README
Tema 1

Nuță Leonard-Florian
Grupa 333AA
Facultatea de Automatică și Calculatoare

Descrierea generală a abordării problemei

Am ales să realizez un automat cu stări la care, pe lângă **starea următoare** (next_state), am adăugat și **linia următoare** (next_row) și **coloana următoare** (next_col).

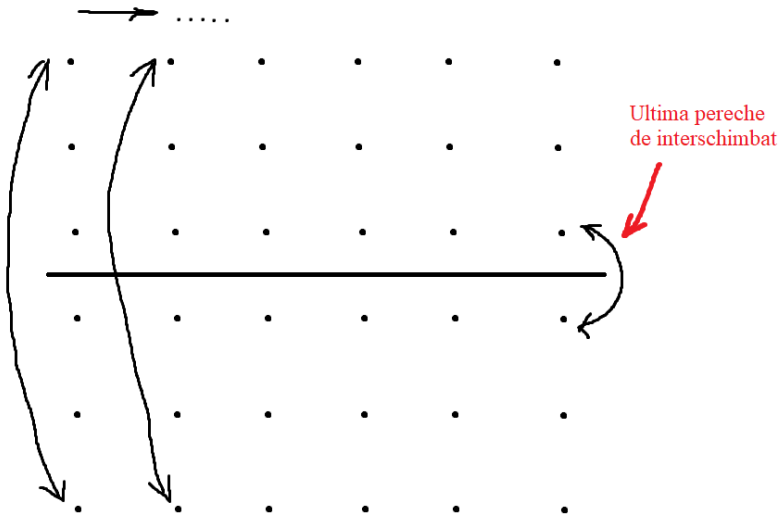
Automatul cuprinde 30 de stări (31 cu default), partea de Mirror cuprinzând stările 0,1,2,3 și 4, partea de Grayscale cuprinzând stările 5,6,7 și 8, iar partea de Filter ținând de la starea 9 până la starea 29.

Mirror

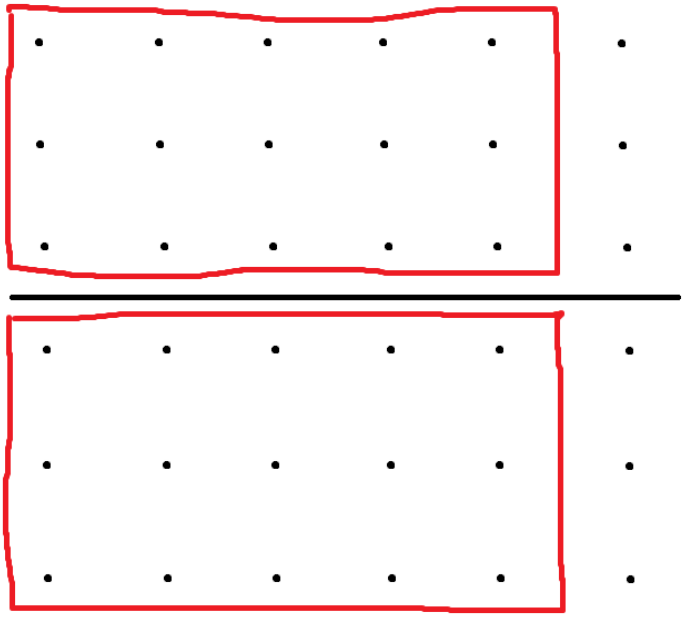
Pentru Mirror am ales să merg în perechi de pixeli, unul din jumătatea de sus a imaginii, iar celălalt în jumătatea de jos (ne vom imagina că matricea noastră este separată orizontal printr-o axă invizibilă). Parcurgând matricea de la stânga la dreapta, linie cu linie, de sus în jos, am efectuat interschimbarea de pixeli până la ultima pereche posibilă, și anume cea care are pixelul de sus pe poziția (31,63), unde linia 31 este cea aflată în imediata vecinătate a axului imaginar, iar coloana 63 este ultima coloană din matrice fiindcă imaginea are o rezoluție de 64x64.

Algoritmul sună în felul următor:

1. Memorez pixelul de sus în pix1.
2. Mă deplasez pe linia simetrică față de axul imaginar și memorez pixelul în pix2.
3. Înlocuiesc pixelul de jos cu pix1.
4. Mă deplasez înapoi sus și înlocuiesc pixelul de sus cu pix2
5. Repet acest lucru pentru toate perechile de pixeli posibile.



Diferitele cazuri de deplasare sunt următoarele:

1. 

