Tema 1 – Familiarizare cu limbajul Octave

Responsabili: Ana Ion (ana.ion4@gmail.com)
Ana-Cristina Surdu (anacristinasurdu@gmail.com)

Data publicarii: 25 februarie 2011
Data predarii: 14 martie 2011

Obiective: In urma realizarii temei propuse, studentul va putea folosi limbajul Octave pentru crearea de programe care realizeaza calcul matriceal. De asemenea studentul va invata cum sa utilizeze lucrul cu fisiere in limbajul Octave.

Subiectul temei: procesare de imagini

O imagine este reprezentata printr-o matrice de numere de dimensiune m si n, unde m este numarul de linii si n este numarul de coloane. Fiecare element din aceasta matrice corespunde unui pixel si are o valoare care se traduce in culoare (potrivit RGB) si intensitate.

Asupra imaginilor se pot realiza diferite procesari folosind algoritmi care modifica pixelii local sau global. Aceasta tema vizeaza aplicarea algoritmilor de modificare a contrastului, a realizarii de filtre, precum si calculul entropiei unei imagini.

Tema este alcatuita din 3 parti care pot fi rezolvate in mod individual. Se va scrie o functie Octave separata pentru fiecare dintre aceste parti. Se va respecta formatul de trimitere a temelor descris in **regulamentul** cursului de Metode Numerice.

Imaginile asupra carora se vor aplica prelucrarile vor avea formatul PGM si se vor utiliza doar imagini in tonuri de gri pentru care fiecare pixel are o valoare intre 0 si 255.

