + 3) *mod* 4 (Fig. 6.1c)
   (b) (*dir* + 7) *mod* 8 dacă *dir* este par (Fig. 6.1d)
   (*dir* + 6) *mod* 8 dacă *dir* este impar (Fig. 6.1e)
   Primul pixel găsit care are aceeaşi valoare ca şi pixelul curent este noul element P_n al conturului. Se actualizează valoarea lui *dir*.
3. Dacă elementul curent P_n al conturului este egal cu al doilea element P_1 din contur şi dacă elementul anterior P_{n-1} este egal cu primul element P_0, atunci algoritmul se încheie. Altfel se repetă pasul (2).
4. Conturul detectat este reprezentat de pixelii $P_0 \ldots P_{n-2}$.

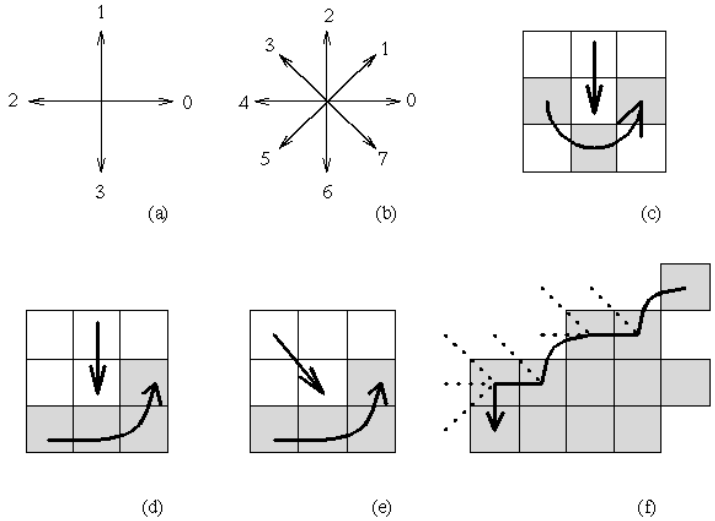

Fig. 6.1(a) Reprezentarea direcției, vecinătate de 4, (b) vecinătate de 8, (c) secvența de căutare în cazul vecinătății de 4 pixeli, (d),(e) secvența de căutare în cazul vecinătății de 8 pixeli, (f) urmărirea conturului pentru vecinătate de 8 (liniile întrerupte arată pixelii care au fost testați în timpul algoritmului de urmărire a conturului).

Observații:

- Algoritmul funcționează pentru toate regiunile care au suprafața mai mare de un pixel.
- Determinarea conturului unei regiuni formate dintr-un pixel este o problemă trivială.
- Algoritmul descris mai sus determină conturul exterior al regiunilor dar nu găseşte conturul găurilor din interiorul regiunilor.
- Pentru a determina şi conturul găurilor care apar într-un obiect, conturul trebuie urmărit începând cu fiecare regiune sau element din conturul unei găuri dacă acest element nu face parte dintr-un contur parcurs deja.
- Dacă obiectele au lățimea egală cu un pixel trebuie adăugate condiții suplimentare.

6.2.2. Extragerea codurilor înlănțuite

Codul înlănțuit reprezintă o modalitate eficientă de reprezentare a conturului unui obiect dintr-o imagine alb negru. Codul înlănțuit încorporează informații despre lungimea conturului obiectului, despre aria sa şi despre momente. Codurile înlănțuite sunt folosite la calcularea unor parametri pentru diferite tipuri de curbe. Codurile înlănțuite sunt reversibile, adică conturul unui obiect poate fi reconstruit având la dispoziție codul înlănțuit.

Ideea de bază la reprezentarea codului înlănțuit este că fiecare pixel de pe conturul unui obiect are un vecin adiacent care face parte din contur şi direcția de trecere de la un pixel dat de pe contur la acest vecin poate fi specificată printr-un număr unic ce ia valori între 0 şi 7 (vecinătate de 8). Codurile înlănțuite pot fi definite şi folosind o vecinătate de 4. Un exemplu este dat în Fig. 6.4.

