ALGORITMI PARALELI ȘI DISTRIBUIȚI
Tema #3 Procesare de imagini folosind MPI

Termen de predare: 06 Ianuarie 2020

Responsabili Temă: **Elena Apostol, Alexandru Hogea, Alexandru Maxim**

Contextul temei:

Procesarea de imagini este una dintre cele mai raspandite procesari care au loc in sfera digitala. Imaginile provin dintr-o multitudine de surse si pot fi procesate in la fel de multe moduri. Totusi, pe langa diversitatea de filtre care se pot folosi, conteaza si mai mult modul in care se aplica. Programarea distribuita este utila in acest scop. De ce? Pentru ca o imagine poate fi fragmentata in bucati ce pot fi procesate individual, in paralel.

Obiectivele temei:

Tema consta in realizarea unui algoritm distribuit de procesare a imaginilor folosind filtre. Aplicarea unui filtru presupune modificarea valorilor pixelilor unei imagini. Acest proces poarta numele de **convolutie**. Valoarea fiecarui pixel va fi actualizata cu suma inmultirilor dintre fiecare element din matricea **kernel** a filtrului rotita cu 180° cu valoarea fiecarui pixel din vecinatatea pixelului dorit (inclusiv valoarea pixelului respectiv).

$$\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right)[2,2] = (i \cdot 1) + (h \cdot 2) + (g \cdot 3) + (f \cdot 4) + (e \cdot 5) + (d \cdot 6) + (c \cdot 7) + (b \cdot 8) + (a \cdot 9).$$

Figura 1: Exemplu de produs de convolutie intre doua matrici, prima reprezentand matricea kernel (inainte de rotatie) si a doua reprezentand pixelul pentru care se doreste actualizarea valorii si vecinii sai. Pixelul asupra caruia se actioneaza este pixelul central, de pe pozitia [2,2]

O explicatie grafica a produsului de convolutie se regaseste in figura [1] de mai jos:

[1] Spectral–spatial feature extraction using orthogonal linear discriminant analysis for classification of hyperspectral data - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Performing-convolution-by-matrix-multiplication-f-is-set-to-3-in-this-figure_fig1_313425538[accessed 6 Dec, 2019]

