Proiect la Programare Procedurală

Sava Daniel

În acest proiect, se lucrează cu imagini color în formatul BMP (bitmap), care vor fi manipulate în limbajul C ca fișiere binare. Acest format nu comprimă imaginile, ci stochează trei octeți per pixel pentru imaginile color.

Datorită modului în care este stocată imaginea, se va crea o structură ***Pixel***, care va conține cei trei octeți ai unui pixel, reprezentând cele trei canale de culoare (R – roșu, G – verde, B – albastru). Intensitatea fiecărui canal de culoare este dată de o valoare naturală cuprinsă între 0 și 255. Deoarece codarea unei imagini BMP respectă standardul little-endian, octeții corespunzători celor trei canale de culoare sunt memorați de la dreapta la stânga, adică în ordinea B, G, R.

Pentru o ușoară manipulare a imaginii, se va crea o structură ***Image***, care va memora date referitoare la header, conținut, lățime, înălțime și padding.

Pentru memorarea ferestrelor cu grad de corelație cu unul dintre șabloane cel puțin egal cu un prag dat, se va crea o structură ***Window***, care va memora linia și coloana colțului din stânga sus, înălțimea și lungimea ferestrei (care sunt aceleași cu cele ale șablonului), gradul de corelație și culoarea corespunzătoare.

Modulul de criptare/ decriptare

Funcția ***xorshift32*** generează numere întregi fără semn pe 32 de biți, cu un caracter pseudo-aleator (numere având proprietăți statistice asemănătoare celor ale unei secvențe de numere perfect aleatoare, adică o secvență de numere pentru care probabilitatea de apariție a unei anumite valori este independentă de toate valorile generate anterior), plecând de la o valoare inițială, numită seed. Se va folosi generatorul propus de George Marsaglia în 2003 (https://en.wikipedia.org/wiki/Xorshift).

- Primul parametru este numărul de numere ce se doresc a fi generate
- Al doilea parametru este seed-ul
- Funcția returnează un vector alocat dinamic cu (n + 1)(seed-ul + n numere) numere pseudo aleatoare

Funcția ***loadImageIntoMemory*** încarcă o imagine BMP în memoria internă în formă liniarizată.

- Funcția are ca parametru calea imaginii BMP
- Funcția returnează o structură ***Image***, care conține date despre imagine
  a) Se obține header-ul, care ocupă primii 54 de octeți ai fișierului, conține informații despre formatul BMP, precum și informații despre

dimensiunea imaginii, numărul de octeți utilizați pentru reprezentarea unui pixel etc.

b) Dimensiunea imaginii în pixeli este exprimată sub forma width×height, unde width reprezintă numărul de pixeli pe lățime (memorată pe patru octeți fără semn începând cu octetul al 18-lea din header), iar height reprezintă numărul de pixeli pe înălțime (memorată pe următorii 4 octeți fără semn, respectiv începând cu octetul al 22-lea din header)
c) Se calculează padding-ul; pentru rapiditatea procesării imaginilor la citire și scriere, imaginile în format BMP au proprietatea că fiecare linie are un număr de octeți care să fie multiplu de 4, lucru realizat prin adăugarea unor octeți de padding
d) Se citește conținutul imaginii începând cu colțul din stânga jos, datorită modului în care este stocată o imagine BMP, având grijă ca la finalul fiecărei linii să se sară peste octeții de padding

Funcția ***saveImageIntoFile*** salvează în memoria externă o imagine BMP stocată în formă liniarizată în memoria internă.

- Primul parametru este calea fișierului în care va fi salvată imaginea
- Al doilea parametru este structura care conține imaginea în formă liniarizată
  a) Se scrie header-ul
  b) Se scrie conținutul imaginii începând de la final, datorită modului în care este stocată o imagine BMP, și având grijă ca la finalul fiecărei linii să se scrie și eventualii octeții de padding

Funcția ***durstenfeld*** generează o permutare aleatoare de dimensiune dată cu ajutorul algoritmului lui Durstenfeld (https://en.wikipedia.org/wiki/Fisher%E2%80%93Yates_shuffle) și a numerelor pseudo-aleatoare generate anterior cu ajutorul funcției ***xorshift32***.