În explicațiile care urmează vom folosi doar vecinătatea de 8 pixeli. Fiind dat un pixel, se consideră cei 8 vecini ai săi. Fiecăruia dintre ei i se poate asocia un număr de la 0 la 7 care reprezintă una din cele 8 direcții posibile de trecere de la pixelul curent la unul din vecini (vezi Fig. 6.2). Această orientare se păstrează pentru toată imaginea.

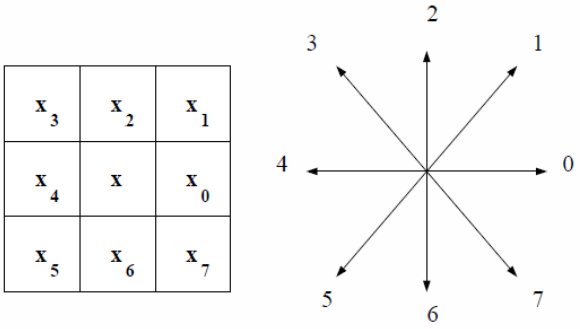

Fig. 6.2 Vecinătatea de 8 şi cele 8 direcții asociate

Codul înlănțuit al conturului unei imagini binare este o secvența de numere întregi $c=\{c_0, c_1, \dots, c_{n-1}\}$, cu c_i aparținând mulțimii $\{0,1, \dots ,7\}$ pentru $i=0, 1, \dots, n-1$. Numărul de elemente din mulțimea c reprezintă lungimea codului înlănțuit. Elementul c_0 este punctul *inițial* si c_{n-1} este punctul *final* al codului. Pornind de la un punct de referință, conturul unui obiect dintr-o imagine alb negru poate fi urmărit parcurgând codul înlănțuit.

Fig. 6.3 ilustrează procesul de urmărire a conturului unui avion folosind vectori de direcție. Informația din Fig. 6.3 este „nivelată" pentru a determina codul înlănțuit al conturului. Sa presupunem că alegem pixelul cel mai de sus din stânga imaginii din Fig. 6.3 ca punct de referință pentru codificarea conturului. Codul înlănțuit pentru conturul avionului este dat de secvența: 7, 6, 7, 7, 0, … , 1, 1, 1.

Fiind dat punctul de referință şi codul înlănțuit, conturul avionului poate fi reconstruit complet. Codul înlănțuit reprezintă o modalitate eficientă de a stoca informația de contur deoarece în reprezentarea lui sunt necesari 3 biți ($2^3 = 8$) pentru a determina oricare din cele 8 direcții de deplasare.