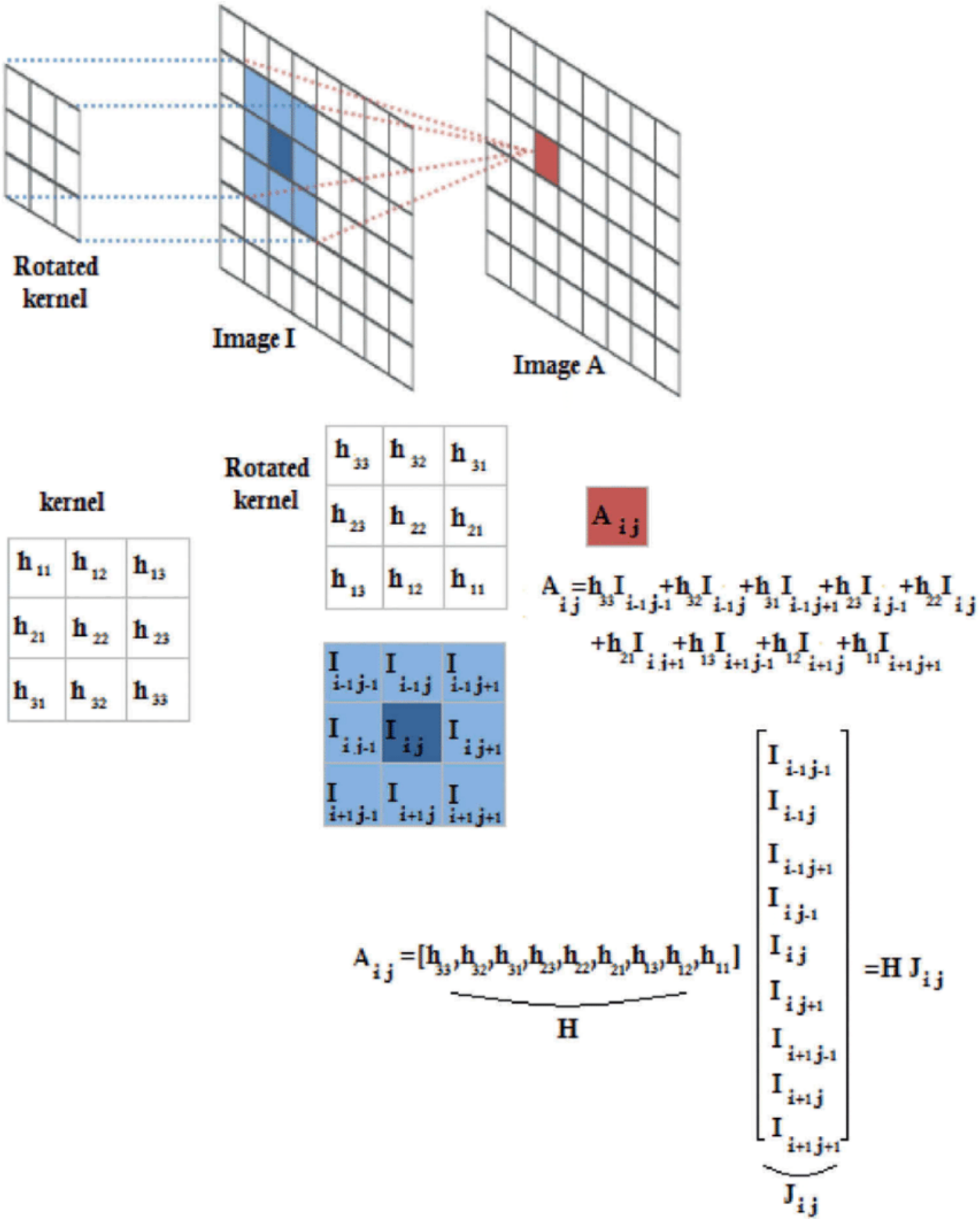

Figura 2: Reprezentare grafica a produsului de convolutie unde valoarea pixelului din centru este actualizata.

Aplicarea filtrelor:

Programul va trebui sa opereze pe unul sau mai multe filtre din urmatoarea lista: “smooth“, “blur“, “sharpen“, “mean“ si “emboss“. Filtrele vor fi reprezentate printr-o matrice de 3 x 3. Mai jos aveti detaliat fiecare filtru:

A. Smoothing filter Amelioreaza diferentele din imagini:

$$K = \frac{1}{9} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

B. Approximative Gaussian Blur filter Reduce zgomotul de fundal din imagini:

$$K = \frac{1}{16} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

C. Sharpen Accentueaza detalii in imagini:

$$K = \frac{1}{3} \cdot \begin{bmatrix} 0 & -2 & 0 \\ -2 & 11 & -2 \\ 0 & -2 & 0 \end{bmatrix}$$

D. Mean removal Asemanator cu “Sharpen“, doar ca utilizeaza si valorile pixelilor vecini aflati pe diagonale:

$$K = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

E. Emboss Utilizat pentru detectarea muchiilor:

$$K = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

Interactiunea cu programul

Fisiere de intrare si iesire:

Programul vostru va trebui sa citeasca si sa genereze fisiere **.pnm/.pgm**. Structura acestor fisiere este urmatoarea:

```
P(5 or 6)\n
width height\n
maxval (max 255)\n
height * width * (1 or 3) bytes representing the image
```

Daca pe prima linie apare **P5** inseamna ca poza este **PGM**, deci alb-negru. Asadar, fiecare pixel va avea o valoare intre 0 si **maxval**. Daca apare **P6**, atunci poza este **PNM**, deci color si fiecare pixel va avea cate un byte pentru fiecare *canal de culoare* (RGB - rosu, verde si albastru). In acest caz, operatiile se vor face pentru fiecare canal de culoare al pixelului.

Rularea programului:

Programul va rula astfel:

```
mpirun -np N ./tema3 image_in.pnm image_out.pnm filter1 filter2 ... filterX
```

Punctare:

- 5 puncte - implementarea filtrelor pe un singur proces
- 10 puncte - implementarea filtrelor in varianta distribuita, folosind MPI, pe minim 2 procese
- 45 puncte - poza rezultata sa fie corecta, indiferent de nr. de procese pe care este rulat algoritmul
- 20 puncte - scalarea programului in functie de numarul de procese folosite
- 20 puncte - documentarea comportamentului de scalare in README

O tema care nu compileaza nu va fi punctata.

Atentionari:

Dimensiunea pozei de intrare **va fi egala** cu dimensiunea pozei de iesire.

Asa cum am precizat la punctul anterior, **se pot aplica mai multe filtre**.

Deoarece pixelii de la marginile pozei nu au vecini pe toate directiile, va trebui **sa bordati matricea imaginii cu 0** pentru a putea calcula noile valori si pentru acei pixeli.

Scopul final al temei este **scalarea** prin aplicarea unui algoritm distribuit. Aveti, asadar, grija la modul in care partitionati algoritmul, astfel incat utilizarea mai multor procese sa ajute **la scaderea timpului de rulare**! Numarul de procese minim este 2 si maxim, **numarul de core-uri disponibile**. Daca, de exemplu, aveti 8 core-uri, veti testa (si documenta) cu 2, 3, ..., 8 procese.
Atentie! E posibil ca problema sa nu scaleze pentru poze de intrare de dimensiune mica.

Pentru a converti o imagine dintr-un anume format (ex. JPG) in format PGM/PNM se poate folosi **GIMP**. Se deschide poza in editor. Se selecteaza optiunea File – Export as... si se va alege formatul dorit (PGM/PNM). Se va alege apoi optiunea RAW, pentru formatul binar.

Arhivare:

Arhiva va trebui sa contina **fisierele sursa**, un fisier **README** si un fisier **Makefile** care sa aiba doua reguli: *build* (din care sa rezulte un executabil numit **tema3**) si *clean*.

Arhiva va trebui sa fie de tip **.zip** si numele sau sa respecte urmatorul format: **Nume_Prenume_Grupa** (ex: **Hogea_Alexandru_332CC.zip**).

Echipa noastra va ureaza "Sarbatori fericite!"