

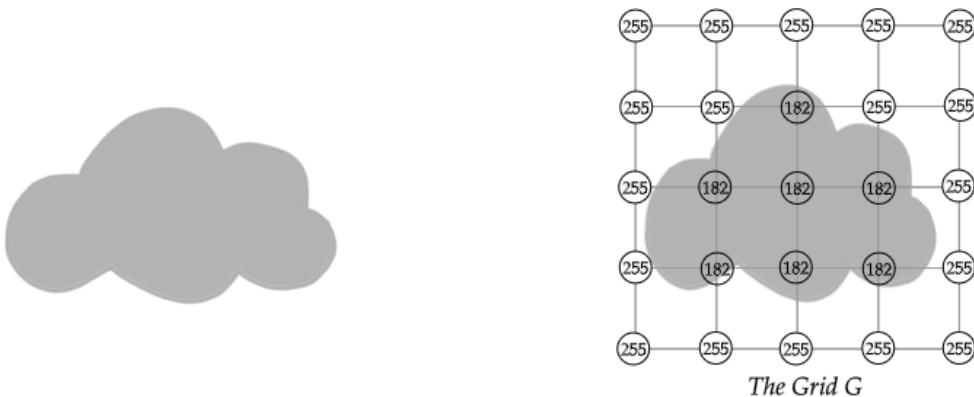
Cerință

Pornind de la implementarea secvențială, să se paralelizeze, cu ajutorul Pthreads în C/C++, un program care generează contururi pentru hărți topologice folosind algoritmul Marching Squares. Scopul final este de a se obține aceleași fișiere de ieșire ca la varianta secvențială, dar cu timpi de execuție îmbunătățiti. Tema are scopul de a vă obișnui cu ideea de a porni de la un cod existent și de a-l optimiza (în acest caz, folosind Pthreads).

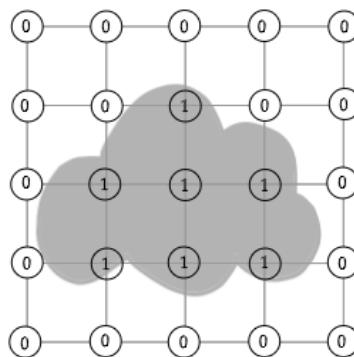
Algoritmul Marching Squares

Marching Squares este un algoritm de grafică introdus în anii 1980 care poate fi folosit pentru delimitarea contururilor dintr-o imagine. El poate fi folosit pentru a desena linii de altitudine pe hărți topografice, temperaturi pe hărți termice, puncte de presiune pe hărți de câmp de presiune, etc. Algoritmul funcționează după pașii descriși în continuare.

În primă fază, întreg domeniul D al imaginii de intrare este împărțit în pătrate de dimensiune fixă (pe baza unei valori prestabilite), și apoi se creează un grid G format din colțurile acestor pătrate (astfel, dacă avem N^2 pătrate, atunci G va fi format din $(N + 1)^2$ puncte). Acest lucru se poate observa în imaginile de mai jos.

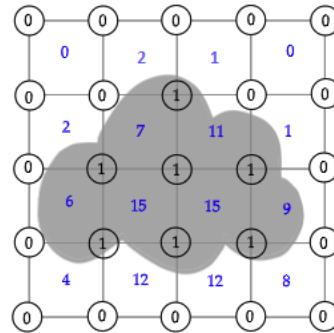
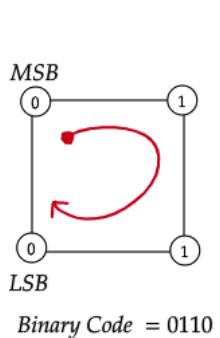


În continuare, se iterează prin grid pentru a se determina starea fiecărui punct raportată la o valoare de izolare σ . Pe baza valorii de izolare și a grid-ului, se creează mai departe un alt grid binar F , unde fiecare punct este 0 dacă valoarea punctului corespunzător din G este mai mare decât σ , sau 1 altfel. Un exemplu de astfel de grid binar (pentru $\sigma = 200$) se poate observa în imaginea de mai jos.

