Proiect PI – Efecte speciale interesante pe imagini color

Vădean Cătălin

Grupa 30233

Contents

1. Obiective și managementul sarcinilor

Proiectul își propune realizarea unor efecte speciale interesante pe imagini color, folosind limbajul de programare C++. În urma discuțiilor cu asistentul de laborator, obiectivul este să se implementeze aproximativ două efecte mai complexe (cum ar fi posterize) și alte cinci efecte mai simple (de exemplu, efect monocrom, sepia, invert etc.).

Abordarea are în vedere aspecte de natură tehnică și creativă, pentru a demonstra înțelegerea conceptelor de procesare a imaginilor color și a modului în care se pot manipula canalele de culoare, se pot aplica filtre sau se pot realiza transformări geometrice. Prin aceste efecte, se urmărește punerea în practică a cunoștințelor de bază despre convoluții, transformări în diverse spații de culoare și alte operații fundamentale.

Scopul principal este crearea unei platforme modulare care să permită adăugarea și testarea mai multor efecte, fiecare dintre ele fiind implementat ca o funcție separată. Acest lucru va facilita extinderea proiectului pe parcurs și va oferi un cadru clar de organizare și testare.

---

2. Planul proiectului și organizarea pe etape

2.1 Identificarea principalelor sarcini

În prima fază, este necesară documentarea cu privire la diverse metode de procesare a imaginilor color, în special a tehnicilor de ajustare a canalelor de culoare și a filtrelor. Se va alege un set concret de efecte care să fie realizate până la finalul proiectului, respectând obiectivul stabilit împreună cu asistentul.

Pentru fiecare efect, se vor analiza cerințele (parametri, intrări/ieșiri, cost computațional), astfel încât să existe o imagine de ansamblu asupra modului în care acestea vor fi integrate într-un program unitar. De asemenea, este importantă planificarea timpului și a resurselor, pentru a ne asigura că implementarea și testarea vor fi realizate în intervalul stabilit.

2.2 Resurse și posibile provocări

Proiectul va fi dezvoltat în C++ și se va folosi biblioteca OpenCV pentru gestionarea facilă a imaginilor (încărcare, salvare, acces pixel cu pixel, funcții de bază pentru transformări). Este necesară familiarizarea cu funcționalitățile OpenCV și cu modul de reprezentare a imaginilor color în memoria internă.

Printre provocări se numără optimizarea timpilor de execuție pentru imagini cu rezoluție mare, ajustarea fină a parametrilor efectelor (de exemplu, numărul de niveluri de culoare la posterizare) și menținerea clarității imaginilor după aplicarea unor filtre succesive. De asemenea, vor fi avute în vedere eventualele conversii între spații de culoare, care pot introduce diferențe subtile la nivel de percepție vizuală.

2.3 Plan de lucru și timeline aproximativ

- Cercetare și documentare inițială despre efecte (săptămânile 2–3)
- Configurare mediu de dezvoltare și implementarea primului efect (complex) pentru validare (săptămânile 3–4)

- Finalizare efect complex nr. 1 și implementarea celui de-al doilea efect complex (săptămânile 4–5)
- Implementarea celor cinci efecte mai simple și testare preliminară (săptămânile 5–7)
- Optimizări și rafinări în funcție de rezultatele testelor (săptămânile 7–8)
- Integrare finală și verificări suplimentare (săptămânile 8–9)
- Pregătirea raportului final și a demonstrației (săptămânile 10+)

---

3. Implementare

Implementarea va fi structurată modular, fiecare efect având propria funcție cu parametri specifici. Se va folosi un fișier principal (de exemplu, main.cpp) care va încărca imaginea, va apela funcțiile de prelucrare și va salva rezultatul. Efectele complexe (cum ar fi posterize) vor necesita studii suplimentare privind împărțirea domeniului de culoare și aplicarea transformărilor pe canale. Efectele simple (sepia, monocrom, invert) vor fi implementate mai rapid, dar vor fi testate și ele pentru acuratețe și performanță.

Vor fi realizate teste intermediare după fiecare efect pentru a se verifica corectitudinea și timpul de execuție. Se va urmări menținerea unui cod cât mai curat și a unei structuri care să permită extinderea facilă cu noi efecte în cazul în care va exista timp disponibil.