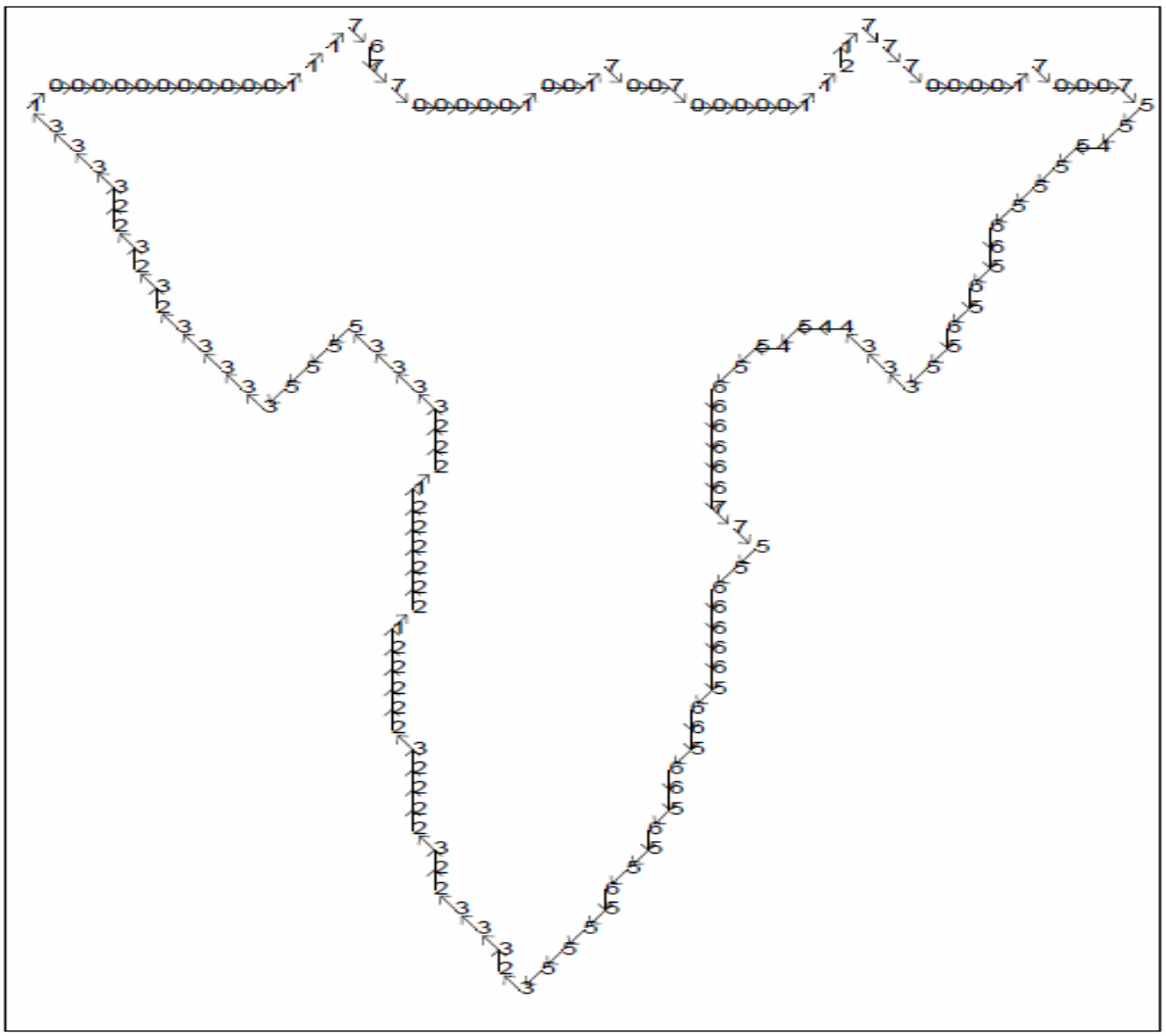

Fig. 6.3 Direcțiile codului înlănțuit cu numerele corespunzătoare

Codurile înlănțuite pot fi reprezentate independent de poziție prin ignorarea „punctului inițial” („punctul de început”). În cazul contururilor închise codurile înlănțuite pot fi normalizate în raport cu punctul de start prin alegerea acestuia astfel încât secvența rezultată la reprezentarea codului înlănțuit să formeze un număr întreg cu valoare absolută minimă.

„Derivata” codului înlănțuit este o reprezentare utilă deoarece este invariantă la rotația conturului. Derivata (o diferență *modulo* 4 sau 8) este o altă secvență de numere care indică direcția relativă a segmentelor codului înlănțuit; ele reprezintă numărul de rotații cu 90 de grade sau cu 45 de grade necesare pentru a obține direcția următorului segment din codul înlănțuit. O diferență *modulo* 4 sau 8 se numeşte derivata codului înlănțuit (vedeți Fig. 6.4).

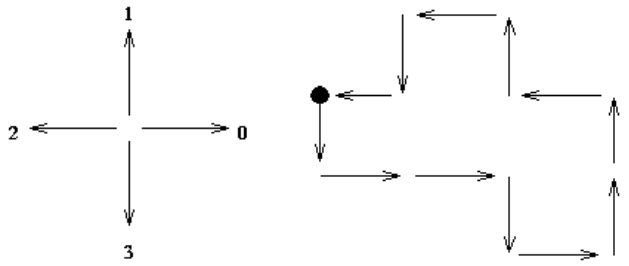

Fig. 6.4 Cod înlănțuit pentru vecinătate de 4 şi derivata codului înlănțuit.