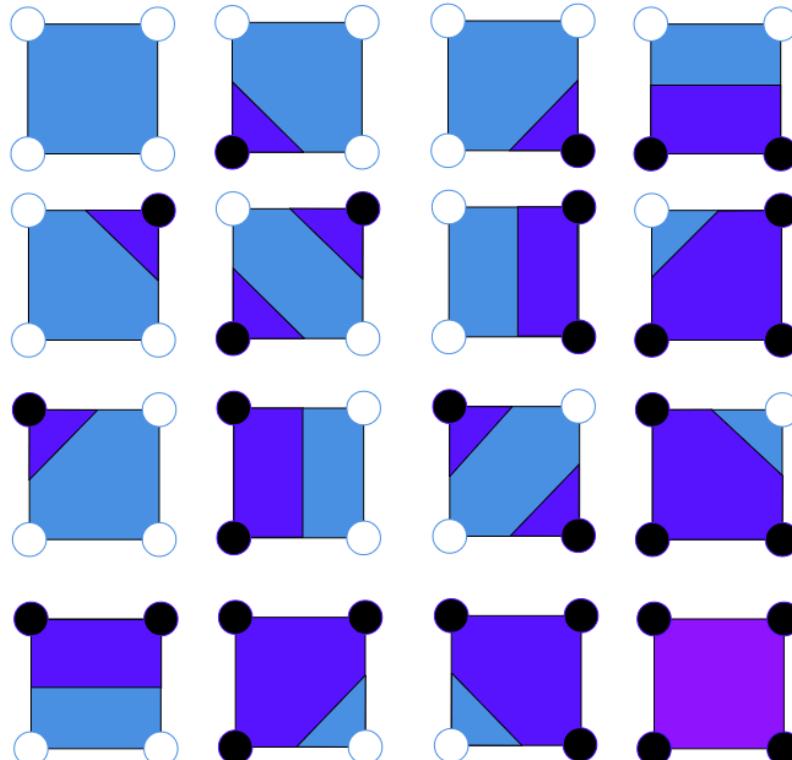


Fiecare pătrat component din F este mai departe asociat cu o configurație, formându-se câte un cod binar bazat pe valoarea fiecărui vârf al păratului, prin parcurgerea în sensul acelor de ceasornic de la colțul din stânga sus la colțul din stânga jos. În partea stângă a imaginii de mai jos, se poate observa o astfel de

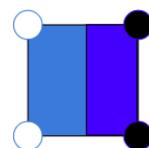
abordare, iar echivalentul decimal al codului binar specific unui pătrat reprezintă configurația sa, aşa cum este prezentat în partea dreaptă a imaginii.



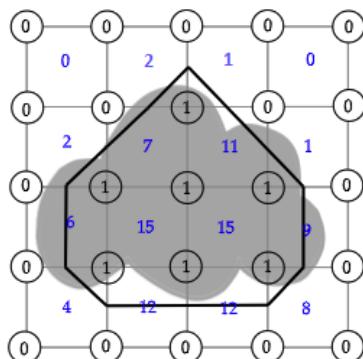
Configurația fiecărui pătrat din grid este potrivită cu o intrare în tabela de căutare a contururilor din imaginea de mai jos.



Pentru exemplul de mai sus în care rezultase codul binar 0110, conturul echivalent este cel din imaginea de mai jos.



Astfel, pe baza gridului binar calculat anterior și al tablei de contururi, rezultatul final al aplicării algoritmului Marching Squares pe imaginea inițială este cel din imaginea de mai jos.



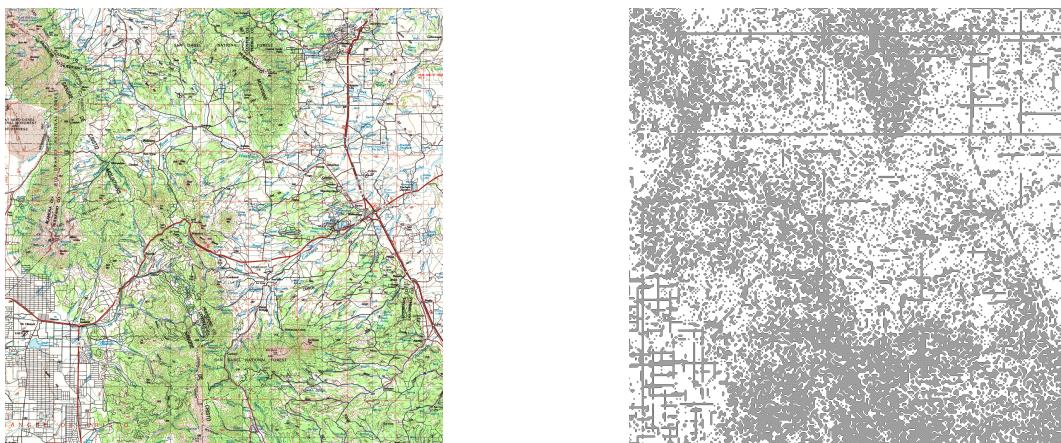
Detalii adiționale despre Marching Squares, pseudocodul, precum și imaginile prezentate mai sus, se pot găsi la [această adresă](#).

Detalii tehnice

În cadrul acestei teme, aveți deja implementat algoritmul Marching Squares într-o variantă secvențială, scopul fiind de a-l paraleliza folosind Pthreads în C/C++. Implementarea secvențială o găsiți în [repository-ul temei](#), în directorul **src** (fișierele **tema1_par.c**, **helpers.c** și **helpers.h**). Cei doi pași ai algoritmului (crearea grid-ului, respectiv identificarea contururilor din imaginea finală) se găsesc în funcțiile *sample_grid()* și *march()*.

În implementarea secvențială pe care o aveți la dispoziție, pentru uniformitate și un sampling adecvat, toate imaginile cu dimensiuni mari sunt scalate la 2048x2048 pixeli. Acest lucru se realizează prin intermediul funcției *rescale_image()* din fișierul **tema1_par.c**. Pentru redimensionarea imaginii, se folosește interpolare bicubică, despre care puteți afla mai multe detalii [aici](#). Un alt element important de menționat este faptul că imaginile cu care se lucrează sunt în format **PPM**.

Exemple de câte o imagine de intrare (stânga) și de ieșire (dreapta) se pot observa în imaginea de mai jos.



Implementarea paralelă

Scopul vostru este de a implementa varianta paralelă a algoritmului Marching Squares folosind Pthreads în C/C++, pornind de la implementarea secvențială, pe baza scheletului aflat în fișierul **tema1_par.c** din directorul **src** al repository-ului temei. În același director, găsiți și cele două fișiere cu funcții ajutătoare, pe