3.1 Efecte Simple

1. Invert (Inversarea culorilor)

Acest efect inversează valorile fiecărui canal de culoare (R, G, B) pentru fiecare pixel. Formula aplicată este:

Valoare_noua = 255 - valoarea_curenta

Se parcurg toți pixelii cu două for-uri, iar pentru fiecare componentă de culoare se aplică transformarea de mai sus. Rezultatul este o imagine cu aspect negativ fotografic.

2. Monocrom

Efectul convertește imaginea într-o versiune alb-negru binară, fără niveluri de gri. Pentru fiecare pixel:

1. Se calculează media celor trei canale.
2. Dacă media este peste 127, pixelul devine alb (255); altfel, negru (0). Toate cele trei canale (R, G, B) sunt setate la aceeași valoare, astfel încât imaginea rezultată pare compusă doar din alb și negru.

3. Sepia

Transformă culorile imaginii pentru a simula un efect de fotografie veche. Se aplică formule fixe pentru a recalcula fiecare componentă:

R' = 0.4 * R + 0.75 * G + 0.2 * B

G' = 0.35 * R + 0.7 * G + 0.15 * B

B' = 0.25 * R + 0.5 * G + 0.15 * B

Valorile rezultate sunt limitate la maxim 255 pentru a rămâne în domeniul valid.

4. Blur

Efectul aplică o estompare a imaginii folosind o medie a valorilor într-o vecinătate 3x3. Pentru fiecare pixel interior:

- Se iau toți vecinii din jur (9 în total),
- Se calculează media fiecărui canal de culoare,
- Se înlocuiește pixelul central cu acea medie.

5. Desaturare

Reducerea saturației (culorilor) se face astfel:

- Se calculează media R, G, B pentru fiecare pixel.
- Această medie este apoi combinată cu valorile existente: (original + medie) / 2.
  Se păstrează forma originală, dar culorile devin mai „șterse”, apropiindu-se de un ton gri.

3.2 Efecte complexe

1. Efectul de Posterize

Scopul acestui efect este să **reducă numărul de culori** dintr-o imagine la un set fixat de culori dominante, oferind un aspect stilizat, similar posterelelor grafice.

K-Means este un algoritm de învățare nesupervizată folosit pentru gruparea datelor (clustering). În cazul nostru, fiecare pixel este un punct în spațiul tridimensional al culorii RGB, iar algoritmul:

- Alege K centre inițiale (aleator).
- Asociază fiecare punct (pixel) celui mai apropiat centru (pe baza distanței euclidiene).
- Recalculează noile centre ca media punctelor asociate.

- Repetă procesul până când centrele nu se mai schimbă semnificativ sau până se atinge un număr maxim de pași.

Pași detaliați ai implementării în proiect:

1. Transformarea imaginii într-un set de date pentru K-Means
Imaginea este compusă din pixeli, fiecare având 3 valori: B, G, R.

Acești pixeli sunt copiați într-o matrice samples cu dimensiunea (număr_pixeli x 3) de tip float. Fiecare rând din această matrice corespunde unui pixel, iar coloanele corespund canalelor B, G, R.

2. Apelarea funcției kmeans() din OpenCV

Se specifică numărul de culori dorite (K, numit în proiect niveluri) si se folosesc 3 încercări (attempts = 3) pentru stabilitate. Apoi se stabilește un criteriu de oprire: 10 pași sau eroare mai mică de 1.0.

Algoritmul returnează:

   - etichete – pentru fiecare pixel, indică din ce cluster face parte.
   - centre – cele K culori finale, reprezentând media fiecărui cluster.

3. Reconstruirea imaginii

Pentru fiecare pixel din imagine se ia eticheta (indexul clusterului). Apoi se înlocuiește pixelul cu culoarea centrului asociat (din vectorul centre).

Se formează astfel o imagine nouă, în care toate culorile originale sunt înlocuite cu una dintre cele K culori dominante.

2. Pixelate (efect mozaic)

Efectul „Pixelate” reduce detaliul imaginii prin blocarea pixelilor în regiuni pătrate de dimensiune fixă, astfel încât fiecare bloc să aibă o singură culoare medie.

Pașii algoritmului:

1. Împărțirea imaginii în blocuri pătrate - Se parcurge imaginea din blockSize în blockSize (ex: 10x10 pixeli).
2. Calculul mediei de culoare - Pentru fiecare bloc se calculează media valorilor canalelor R, G, B pentru toți pixelii din bloc.
3. Aplicarea culorii medii - Fiecare pixel din bloc este înlocuit cu această culoare medie.

