Neleptcu Daniel-Andrei – Grupa 332AB

Tema 1 – Prelucrarea imaginilor

1. Principiul general de functionare al codului.

Deoarece citirea, prelucrarea si scrierea pixelilor nu se poate realiza intr-un singur ciclu de ceas am decis sa am un bloc always care trateaza partea algoritmica a codului (care este trigger-uit de schimbarea starii actuale) si un bloc always care trateaza evolutia starilor si marcarea stadiului in care a ajuns algoritmul.

Primul bloc always este cel principal trigger-uit de frontul pozitiv al ceasului in care marchez punctul de pornire la prima executie a acestuia, avand ca la restul executiilor sa actualizez starea curenta si sa verific daca am ajuns la starea finala a unei etape (mirror/grayscale/sharpen).

In al doilea bloc always este implementat state machine-ul care rezolva cerintele. Starile 0->8 sunt dedicate pentru mirror, 9->16 sunt dedicate pentru grayscale si 17->33 sunt dedicate pentru sharpen.

2. Mirror

Algoritmul este destul de simplu, parcurg fiecare coloana pana la elementul 31 si interschimb cu complementul sau 63 – i, stocand elementul si complementul sau in doua variabile temporare. Cand se ajunge la elementul 32 se incrementeaza j-ul (trec la urmatoarea coloana) si repet procesul pana cand ajung la ultima coloana. Starea 8 marcheaza completarea procesului de mirror si trecerea la grayscale.

3. Grayscale

Pentru simplitate am decis sa stochez cele 3 parti din pixel (R, G, B) in 3 variabile: a, b, c. Am aflat minimul si maximul acestor 3 variabile si le-am adunat in grayscale_minmax care reprezinta pixelul de grayscale, caruia i-am concatenat valoarea 0 pe 8 biti la dreapta (valoarea fiind pe 8 biti are deja bitii la stanga 0 deci nu mai trebuie concatenati si acolo). Dupa ce scriu pixelul de grayscale voi trece la coloana urmatoare pana ajung la ultimul element, apoi voi trece pe randul urmator si repet pana cand ajung la elementul 63:63

4. Sharpen

Am decis sa impart problema in 4 etape:

1. Tratarea colturilor : acest lucru l-am facut separat pentru fiecare colt, deoarece acestea nu respecta vreo regula si fiecare se inmulteste cu elemente nenule de pe pozitii diferite.
2. Tratarea elementelor de pe linia 0, deoarece acestea au elemente nule deasupra
3. Tratarea elementelor de pe linia 63, deoarece acestea au elemente nule dedesupt

4. Tratarea elementelor de pe liniile 1-62:
   4.1 Tratarea elementelor de pe coloanele 1-62 deoarece acestea nu au elemente nule in vecinatate
   4.2 Tratarea elementelor de pe coloanele 0 si 63 pentru ca acestea au elemente nule in stanga respectiv dreapta.

Pentru realizarea sharpen-ului am ales sa pastrez 3 linii din matricea de pixeli realizata la pasul anterior pentru a putea realiza convolutia pe pixelii nemodificati de sharpen. Initial retin primele 3 randuri, apoi realizez etapa 1(colturile ss si sd) si 2. Dupa aceea intru in etapa 4. Dupa completarea unei linii de la etapa 4 voi avansa cu cele 3 linii memorate din matricea de pixeli astfel (linia 0 devine linia 1, linia 1 devine linia 2 si linia 2 devine a doua linie fata de cea la care m-am aflat anterior). Repet procesul pana cand ajung la ultima linie unde voi realiza etapa 1 (colturile js si jd), respectiv etapa 3. Verificarea rezultatului se face in operatorul ternar astfel: daca valoarea rezultata in urma aplicarii convolutiei este mai mare de 255, atunci valoarea va ramane 255, daca valoarea rezultata este mai mica decat 0, atunci aceasta ramane 0 (am observat comparand informatiile date de checker-ul offline si prelucrarea manuala a datelor din fisierele .data din interiorul folderului tests). Apoi concatenez rezultatul pe 8 biti cu valoarea 0 pe 8 biti la stanga si la dreapta si scriu pixelul.

DONE (●'◡'●)