Echivalent cu if(col<63) → Doar mărim coloana

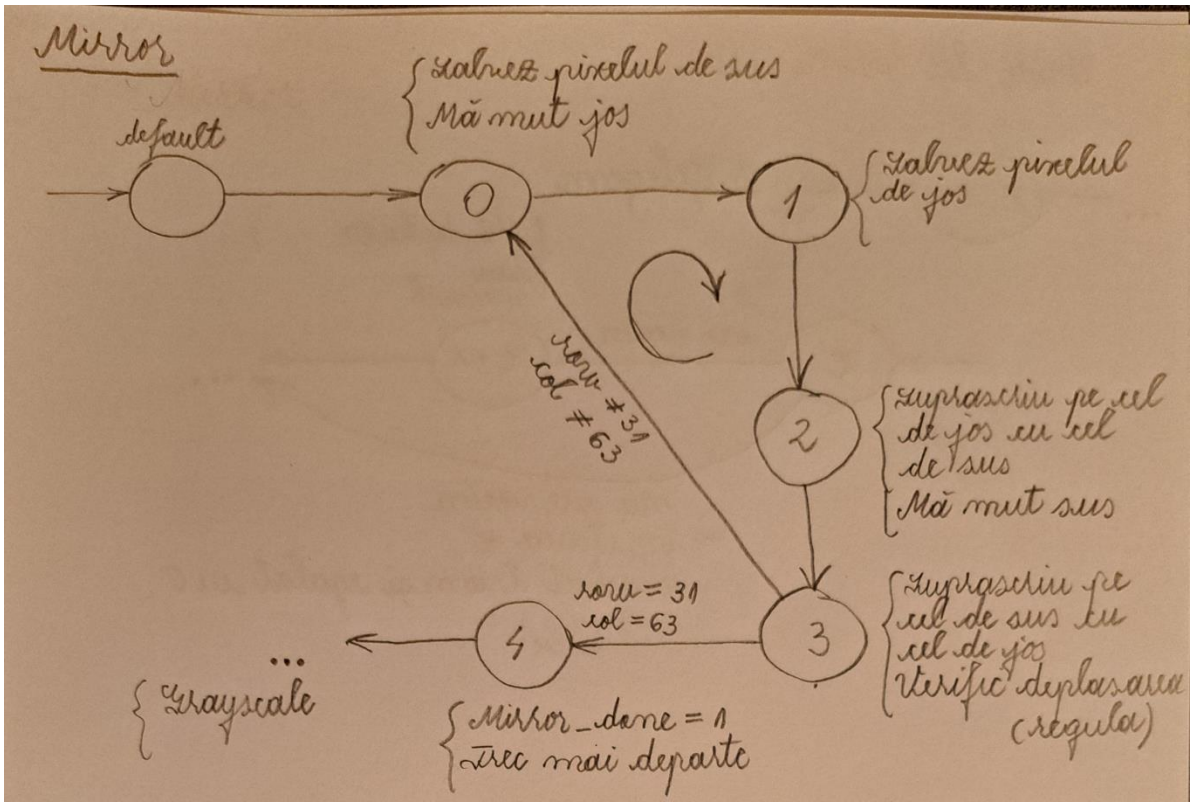
2. 

Echivalent cu if(col==63 && row<31) → Creștem rândul și setăm coloana la 0

3.

Echivalent cu if(col==63 && row==31) → S-a ajuns la ultima pereche, Mirror este gata.

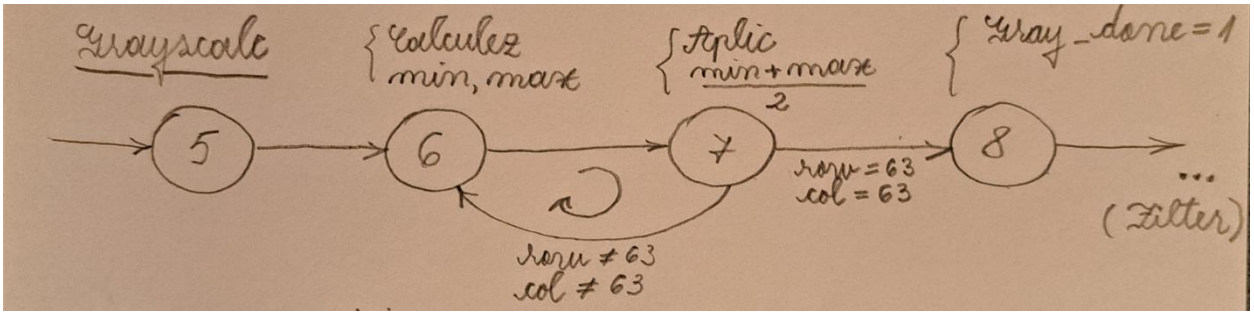
Partea de Automat



Grayscale

La Grayscale lucrurile devin mai simple întrucât trebuie doar să realizăm pentru fiecare pixel maxim-ul dintre canalele R, G și B, respectiv minimul dintre acestea, să egalăm R și B cu 0 și G să fie dat de formula **(min+max)/2.** Se schimbă doar modul de parcurgere al matricii (ne oprim la elementul din colțul din dreapta-jos).