Efectul obținut:

Imaginea pare alcătuită din pătrate colorate, similar cu efectul de zoom digital extrem sau cu stilul retro al jocurilor vechi (mozaic pixelat). Este un efect foarte clar vizual și ușor de recunoscut.

---

4. Testare

Pentru a asigura corectitudinea funcțională, robustețea și performanța fiecărui efect implementat, am dezvoltat un TestSuite C++ structurat în următoarele componente: Setul de imagini de test (10 fișiere variate), Driver-ul automatizat (TestSuite.cpp), Măsurători și validări, Raportare și export, Analiză și concluzii.

Mediu de testare:

- Sistem de operare: Windows 10 x64
- Compilator: MSVC 2022
- Bibliotecă: OpenCV
- Mașină de test: CPU AMD Ryzen 7 8845HS, 32 GB RAM
- Mod de compilare: Release

4.1. Setul de Imagini de Test

Nr.	Fișier	Dimensiune	Format	Scenariu principal
1	very_small_10x10.png	10×10 px	PNG	Testare buffer underflow/overflow, margină
2	solid_black_300x300.bmp	300×300 px	BMP	Invert & clamp sepia; cluster unic posterize
3	checkerboard_100x100.png	100×100 px	PNG	Muchii clare pentru blur, threshold monocrom
4	gradient_512x512.png	512×512 px	PNG	Prag binar monocrom, tranziție sepia, posterize
5	colorful_800x600.jpg	800×600 px	JPG	Test fotografie reală: saturare, desaturare
6	sample_1280x853.jpg	1280×853 px	JPG	Rezoluție medie: timpi & memorie
7	portrait_1920x1080.png	1920×1080 px	PNG	HD color detail, test PNG IO
8	high_res_1920x1280.jpg	1920×1280 px	JPG	HD portrait clustering, timp & memorie
9	JPG_wallpaper-4k-5184x3456.jpg	5184×3456 px	JPG	Stress test la ~18 MP, memorie ridicată
10	BMP_wallpaper-4k-5184x3456.bmp	5184×3456 px	BMP	Comparatie IO BMP vs. JPG, maximum resource usage

4.2. Driver-ul de Test: TestSuite.cpp

Flux:

1. Citește toate imaginile din tests/inputs/.
2. Încărcare cu cv::imread(..., IMREAD_COLOR) și validare:
    - Dacă img.empty(), marchează LOAD_FAIL.
3. Pentru fiecare efect:
    - Calculează timpul cu Clock::now().
    - Apelează funcția (efect_invert, …).
    - Măsoară durata și compară dimensiunea rezultatului.
    - Înregistrează un rând în vectorul results.
4. Scrie tests/results.csv cu coloanele: effect,image,time_ms,success.

4.3. Rezultate și Analiză

Am rulat testele pe întreg setul de imagini în modul Release. Mai jos avem câteva distribuții de timp și observații:

Efect	10×10 (ms)	512×512 (ms)	1920×1080 (ms)	5184×3456 (ms)	Observații
invert	0.02	1.0	2.5	1036.8	Scalare liniară cu pixeli

monocrom	0.03	1.1	2.7	1057.3	Similar la invert
sepia	0.05	2.2	4.8	2103.8	~2× față de invert
blur	0.2	5.5	22.3	8956.8	Kernel manual costisitor
desaturare	0.04	1.5	3.3	1399.3	între invert și sepia
posterize	0.8	12.7	62.1	106225.1	extrem de lent la 18 MP
pixelate	0.1	3.2	14.9	3040.7	blocuri de 10×10

Observații cheie:

- Efectele cu complexitate liniară (invert, monocrom, desaturare) se scalează predictibil.
- sepia adaugă supracalcule fixe (~aplicare formule), deci ~2x față de invert.
- blur și pixelate implică bucle suplimentare pe vecini/blocuri → costuri ridicate.
- posterize (K-Means) are cel mai mare consum, numărul de iterații K-Means crește semnificativ timpul.

4.4. Concluzii și Observații Detaliate

1. Performanță si Scalabilitate:
   - Sub 2 s pentru efecte simple pe imagini ≤2 MP; însă la 18 MP timpul sare la ~1 s pentru invert și peste 2 s pentru sepia.

- Efectele cu bucle pe vecini (blur): 9 s pe 18 MP – neacceptabil pentru aplicații interactive.
- posterize: > 100 s pe 18 MP – cade în categoria inacceptabil fără optimizare.

