Prezentarea temei de proiect
la disciplina Programare Procedurală
Seriile 13, 14 – noiembrie 2018

Ideea proiectului

Persoana A (emițător) îi trimite persoanei B (receptor) o imagine I pe care o criptează folosind un algoritm de criptare. B poate decripta imaginea criptată primită de la A obținând astfel imaginea inițială I.

Ideea proiectului

Persoana A (emițător) îi trimite persoanei B (receptor) o imagine I pe care o criptează folosind un algoritm de criptare. B poate decripta imaginea criptată primită de la A obținând astfel imaginea inițială I după care procesează imaginea (folosește un algoritm de recunoaștere de cifre scrise de mână în imaginea I).

Ideea proiectului

Persoana A (emițător) îi trimite persoanei B (receptor) o imagine I pe care o criptează folosind un algoritm de criptare. B poate decripta imaginea criptată primită de la A obținând astfel imaginea inițială I după care procesează imaginea (folosește un algoritm de recunoaștere de cifre scrise de mână în imaginea I). Persoana B trimite apoi persoanei A imaginea rezultată obținută pe care o criptează. Persoana A decriptează imaginea criptată primită de la B și poate vizualiza soluția, recunoașterea cifrelor scrise de mână.

Formatul BMP (bitmap)

Formatul BMP (bitmap)

- o imagine este un fișier binar
- imagine color RGB
- fiecare pixel are o culoare dată de un triplet (R,G,B) cu valori pe fiecare canal intre 0 si 255 = 1 octet/canal
- albastru = (0,0 ,255), negru = (0,0,0), rosu = (255, 0, 0), verde = (0,255,0)
- 1 pixel color = 3 canale = 3 octeți
- dimensiuni 500 x 400 pixeli (lățime x înălțime)
- 500 x 400 x 3 octeți = 600000 octeți

test.bmp

Formatul BMP (bitmap)

test.bmp

- 500 x 400 x 3 octeți = 600000 octeți + 54 octeți (header) = 600054 octeți

Formatul BMP (bitmap)

cifra0.bmp

- 11 x 15 x 3 octeți = 495 octeți + 54 octeți (header) = 549 octeți
  Se adaugă octeți de padding astfel încât fiecare linie e multiplu de 4

Formatul BMP (bitmap)

cifra0.bmp

- (11 x 3 + 3) x 15 = 540 octeți + 54 octeți (header) = 594 octeți
  Se adaugă 3 octeți de padding (de obicei au valoarea 0)

Formatul BMP (bitmap)

- formatul BMP cuprinde o zonă de date cu dimensiune fixă, numită *header* și o zonă de date cu dimensiune variabilă care conține pixelii imaginii propriu-zise.
- header-ul, care ocupă primii 54 de octeți ai fișierului, conține informații despre formatul BMP, precum și informații despre dimensiunea imaginii (octetul cu numărul 2), numărul de octeți utilizați pentru reprezentarea unui pixel etc.
- dimensiunea imaginii în pixeli este exprimată sub forma *W×H* (lățime × înălțime). Lățimea imaginii exprimată în pixeli este memorată pe patru octeți fără semn începând cu octetul al 18-lea din header, iar înălțimea este memorată pe următorii 4 octeți fără semn, respectiv începând cu octetul al 22- lea din header.

Modulul de criptare/decriptare

CRIPTAREA IMAGINILOR

Un **cifru simetric** este format din:

- un algoritm de criptare
- un algoritm de decriptare
- o cheie secretă comună

- Un **generator de numere pseudo-aleatoare** este un algoritm determinist care generează o secvență de numere având proprietăți statistice asemănătoare celor ale unei secvențe de numere perfect aleatoare plecând de la o valoare inițială (seed).