Partea de automat



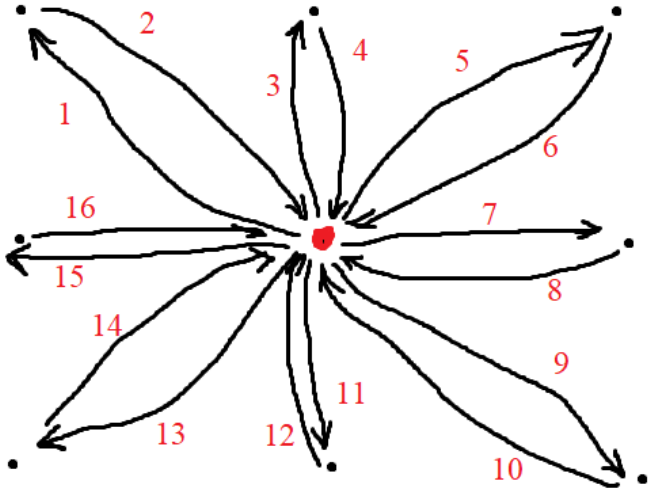
Filter

La Filter trebuie sa avem grijă la mai multe aspecte:

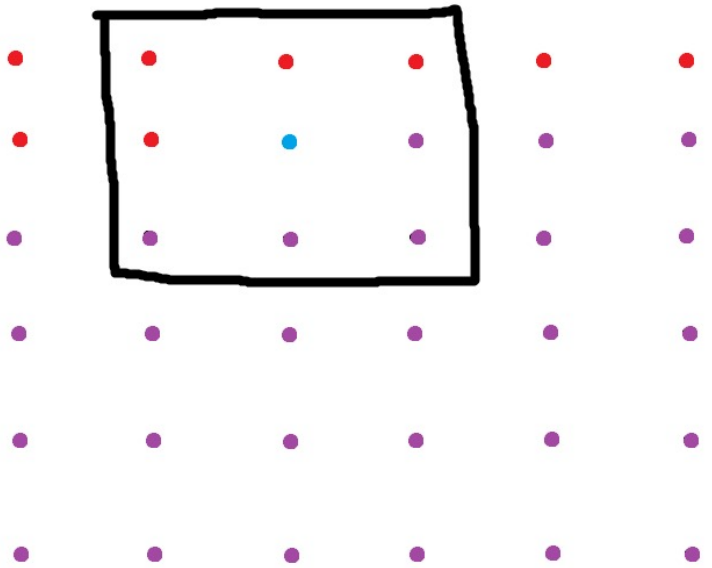
1. Verificarea posibilității ca un pixel să aibă anumiți vecini în funcție de poziția sa în matrice.
2. Deodată ce pixelul a fost modificat, trebuie să ținem cont de faptul că și acesta va deveni la rândul său vecin pentru alt pixel căruia îi aplicăm aceeași procedură.
3. Neîncadrarea în intervalul 0 – 255 a rezultatului obținut în urma înmulțirii element cu element a matricii de vecini cu SharpMatrix și adunarea tuturor elementelor

Am declarat suma pe 15 biți și am avut grijă ca atunci când este pozitivă si mai mare ca 255, aceasta să devină chiar 255, iar dacă este negativă să devină 0.

Pentru aflarea vecinilor, am ales să fac următoarea parcurgere.



Deși puteam sa facem cache până la 6 rânduri din matrice, am profitat de ipoteza exercițiului 2 și am salvat ce era in canalul G în canalul R, fiindcă R era gol. Accesul la informație s-a facut la biții [23:16] (R) doar în cazul vecinilor din stânga, stânga-sus, sus, dreapta-sus (Mai multe detalii in comentarii în cod). Imaginea de mai jos explică și mai bine tot.



Rosu - Pixel modificat a carui valoare veche am stocat-o in R inainte sa-l modific
Albastru - Pixel curent unde ma aflu
Mov - Pixel care inca nu a fost modificat pentru ca nu am ajuns la el sa ii fac vecinii (sensul de parcurgere a matricii este linie cu linie, de sus in jos, de la stanga la dreapta)

Partea de Automat