- Primul parametru este numărul de elemente al permutării
- Al doilea parametru este vectorul ce conține numerele pseudo-aleatoare
- Funcția returnează un vector alocat dinamic care conține permutarea obținută

Funcția ***shufflePixels*** permută pixelii unei imagini conform unei permutări date (în cazul de față, cea obținută în urma algoritmului lui Durstenfeld).

- Primul parametru este imaginea ai cărei pixeli se permută
- Al doilea parametru este vectorul ce conține permutarea

Funcția ***encryptImage*** aplică algoritmul de criptare descris în enunțul proiecutlui.

- Primul parametru este calea imaginii inițiale
- Al doilea parametru este calea imaginii criptate
- Al treilea parametru este calea unui fișier text care conține cheia secretă
  a) Se generează o secvență $R = R_1, R_2, \dots R_{2*width*height-1}$ de numere, folosind generatorul ***xorshift32*** inițializat cu valoarea nenulă R_0
  b) Se generează o permutare aleatoare σ de dimensiune width×height, folosind funcția ***dursenfeld***
  c) Se permută prin intermediul funcției ***shufflePixels*** pixelii imaginii inițiale (P) conform permutării σ, obținându-se o imagine intermediară (P'), folosind relația $P'_{\sigma(k)} = P_k$
  d) Imaginea criptată (C) se obține aplicând asupra fiecărui pixel al imaginii intermediare (P') următoarea relație de substituție:

$$C_k = \begin{cases} SV \oplus P_0' \oplus R_{width*height} & , k = 0 \\ C_{k-1} \oplus P_k' \oplus R_{width*height+k} & , k = 1, 2, \dots width*height - 1 \end{cases}$$

  - Cu $\oplus$ se notează operația sau-exclusiv (XOR) între 2 octeți
  - Pentru 2 pixeli $P_1 = (P_1^R, P_1^G, P_1^B)$ și $P_2 = (P_2^R, P_2^G, P_2^B)$, se notează cu $P_1 \oplus P_2$ pixelul $(P_1^R \oplus P_2^R, P_1^G \oplus P_2^G, P_1^B \oplus P_2^B)$
  - Pentru un pixel $P = (P^R, P^G, P^B)$ și un întreg fără semn X pe 32 de biți format din octeții (X_3, X_2, X_1, X_0), se va nota cu $P \oplus X$ pixelul $(P^R \oplus X_2, P^G \oplus X_1, P^B \oplus X_0)$. Pentru a accesa octeul i al lui X, se va folosi formula X >> (8 * i)) & 0xFF

Funcția ***inversePermutation*** calculează inversa unei permutări date.

- Primul parametru este numărul de elemente al permutării
- Al doilea parametru este permutarea
- Funcția returnează unn vector alocat dinamic ce conține inversa permutării date

Funcția ***decryptImage*** aplică algoritmul de decriptare descris în enunțul proiectului.

- Primul parametru este calea imaginii inițiale
- Al doilea parametru este calea imaginii criptate
- Al treilea parametru este calea unui fișier text care conține cheia secretă

a) Se generează o secvență $R = R_1, R_2, \dots R_{2*width*height-1}$ de numere, folosind generatorul ***xorshift32*** inițializat cu valoarea nenulă R_0
b) Se generează o permutare aleatoare σ de dimensiune width×height, folosind funcția ***dursenfeld*** și se calculează inversa sa, σ^{-1}

c) Se aplică asupra fiecărui pixel din imaginea criptată (C) inversa relației de substituție folosită în procesul de criptare:

$$C_k' = \begin{cases} SV \oplus C_0 \oplus R_{width*height} & , k = 0 \\ C_{k-1} \oplus C_k \oplus R_{width*height+k} & , k = 1, 2, \ldots width*height - 1 \end{cases}$$

d) Se permută prin intermediul funcției ***shufflePixels*** pixelii imaginii criptate (C') conform permutării σ^{-1}, folosing relația $D_{\sigma^{-1}(k)} = C_k'$

Funcția ***printChiSquareTest*** afișează valorile testului χ^2 pentru o imagine BMP pe fiecare canal de culoare (R, G, B), folosind formula dată în enunțul proectului.