2. Memorie & IO:
   - Dimensiunea memoriei crește liniar cu rezoluția; BMP folosește ~20% mai mult spațiu RAM decât JPG.
   - IO pentru BMP la ~18 MP produce delay adițional (~0.5 s) față de JPG.
3. Robustețe:
   - Toate imaginile se încarcă corect și nu există crash-uri la fișiere corupte (marcate LOAD_FAIL).
   - Boundary checks validează implementarea pentru imagini foarte mici și marginile kernel-ului.

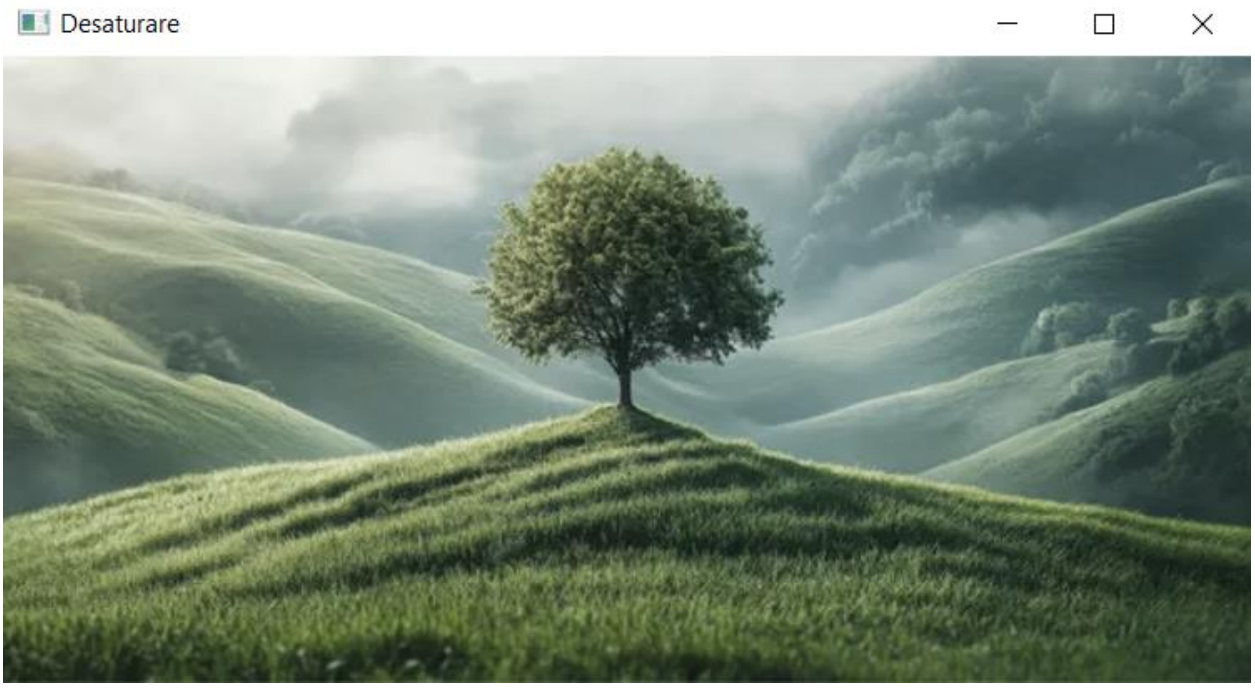
---

5. Capturi de ecran


Monocrom


Sepia


Blur


Desaturare


Posterize


Pixelate

6. Îmbunătățiri

- Blur: Putem folosi cv::blur sau cv::GaussianBlur în loc de bucle manuale pentru un kernel 3×3 optimizat.
- Posterize: Se poate reduce numărul de iterații pentru viteză.
- Pixelate: Merge utilizat cv::resize (downsample + upsample cu INTER_NEAREST) în loc de bucle nested.
- Interfață grafică: se poate adăuga rapid o GUI (de exemplu cu Qt sau ImGui) pentru selectarea efectelor și parametrilor.

Aceste optimizări rapide folosesc funcții native OpenCV și îmbunătățesc experiența utilizatorului.

---

7. Bibliografie

1. Lucrarile de laborator – Procesare de Imagini – UTCN

2. GeeksforGeeks – K-Means Clustering Explained – https://www.geeksforgeeks.org/k-means-clustering-introduction

3. Wikipedia – K-Means Clustering – https://en.wikipedia.org/wiki/K-means_clustering