- Generatorul **XORSHIFT32** propus de George Marsaglia în 2003 generează numere întregi fără semn pe 32 de biți, cu un caracter pseudo-aleator foarte bun, folosind operații de deplasare pe biți și XOR.

```
unsigned int r, seed, k;

...........
r = seed;
for(k = 0; k < n; k++)
{
    r = r ^ r << 13;
    r = r ^ r >> 17;
    r = r ^ r << 5;
    printf("%u\n", r);
}
```

```
seed = 1000;

266172694
3204629577
385443340
2099747088
1321081938
733430728
3121244111
680290934
....................
```

- O **permutare aleatoare de lungime n** este o permutare selectată sau generată aleator din mulțimea celor $n!$ permutări de lungime n existente.
- O metodă eficientă de generare a unei permutări aleatoare p de lungime n este **algoritmul lui Durstenfeld** (1964):

```
unsigned int r, n, k, p[100];
...........
for(k = 0; k < n; k++)
    p[k] = k;

for(k = n-1; k >= 1; k--)
{
    r = random(0, k);
    aux = p[r];
    p[r] = p[k];
    p[k] = aux;
}
```

n = 6

k	r	p
–	–	1 2 3 4 5 6
5	3	1 2 3 **6** 5 **4**
4	1	1 **5** 3 6 **2** 4
3	1	1 **6** 3 **5** 2 4
2	2	1 6 **3** 5 2 4
1	0	**6** **1** 3 5 2 4

unde prin `random(0,k)` am notat un număr aleator cuprins între 0 și k.

- Fie $I = (I_{i,j})_{\substack{0 \le i < H \\ 0 \le j < W}}$ o imagine color cu lățimea de W pixeli și înălțimea de H pixeli în formă matriceală:

$$I = \begin{pmatrix} I_{0,0} & I_{0,1} & \dots & I_{0,W-1} \\ I_{1,0} & I_{1,1} & \dots & I_{1,W-1} \\ \dots & \dots & \dots & \dots \\ I_{H-1,0} & I_{H-1,1} & \dots & I_{H-1,W-1} \end{pmatrix}$$

Liniarizarea imaginii I presupune crearea unui tablou unidimensional L prin alipirea liniilor tabloului bidimensional I, de sus în jos:

$$L = \left(\underbrace{I_{0,0}, \dots, I_{0,W-1}}_{\text{Linia } 0}, \underbrace{I_{1,0}, \dots, I_{1,W-1}}_{\text{Linia } 1}, \dots, \underbrace{I_{H-1,0}, \dots, I_{H-1,W-1}}_{\text{Linia } H-1} \right) = (L_0, L_1, \dots, L_{W*H-1})$$

- Valoarea unui pixel L_k va fi un triplet de octeți fără semn $L_k = (L_k^R, L_k^G, L_k^B)$.
- Vom nota cu $\oplus$ operația sau-exclusiv (XOR) între 2 octeți fără semn.
- Pentru pixelii $P_1 = (P_1^R, P_1^G, P_1^B)$ și $P_2 = (P_2^R, P_2^G, P_2^B)$ vom nota 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 fără semn (X_3, X_2, X_1, X_0) vom nota cu $P \oplus X$ pixelul $(P^R \oplus X_2, P^G \oplus X_1, P^B \oplus X_0)$.

CIFRUL SIMETRIC PROPUS

- **Cheia secretă comună** este compusă din valorile R_0 și SV, ambele numere întregi nenule fără semn pe 32 de biți.

- Pentru o imagine color în clar P de dimensiune $W \times H$ în formă liniară $P = (P_0, P_1, \dots, P_{W*H-1})$ și cheia secretă (R_0, SV) **algoritmul de criptare** este următorul:

  a) se generează o secvență $R = (R_1, R_2, \dots, R_{2*W*H-1})$ de numere întregi aleatoare fără semn pe 32 de biți, folosind generatorul XORSHIFT32 inițializat cu o valoare nenulă R_0;

  b) se generează o permutare aleatoare σ de dimensiune $W \times H$, folosind algoritmul lui Durstenfeld și numerele pseudo-aleatoare $R_1, \dots, R_{W*H-1}$;

  c) se permută pixelii imaginii P conform permutării σ, obținându-se o imagine intermediară P', respectiv $P'_{\sigma(k)} = P_k$ pentru orice $k \in \{0,1, \dots, W * H - 1\}$;

d) imaginea criptată $C = (C_0, C_1, \dots, C_{W*H-1})$ se obține aplicând asupra fiecărui pixel al imaginii $P' = (P_0', P_1', \dots, P_{W*H-1}')$ următoarea relație de substituție:

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

- Cifrul fiind unul simetric, **algoritmul de decriptare** este complementar celui de criptare.
- Astfel, algoritmul de decriptare al unei imagini color criptate C de dimensiune $W \times H$ în formă liniară $C = (C_0, C_1, \dots, C_{W*H-1})$ și cheia secretă (R_0, SV) este următorul:

a) se generează o secvență $R = (R_1, R_2, \dots, R_{2*W*H-1})$ de numere întregi aleatoare fără semn pe 32 de biți, folosind generatorul XORSHIFT32 inițializat cu valoarea nenulă R_0;

b) se generează permutarea aleatoare σ de dimensiune $W \times H$, folosind algoritmul lui Durstenfeld și numerele pseudo-aleatoare $R_1, \dots, R_{W*H-1}$, după care se calculează inversa sa σ^{-1};

c) se aplică asupra fiecărui pixel din imaginea criptată $C = (C_0, \dots, C_{W*H-1})$ inversa relației de substituție folosită în procesul de criptare, obținându-se o imagine intermediară $C' = (C'_0, C'_1, \dots, C'_{W*H-1})$:

$$C'_k = \begin{cases} SV \oplus C_0 \oplus R_{W*H}, & k = 0 \\ C_{k-1} \oplus C_k \oplus R_{W*H+k}, & k \in \{1,2, \dots, W * H - 1\} \end{cases}$$

d) imaginea decriptată $D = (D_0, D_1, \dots, D_{W*H-1})$ se obține permutând pixelii imaginii C' conform permutării σ^{-1}, respectiv $D_{\sigma^{-1}(k)} = C'_k$ pentru orice $k \in \{0,1, \dots, W * H - 1\}$.

ANALIZA PERFORMANȚELOR CIFRULUI PROPUS

- Asupra unei imagini criptate, o persoană neautorizată (un atacator care nu cunoaște cheia secretă) poate efectua mai multe tipuri de atacuri pentru a afla fie cheia secretă, fie imaginea în clar (în multe cazuri, chiar și determinarea doar a unei părți dintr-o imagine poate furniza informații foarte importante atacatorului).
- Unul dintre cele mai simple atacuri îl constituie **atacul în frecvență**, prin care atacatorul mai întâi va determina, pe fiecare canal de culoare, frecvențele valorilor pixelilor, după care va încerca să asocieze valorile lor, în ordinea descrescătoare a frecvențelor, cu valorile pe care el le consideră ca fiind cele mai probabile.
- Pentru a rezista în faţa unor astfel de atacuri de tip statistic, un cifru trebuie să producă, indiferent de cheia secretă utilizată, imagini criptate cu o distribuţie uniformă a valorilor pixelilor din fiecare canal de culoare, pentru a ascunde astfel distribuţia neuniformă a valorilor pixelilor din imaginea iniţială.
- Instrumentul de analiză vizuală cel mai des utilizat pentru a studia distribuţia valorilor pixelilor unei imagini color îl constituie *histograma culorilor*, în care sunt reprezentate grafic frecvenţele valorilor pixelilor unei imagini, separat pentru fiecare canal de culoare.

ANALIZA PERFORMANȚELOR CIFRULUI PROPUS

Imaginea în clar

Imaginea criptată

Histograma imaginii în clar

Histograma imaginii criptate

ANALIZA PERFORMANȚELOR CIFRULUI PROPUS

- Analiza vizuală a uniformităţii distribuţiei valorilor pixelilor folosind histograma culorilor prezintă mai multe dezavantaje, unul dintre ele fiind faptul că precizia evaluării depinde într-un mod decisiv de acuitatea vizuală a evaluatorului uman şi de calitatea grafică a histogramei.
- Dezavantajele analizei vizuale pot fi eliminate prin utilizarea unui instrument statistic de evaluare a uniformităţii distribuţiei valorilor dintr-un şir, respectiv testul χ^2 (**testul chi-pătrat**).
- Valoarea testului χ^2 corespunzătoare unei imagini de dimensiune $m \times n$ pixeli se calculează folosind formula următoare:

$$\chi^2 = \sum_{i=0}^{255} \frac{(f_i - \bar{f})^2}{\bar{f}}$$

unde f_i este frecvenţa valorii i $(0 \leq i \leq 255)$ pe un canal de culoare al imaginii, iar $\bar{f}$ este frecvenţa estimată teoretic a oricărei valori i , respectiv $\bar{f} = \frac{m \times n}{256}$.