- Funcția are ca parametru calea imaginii.

Modulul de recunoaștere de pattern-uri într-o imagine

La marginile imaginii, șablonul iese din imagine. Pentru a simplifica implementarea, se vor considera numai pozițiile din imagine pentru care șablonul încape în imagine.

Funcția ***grayscaleImage*** transformă o imagine color în formatul BMP în imagine grayscale, folosind formula R' = G' = B' = 0.299 * R + 0.587 * G + 0.114 * B, pentru fiecare pixel din imagine.

- Primul parametru este calea imaginii sursă, cea pe care se face grayscale.
- Al doilea parametru este calea fișierului în care va fi salvată imaginea grayscale.

Funcția ***calculateCorrelation*** calculează corelația, folosind formula dată în enunțul proiectului, dintre o imagine și un șablon, pentru fereastra ce începe la o linie și o coloană dată.

- Primul parametru este structura ce conține imaginea
- Al doilea parametru este structura ce conține șablonul
- Al treilea parametru este linia de la care se începe calcularea corelației
- Al patrulea parametru este coloana de la care se începe calcularea corelației

Funcția templateMatching returnează prin intermediul parametrilor toate ferestrele șabloanelor care au un grad de corelație cu imaginea cel puțin egal cu un prag dat

- Primul parametru este calea imaginii pe care se face template-matching
- Al doilea parametru este calea șablonului cu care se face template-matching

- Al treilea parametru este pragul pe care trebuie să îl aibă un șablon pentru a fi considerat detecție corectă
- Al patrulea parametru este vectorul de ferestre prin intermediul căruia se returnează toate ferestrele care au gradul de corelație cel puțin egal cu pragul dat
- Al cincilea parametru este numărul de elemente al vectorului de detecții

Funcția ***drawBorderWindow*** desenează conturul unei ferestre date cu o culoare dată.

- Primul parametru este structura care conține imaginea pe care se va desena
- Al doilea parametru este structura ce memorează date despre fereastră
- Al treilea parametru este o structură de tip ***Pixel*** ce va memora culoarea cu care se va desena conturul

Funcția ***initColorsForPixels*** returnează un vector de ***Pixel*** ce conține culorile cu care vor fi conturate cifrele. Culorile se găsesc în documentația proiectului.

Funcţia ***getAllMatches*** returnează prin intermediul parametrilor toate detecțiile care au un grad de corespondență cel puțin egal cu un prag dat

- Primul parametru este vectorul care va memora detecțiile
- Al doilea parametru este numărul de elemente al vectorului de detecții

a) Se încarcă imaginea pe care se face template matching și se transformă în grayscale
b) Se stabilește pragul de detecție
c) Se încarcă şabloanele unul câte unul şi se face template-maching cu fiecare, iar toate detecțiile se reunesc

Funcția ***intersection*** returnează o valoare de tip ***double***, reprezenând gradul de intersecție a două ferestre, folosind formula dată în documentația proiectului.

- Cei doi parametri sunt cele două ferestre.

a) Se stabilesc coordonatele colțului din stânga sus.
b) Se stabilesc coordonatele colțului din dreapta jos.
c) Se calculează gradul de intersecție.

Funcția ***compareByCorrelation*** reprezintă criteriul de comparare pentru funția qsort. În acest caz, aceasta ajută la sortarea descrescătoare a detecțiilor după corelație.

Funcția ***nonMaximalElimination*** elimină toate detecțiile care se suprapun, păstrându-le doar pe cele cu gradul de corelație cel mai mare.

- Primul parametru este vectorul care memorează detecțiile
- Al doilea parametru este numărul de detecții.

a) Pentru a nu muta elementele din vector de fiecare dată când trebuie eliminată o detecție, se va lua un vector auxiliar care va memora pentru fiecare element dacă a fost eliminat sau nu
b) La final, se elimină toate elementele marcate și se realocă memorie corespunzător.

Functia drawBorders desenează contururile ferestrelor

- Primul parametru este calea imaginii pe care se vor desena contururile
- Al doilea parametru este calea fișierului unde va fi salvată imaginea
- Al treilea parametru este vectorul care conține detecțiile
- Al patrulea parametru este numărul de detecții