Exemplu format PGM:

```
P2                              tipul imaginii; toate fisierele vor avea acelasi tip (P2) => ignorati linia
# feep.pgm              comentariu
24 7            numar de coloane, numar de linii
255             valoarea maxima a pixelilor din imagine; aceasta valoare va fi mereu 255
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0                matricea de pixeli
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0 15  0
0  3  3  3  0  0  0  7  7  7  0  0  0 11 11 11  0  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0  0  0
0  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Pentru a vizualiza imagini in format PGM puteti folosi diferite viewere(ex: gimp). O imagine poate fi convertita dintr-un format specific(JPG de exemplu) intr-o imagine PGM. Pentru aceasta deschideti imaginea folosing GIMP, Save as..., selectati salvarea fiserului cu extensia PGM in format ASCII (nu RAW). Puteti vizualiza acum imaginea initiala, gray-scaled.

Rezultatele prelucrarilor vor fi scrise fiecare intr-un fisier ce respecta structura formatului PGM, astfel incat veti putea vizualiza rezultatele programului tot prin deschiderea imaginii folosind un viewer.

Cerinte

1. Modificarea contrastului (30p)

Sa se scrie o functie Octave “contrast” care primeste ca parametru numele fisierului care contine imaginea ce urmeaza sa fie prelucrata si scrie rezultatele obtinute in fisierul “out_contrast.pgm”.

Exemplu antet functie:

function contrast(file_in, a, b)

! Este **obligatoriu** ca fisierul in care este scrisa sursa sa fie salvat ca nume_functie.m. In acest caz, functia se va scrie in fisierul “contrast.m”.

Citirea din fisier in Octave se realizeaza intr-un mod foarte asemanator cu cel din C. Documentatie si formate specifice gasiti aici: http://www.mathworks.com/help/techdoc/.

Modificarea contrastului imaginii se face prin ajustarea intensitatii fiecarui pixel in functie de 2 valori date (in antetul de mai sus, a si b). Pentru inceput se determina nivelul minim si nivelul maxim de gri din imaginea data, pentru ca mai apoi sa se rescaleze fiecare valoare astfel incat sa apartina intervalului [a, b]. In realizarea acestei prelucrari se foloseste urmatoarea formula, care se aplica pentru fiecare pixel:

$$In = (b-a) * (I0-min) /(max-min) + a$$

Notatiile din formula de mai sus reprezinta:

In – noua intensitate

I0 – intensitatea initiala

min – cea mai mica valoare a unui pixel din imagine (cel mai “inchis” gri)

max – cea mai mare valoare a unui pixel din imagine (cel mai “deschis” gri)

a – noua valoare a intensitatii minime

b – noua valoare a intensitatii maxime

2. Aplicarea filtrelor (40p = 4x10p)

Sa se scrie o functie Octave care aplica unei imagini date ca parametru de intrare unul dintre filtrele prezentate mai jos. Functia va avea urmatoarea semnatura si va scrie rezultatul intr-un fisier “out_filter_type.pgm”:

function filter(file_in, filter_type)

Tipul filtrului este transmis prin parametrul “filter_type” si poate avea valorile: “smooth”, “blur”, “sharpen” sau “emboss”.

Se considera ca imaginea reziduala pentru pixelii marginali nu se poate calcula, deci implicit este 0.

2.1. Filtre de convolutie

Un filtru de convolutie este reprezentat printr-o matrice de forma

0 0 0

0 1 0

0 0 0 /1 + 0

aplicata punctului curent si punctelor din jurul lui. Fiecare punct capata astfel o pondere, iar suma valorilor ponderate este impartita la un factor, la care se aduna in unele cazuri si un deplasament.

Intrucat matricea de mai sus lasa imaginea nemodificata, filtrul poarta numele de filtru identitate.

Filtrele de convolutie sunt de dimensiuni variate. Unele filtre sunt simetrice, aplicate uniform punctelor din jur, altele sunt nesimetrice, cum ar fi filtrele de detectare a marginilor. Cu cat un filtru este de dimensiune mai mare, cu atat va ramane o margine mare care nu poate fi prelucrata.

2.2. Filtre

2.2.a. Uniformizarea (smooth)

Aceasta operatie presupune crearea unui pixel care are ca valoare media tuturor punctelor din jur, inclusiv a propriei valori. Valoarea ponderilor va fi 1 pentru toate punctele iar factorul la care se imparte va fi 9. Matricea de convolutie are urmatoare forma:

1 1 1

1 **1** 1

1 1 1 /9 + 0 (in cazul acestui filtru rezultatul va fi media aritmetica a celor 9 valori din matrice la care se aduna deplasamentul egal cu “0”)

