4. Segmentarea imaginilor color (3): segmentarea bazata pe regiuni

Scop: implementarea unei unui algoritm de segmentare bazat pe regiuni de tip „region-growing" folosind modelul de culoare construit anterior (lucrarea 2).

4.1. Consideratii teoretice

O alta categorie de metode de segmentare a imaginilor este de a identifica regiuni *(componente conexe*) care satisfac anumite criterii de omogenitate, bazate pe trasaturi derivate din componentele spectrale. Aceste componente sunt definite in spatiul de culoare considerat.

Regiune:= setul maximal de pixeli pentru care este satisfacuta o conditie de uniformitate (predicat de omogenitate):
(a) Regiuni uniforme obtinute prin cresterea unui bloc/seed prin unirea altor pixeli sau blocuri de pixeli
(b) Regiuni uniforme obtinute prin impartirea unor regiuni mai mari care nu sunt omogene

In aceasta lucrare se va implementa o metoda din prima categorie (a), denumita „region growing".

Metoda *region growing* are la baza un proces iterativ prin care regiuni ale imaginii sunt fuzionate incepand de la regiuni primare (care pot fi pixeli sau alte regiuni mici – celule de baza). Iteratiile de crestere se opresc atunci cand nu mai sunt pixeli de procesat!

Algoritm:

1. Se segmenteaza imaginea in celule de baza (dimensiune >= 1 pixel).
2. Fiecare celula este comparata cu vecinii ei folosind o masura de similaritate. In caz afirmativ (valoarea metricii de similaritate < prag) celulele sunt fuzionate intr-un fragment mai mare si se actualizeza trasaturile regiunii folosite la masura similaritatii (de obicei prin mediere ponderata).
3. Se continua procesul de crestere al fragmentelui prin examinarea tuturor vecinilor pana cand nu se mai pot realiza fuziuni.
4. Se trece la urmatoarea celula ramasa nemarcata si se repeta pasii 2-3. Algoritmul se opreste atunci cand nu au mai ramas celule nemarcate.

4.2. Mersul lucrării

4.2.1. Sablonul de procesare recomandat

1. Se filtreaza imaginea cu un filtru trece jos Gaussian (ex. de dim 5) pentru eliminarea zgomotelor.
2. Se alege un punct de start (seed point) ales cu ajutorul mouse-ului (prin tratarea mesajului corespunzator - vezi L3). Puteti folosi ca si exemplu / punct de plecare functiile `L3_ColorModel_Build` si `CallBackFuncL3` prezentate in L3.
3. In functia de procesare principala pe care o apelati din meniul principal (vezi `L3_ColorModel_Build`) este recomandat sa faceti urmatoarele:
   - Sa filtrati imaginea cu un filtru trece jos Gaussian (ex. de dim 5) pentru eliminarea zgomotelor
   - Sa convertiti imaginea sursa din BGR in HSV si sa transmiteti pointerul matricii H (Hue) funciei CallBack de tratare a evenimentului mouse-ului (vezi `L3_ColorModel_Build`).

4. Codul aferent algoritmului de region growing sa il adaugati in functia CallBack de tratare a evenimentului mouse-ului.

4.2.2. Algoritmul de creștere a regiunilor

Codul aferent algoritmului de region growing este recomandat sa il adaugati in functia CallBack de tratare a evenimentului mouse-ului. Componenta de culuare pe care se va lucra este H (Hue).

Se va aloca o matrice de etichete *labels* de dimensiunea imaginii. Se initializeaza cu 0 fiecare element. De fiecare data cand un punct se va adauga la regiune curenta, se va marca pozitia corespondenta din matricea labels. Matricea labels va fi folosita si ca un indicator de parcurgere al pixelilor (0 daca pixelul inca nu a fost parcurs, 1 daca pixelul a fost parcurs).

```
Mat labels = Mat::zeros(H.size(),CV_16UC1);
```

1. Pentru punctul de start avand coordonatele imagine (x, y) calculati o valoare medie (*Hue_avg*) a lui Hue intr-o vecinatate de dimensiune w x w (w= 3, 5, 7 ...) in jurul punctului de start (o masura de precautie suplimentara pentru a nu lua in considerare zgomote.
2. Se adauga elementul de start in lista FIFO. Pentru implemnetarea listei FIFO puteti folosi clasa container QUEUE (vezi anexa). Ati mai folosit-o si la laboratorul de Procesarea Imaginilor: Canny sau Etichetare). Se seteaza eticheta curenta k = 1 (in final toti pixelli din regiune vor trebui sa aiba eticheta 1) si N = 1 (numarul de pixeli din regiune)
3. Algorimul de region growing:
   while (coada nu seste goala)
   - pentru fiecare vecin (i,j) al pixelului din pozitia „bottom" a listei:
     - daca labels(i,j) =0 (pixel neprocesat inca) si abs(Hue(i,j)-Hue_avg) < *T*:
       - adauga pixelul (i,j) in lista FIFO la pozitia top (este adaugat la regiunea curenta)
       - pixelul (i,j) primeste eticheta k: `labels(i,j)=k`
       - se actualizeaza valoarea medie a lui Hue:

$$Hue\_avg = \frac{N * Hue\_avg + Hue(i,j)}{N+1}$$

       - incrementeaza N
   - sterge elementul de pe pozitia bottom a listei FIFO

   end while
4. Se vor afisa pixelii din regiunea curenta (labels(i,j) = 1) in imaginea destinatei cu alb (cei de fond vor fi negri intr-o imagine/fereastra destinatie.

Observatie
Valoarea pragului *T* se va alege in functie de parametrii modelului de culoare construit anterior (lucrarea 3) *T= konst*hue_std*, unde *konst* = 2 .. 3;

4.2.3. Postprocesarea imaginii segmentate

Imaginea segmentata poate prezenta zgomot. O metoda simpla de postprocesare consta in eliminarea acestor zgomote (pixeli albi in interiorul obiectului si pixeli negrii in zona de fond prin operatii morfologice (dilatare, eroziune) aplicate repetate / succesiv (L3).

Teme de casa / proiect de semestru:
Algoritmul se poate aplica si automat (fara a selecta punctul de start manual):
- k= 1; //eticheta curenta
- 1. se considera primul pixel din imagine (neechitat, neparcurs inca) ca seed point (k=1)

- 2. sa aplica pasii 2 – 3 (din 4.2.2) (cu diferenta ca pixelii din regiunea curenta primesc eticheta k)
- 3. k++ si se repta 1 – 3 pana cand se epuizeaza toti pixelii din imagine

Anexa:

Pentru implemnetarea listei FIFO puteti folosi si clasa container QUEUE

```
#include <queue>
using namespace std;
```

In functia de CallBack:

```
queue <Point> que;

k=1; //ethiceta curenta
N=1; // numarul de pixeli din regiune
que.push(Point(x,y)); // adauga element (seed point) in coada
// acesta primeste eticheta k

while (!que.empty())
{	// Retine poz. celui mai vechi element din coada
	Point oldest=que.front();
	que.pop();	// scoate element din coada

	int xx=oldest.x; // coordonatele lui
	int yy=oldest.y;

	// Pentru fiecare vecin al pixelului (xx, yy) ale carui coordonate
	// sunt in interiorul imaginii
		// Daca abs(hue(vecin) – Hue_avg)<T si labels(vecin) == 0
			// Aduga vecin la regiunea curenta
			// labels(vecin) = k
			// Actualizeaza Hue_avg (medie ponderata)
			// Incrementeza N

}
```