Cod: 3, 0, 0, 3, 0, 1, 1, 2, 1, 2, 3, 2
Derivata: 1, 0, 3, 1, 1, 0, 1, 3, 1, 1, 3, 1

Proprietăți ale codului înlănțuit:

- Codurile înlănțuite descriu un obiect folosind o secvență de segmente de dimensiune unitate având orientări date (vecinătate de 4)
- Primul element al unei astfel de secvențe trebuie să conțină informații despre poziția primului pixel pentru ca regiunea să poată fi reconstruită.
- Codurile pare {0, 2, 4, 6} corespund direcțiilor verticale şi orizontale; codurile impare {1, 3, 5, 7} corespund direcțiilor diagonale.
- Fiecare cod poate fi considerat ca fiind direcția unghiulară, în multipli de 45 de grade, în care trebuie să fie parcurşi pixelii succesivi ai conturului.
- Coordonatele absolute ale primului pixel din contur (cel mai de sus, din stânga) împreună cu codul înlănțuit al conturului reprezintă informația completă despre conturul regiunii.
- O schimbare a două elemente consecutive din codul înlănțuit marchează o schimbare în direcția conturului. Punctul în care apare aceasta schimbare se numeşte *colț*.

6.3. Activități practice

Utilizând Diblook şi fişierele adiționale laboratorului:

1. Implementați algoritmul de urmărire a conturului şi desenați conturul obiectului dintr-o imagine având un singur obiect.
2. Folosind algoritmul de urmărire a conturului scrieți algoritmul care construieşte codul înlănțuit şi derivata codului înlănțuit al unui obiect. Calculați şi afişați ambele coduri (codul înlănțuit şi derivata) pentru o imagine având un singur obiect.
3. Implementați o funcție care reconstruieşte (afişează) conturul unui obiect peste o imagine, având ca şi date de intrare coordonatele punctului de start şi secvența de cod înlănțuit utilizând vecinătate de 8 (*reconstruct.txt*). Încărcați imaginea *gray_background.bmp* şi apelați funcția care reconstruieşte conturul. Trebuie să obțineți conturul cuvântului „EXCELLENT” (având literele unite între ele).
4. **Salvați-vă ceea ce ați lucrat. Utilizați aceeaşi aplicație în laboratoarele viitoare. La sfârşitul laboratorului de procesare a imaginilor va trebui să prezentați propria aplicație cu algoritmii implementați!!!**

<u>Informații adiționale:</u>

Imaginile de test care conțin un singur obiect au:

- 8 biți/pixel;
- indexul 0 pentru pixelii obiect (pixeli negri)
- altă valoare a indexului pentru pixelii de fundal (pixeli albi)

Fişierul *reconstruct.txt* este un fişier text care conține:

- pe prima linie coordonatele punctului de start (rând şi coloană) separate printr-un spațiu;
- pe a doua linie numărul de coduri înlănțuite;
- pe a treia linie codurile înlănțuite (secvența de direcții pentru vecinătate de 8 pixeli) separate printr-un spațiu.

Bibliografie

[1] Border Tracing – Digital Image Processing lectures, The University of Iowa, http://www.icaen.uiowa.edu/~dip/LECTURE/Segmentation2.html#tracing
[2] Contour Representations – Quantitative Imaging Group, Delft University http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip-Contour.html#Heading27
[3] G.X. Ritter, J.N. Wilson – Handbook of Computer Vision Algorithms in Image Algebra Second Edition – Chapter 10.4 Chain Code Extraction and Correlation, CRC Press, New York 2001
[4] Chain Codes – Digital Image Processing lectures, The University of Iowa http://www.icaen.uiowa.edu/~dip/LECTURE/Shape2.html#chaincodes
[5] Representation of Two-Dimensional Geometric Structures, http://homepages.inf.ed.ac.uk/rbf/BOOKS/BANDB/LIB/bandb8_12.pdf