Exemplu de aplicare a filtrului smooth:

Fie matricea urmatoare:

a b c

d **e** f

g h i

Prin aplicarea filtrului smooth, valoarea pixelului evidentiat (**e**), va deveni:

*(1*a + 1*b + 1*c + 1*d + 1*e + 1*f + 1*g + 1*h + 1*i)/9 +0*

2.2.b. Uniformizarea triunghiulara (blur)

Filtrul triunghiular localizeaza schimbarile de culoare dintr-o imagine si creeaza culori intermediare pentru a netezi marginile. Filtrul este:

1 2 1

2 4 2

1 2 1 / 16 + 0

Acest filtru da un efect circular deoarece pixelii care sunt mai departe de margine au pondere mai mica. Efectul obtinut este similar cu cel dintr-o fotografie gresit focalizata.

2.2.c. Sharpen

Acesta este aproximativ inversul filtrului blur, scopul lui fiind detectarea diferentelor intre pixeli si accentuarea acestor diferente. Formula sa este:

0 -2 0

-2 11 -2

0 -2 0 / 3 + 0

Cu cat este mai mare diferenta intre punctele cu ponderi negative si punctul care este modificat, cu atat este mai semnificativa schimbarea valorii punctului central. Gradul de accentuare este stabilit de ponderea din centru.

2.2.d. Emboss

Acesta este de fapt un filtru de detectare a marginilor obiectelor. Functionarea se bazeaza pe diferente intre punctele adiacente(suma ponderilor este zero), astfel incat cu cat diferentele intre puncte sunt mai mari, cu atat

valoarea finala va fi mai mare.

De obicei se foloseste un offset de 127 pentru a lumina imaginea finala. Matricea pe care o veti folosi pentru acest filtru este:

-1 0 -1

0 4 0

-1 0 -1 / 1 + 127

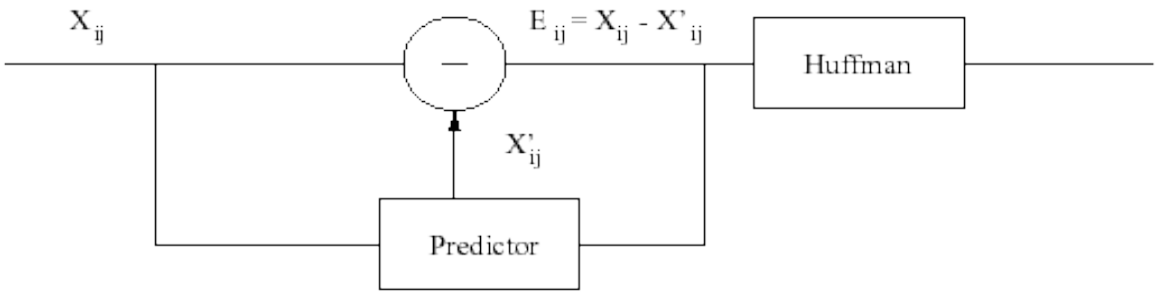
Filtrul actioneaza pe diagonale.

3. Calculul entropiei (30p)

Sa se scrie o functie Octave care sa calculeze valoarea *entropiei* pentru o imagine de dimensiune *mxn*, data ca parametru de intrare. Functia va avea urmatoarea semnatura si va scrie rezultatul intr-un fisier “entropy.txt”:

function entropy(file_in, a, b, c)

Algoritmul ce trebuie implementat este cel descris in continuare. Mai jos este prezentata schema generala:



In figura de mai sus Xij reprezinta valoarea originala a pixelului aflat la coordonatele (i,j), iar X'ij reprezinta valoarea ce se obtine la iesirea componentei Predictor. Algoritmul pe care il veti implementa se bazeaza pe faptul ca valorile unor pixeli invecinati sunt destul de apropiate. Pasii algoritmului ce trebuie implementat sunt:

- Obtinerea valorii pixelilor dupa trecerea prin filtrul Predictor.
- Calcularea imaginii reziduale.
- Calculul *entropiei* imaginii reziduale obtinute.

Componenta Predictor are rolul de a calcula valoarea unui pixel pe baza valorilor pixelilor vecini acestuia dupa formula:

$$P(x,y) = a*I(x-1, y) + b*I(x-1, y-1) + c*I(x, y-1),$$

unde P(x,y) reprezinta valoarea X' a pixelului aflat la coordonatele (x,y), iar I(x,y) reprezinta valoarea originala a pixelului aflat la coordonatele (x,y). Valorile coeficientilor reali a, b si c le veti da ca parametri functiei.

Se considera ca imaginea reziduala pentru pixelii marginali nu se poate calcula, deci implicit este 0.

Pasul urmator al algoritmului il reprezinta obtinerea imaginii reziduale. Valoarea unui pixel al imaginii reziduale se obtine ca diferenta intre valoarea obtinuta pentru respectivul pixel la iesirea componentei Predictor si valoarea originala a aceluiasi pixel, adica:

$$E(i,j) = I(i, j) - ceil\{ a*I(i-1, j) + b*I(i-1, j-1) + c*I(i, j-1)\}$$

Pentru aflarea *entropiei* va trebui sa calculati probabilitatea de aparitie a fiecarui simbol in imaginea reziduala obtinuta. Probabilitatea de aparitie a unui simbol reprezinta raportul dintre numarul de aparitii al acelui simbol si numarul total de pixeli. De exemplu sa presupunem ca 30 de pixeli au valoarea "5" in imaginea reziduala

obtinuta si ca imaginea originala contine 64 de pixeli in total. Probabilitatea de aparitie a simbolului "5" este in acest caz: p("5") = 30 / 64 = 0.46.

Dupa calcularea probabilitatii de aparitie a tuturor simbolurilor din imaginea reziduala *entropia* se calculeaza dupa formula:

$$-\sum_{1}^{m} p(si) * log2(p(si))$$

unde m reprezinta numarul total de simboluri unice din imaginea reziduala.

Valoarea *entropiei* reprezinta o aproximatie a procentului de compresie care se obtine atunci cand se foloseste un algoritm Huffman. Tema va trebui sa scrie in fisierul de iesire aceasta valoare calculata.