ANALIZA PERFORMANȚELOR CIFRULUI PROPUS

- O valoare a testului mai mică sau egală cu valoarea teoretică de prag $\chi_0^2 = 293.25$ indică o histogramă uniformă pe canalul de culoare respectiv, în timp ce o valoare mai mare indică o histogramă neuniformă.
- Pentru cele două imagini de mai sus, valorile testului χ^2 pe canalele de culoare RGB sunt următoarele:
  - pentru imaginea în clar: $(211104.79, 348462.19, 197839.49)$
  - pentru imaginea criptată: $(264.48, 251.41, 330.49)$

CERINȚE

1) Implementați generatorul de numere pseudo-aleatoare XORSHIFT32 sub forma unei funcții sau a unei macrodefiniții.

2) Scrieți o funcție care să încarce o imagine BMP în memoria internă în formă liniarizată. Funcția va avea ca parametru calea imaginii BMP.

3) Scrieți o funcție care să salveze în memoria externă o imagine BMP stocată în formă liniarizată în memoria internă. Funcția va avea ca parametru calea imaginii BMP.

4) Scrieți o funcție care să cripteze o imagine BMP folosind algoritmul de criptare prezentat. Funcția va avea ca parametri calea imaginii inițiale, calea imaginii criptate și calea unui fișier text care conține cheia secretă.

5) Scrieți o funcție care să decripteze o imagine BMP criptată folosind algoritmul de decriptare prezentat. Funcția va avea ca parametri calea imaginii inițiale, calea imaginii criptate și calea unui fișier text care conține cheia secretă.

6) Scrieți o funcție care să afișeze valorile testului χ^2 pentru fiecare canal de culoare al unei imagini BMP. Funcția va avea ca parametru calea imaginii.

7) Scrieți un program care să realizeze următoarele operații:

- criptarea unei imagini color BMP și salvarea imaginii criptate în memoria externă (căile ambelor imagini și a fișierului text care conține cheia secretă se vor citi de la tastatură);
- decriptarea unei imagini color BMP criptate și salvarea imaginii decriptate în memoria externă (căile ambelor imagini și a fișierului text care conține cheia secretă se vor citi de la tastatură);
- afișarea pe ecran a valorilor testului χ^2 pentru imaginea inițială și imaginii criptată, pe fiecare canal de culoare.

Modulul de recunoaștere de pattern-uri (cifre scrise de mână)

Template matching

- metoda de *template matching* constă în găsirea de părți mici ale unei imagini care se potrivesc cu o imagine șablon.

test.bmp

- pentru problema recunoașterii cifrelor scrise de mână vom folosi pe post de șabloane imagini cu 10 cifre.

cifra0.bmp
cifra1.bmp
...
cifra9.bmp

Template matching

Algoritmul de template matching aplicat pentru o imagine color I și un șablon S (o imagine color ce reprezintă o cifră) funcționează astfel:

a) se transformă imaginea color I și șablonul S în imagini grayscale. Un punct de pornire este sursa *grayscale.c* din arhivă;

b) se glisează șablonul curent pe imaginea I centrând șablonul în fiecare punct al imaginii I. Se calculează pentru fiecare poziție (x, y) din imaginea I corelația (https://en.wikipedia.org/wiki/Cross-correlation) dintre șablonul curent și conținutul corespunzător al imaginii dat de fereastra $f_I = f_I(x, y)$ centrată în punctul (x, y). Formula de calcul a corelației dintre șablonul S și fereastra f_I este următoarea:

Template matching

Algoritmul de template matching aplicat pentru o imagine color I și un șablon S (o imagine color ce reprezintă o cifră) funcționează astfel:

a) se transformă imaginea color I și șablonul S în imagini grayscale. Un punct de pornire este sursa *grayscale.c* din arhivă;

b) se glisează șablonul curent pe imaginea I centrând șablonul în fiecare punct al imaginii I. Se calculează pentru fiecare poziție (x, y) din imaginea I corelația (https://en.wikipedia.org/wiki/Cross-correlation) dintre șablonul curent și conținutul corespunzător al imaginii dat de fereastra $f_I = f_I(x, y)$ centrată în punctul (x, y). Formula de calcul a corelației dintre șablonul S și fereastra f_I este următoarea:

Calculul corelației

Formula de calcul a corelației dintre șablonul S și fereastra f_I este următoarea:

$$corr(S, f_I) = \frac{1}{n} \sum_{(i,j)\in S} \frac{1}{\sigma_{f_I}\sigma_S} \left(f_I(i,j) - \bar{f}_I\right) \left(S(i,j) - \bar{S}\right), unde$$

- n reprezintă numărul de pixeli în șablonul S (în particular pentru șabloanele utilizate de dimensiuni 11 × 15 pixeli avem n= 11 * 15 = 165);
- indicii i și j reprezintă linia i și coloana j în șablonul S (15 linii și 11 coloane);
- $S(i,j)$ reprezintă valoarea intensității grayscale a pixelului de la linia i și coloana j în șablonul S. Pentru o imagine grayscale, un pixel $P = (P^R, P^G, P^B)$ este reprezentat de un triplet cu toate valorile egale $P^R = P^R = P^B$. În acest caz valoarea intensității grayscale a lui P este P^R;
- $\bar{S}$ reprezintă media valorilor intensităților grayscale a pixelilor în fereastra S (media celor 165 de pixeli din șablonul S);
- σ_S reprezintă deviația standard a valorilor intensităților grayscale a pixelilor în șablonul S: $\sigma_S = \sqrt{\frac{1}{n-1}\sum_{(i,j)\in S}(S(i,j) - \bar{S})^2}$
- $f_I(i,j)$ reprezintă valoarea intensității grayscale a pixelului de la linia i și coloana j în fereastra f_I.
- $\bar{f}_I$ reprezintă media valorilor intensităților grayscale a pixelilor din fereastra f_I (media celor 165 de pixeli din fereastra f_I);
- σ_{f_I} reprezintă deviația standard a valorilor intensităților grayscale a pixelilor în fereastra f_I: $\sigma_{f_I} = \sqrt{\frac{1}{n-1}\sum_{(i,j)\in f_I}(f_I(i,j) - \bar{f}_I)^2}$

Ferestre cu corelație mai mare decât un prag

Cifra 0, prag mic = 0.35 –> multe detecții

Ferestre cu corelație mai mare decât un prag

Cifra 0, prag mare = 0.70 –> puține detecții

Ferestre cu corelație mai mare decât un prag

Cifra 5, prag mic = 0.35 –> multe detecții

Ferestre cu corelație mai mare decât un prag

Cifra 5, prag mare = 0.70 –> puține detecții

Ferestre cu corelație mai mare decât un prag

Cifrele 0-9, prag = 0.50 –> multe detecții suprapuse

Ferestre cu corelație mai mare decât un prag

Cifrele 0-9, prag = 0.50 –> multe detecții suprapuse

Pentru detecțiile care se suprapun o păstrez pe cea cu scorul mai mare

Suprimarea non-maximelor

În practică, pentru evitarea acestui fenomen se folosește o tehnică numită *eliminarea non-maximelor*. Ea constă în următoarele:

e) se pun într-un tablou unidimensional D toate detecțiile obținute pentru toate cele 10 șabloane;

f) se sortează tabloul D descrescător în funcție de scorul de corelație al fiecărei detecții.

g) se procesează tabloul D sortat de la stânga (detecții cu scor foarte mare) la dreapta (detecții cu scor foarte mic) astfel: toate detecțiile d_j care se suprapun spațial cu detecția curentă d_i, cu i < j și deci și scor(d_i) > scor(d_j) se elimină.

$$suprapunere(d_i, d_j) = \frac{aria(d_i \cap d_j)}{aria(d_i \cup d_j)} = \frac{aria(d_i \cap d_j)}{aria(d_i) + aria(d_j) - aria(d_i \cap d_j)}$$

În acest proiect vom considera că două ferestre se suprapun dacă suprapunerea lor spațială este > 0.2.

Cerințe:

7) Scrieți o funcție care să implementeze operația de template matching dintre o imagine I și un șablon S. Funcția va avea ca parametri imaginea I, șablonul S și pragul p_S și furnizează f_I care au corelația mai mare decât pragul p_S. (**1.5 puncte**)
8) Scrieți o funcție care primește ca parametru o imagine I, o fereastră f_I și o culoare $C = (C^R, C^G, C^B)$ și desenează conturul ferestrei f_I în imaginea I folosind culoarea C (ca în exemplele de mai sus). (**1 punct**)
9) Folosind funcția qsort din librăria stdlib.h scrieți o funcție care sortează un tablou D de detecții în ordinea descrescătoare a corelațiilor detecțiilor. (**0.25 puncte**)
10) Scrieți o funcție care implementează algoritmul de eliminare a non-maximelor. (**1.5 puncte**)

11) Scrieţi un program care să realizeze următoarele operaţii: (0.5 puncte)

- criptează o imagine color BMP şi salvează imaginea criptate în memoria externă (căile ambelor imagini şi a fişierului text care conţine cheia secretă se vor citi de la tastatură sau dintr-un fişier text);
- decriptează o imagine color BMP criptată şi salvează imaginea decriptată în memoria externă (căile ambelor imagini şi a fişierului text care conţine cheia secretă se vor citi de la tastatură sau dintr-un fişier text);
- afişează pe ecran valorile testului χ^2 pentru imaginea iniţială şi imaginea criptată, pe fiecare canal de culoare.
- rulează operaţia de template matching pentru o imagine color BMP şi o colecţie de şabloane color BMP. Folosiţi întotdeauna un prag $p_S = 0.5$ indiferent de şablonul ales. Reuniţi toate detecţiile rezultate pentru fiecare şabloan în parte într-un singur tablou unidimensional D. Căile imaginii şi a şabloanelor BMP se vor citi de la tastatură sau dintr-un fişier text.
- rulaţi funcţia de eliminare a non-maximelor pe tabloul D şi desenaţi în imagine detecţiile rămase folosind o culoare specifică pentru fiecare şablon.

Predarea și notarea proiectului

Termen limită. Termenul limită de predare al proiectului este vineri, 4 ianuarie 2019, ora 23:59. Fiecare zi de întârziere se penalizează cu un punct în minus la nota finală a proiectului, dar nu mai mult de 3 zile.

Predarea proiectului. Proiectul va fi predat sub forma unei arhive zip și trimis pe email laborantului de grupă. Ca excepție, studenții restanțieri vor preda proiectul lui Bogdan (adresa de email: bogdan.alexe@fmi.unibuc.ro). Arhiva zip va conține: (1) fișierele surse în limbajul C folosite pentru rezolvarea proiectului; (2) un document PDF în care sunt explicate funcțiile folosite. Proiectele vor fi prezentate în cadrul celor două laboratoare din luna ianuarie.

Regulament de etică

Tema de proiect este individuală. Prezentarea care însoțește proiectul (expusă la curs) detaliază toți pașii implementării. Toate întrebările legate de proiect le adresați laboranților sau celor care țin cursul. Fișierele sursă primite vor fi verificate cu un instrument care detectează cod asemănător. Proiectele care seamănă între ele vor fi notate cu nota 1.

NOTĂ:

1. În rezolvarea temei de proiect nu este permisă utilizarea unor variabile globale.
2. Pentru sortarea tablourilor se va utiliza funcţia qsort din biblioteca stdlib.h.
3. Tema trebuie rezolvată utilizând strict limbajul C standard, ci nu limbajul C++!
4. Toate tablourile folosite în rezolvarea temei trebuie să fie alocate dinamic.
5. Rezolvările corecte care nu respectă restricţiile indicate (rezolvarea unor cerinţe fără a folosi funcţii, utilizarea unor variabile globale, neutilizarea funcţiei qsort pentru sortarea unui tablou etc.) vor primi punctaje parţiale.
6. Se acordă 1 punct din oficiu.