

Temă Proiect la Programare Procedurală

Seriile 13, 14 - noiembrie 2018

Tema proiectului

Criptografia și procesarea digitală a imaginilor sunt două domenii foarte importante ale matematicii și informaticii. Tema proiectului combină două probleme din aceste subdomenii și anume criptarea și decriptarea unui mesaj (particularizat în acest proiect la o imagine) și recunoașterea unor pattern-uri (particularizate în acest proiect la cifre scrise de mână) într-o imagine.

Scenariul de implementat în acest proiect este următorul: persoana A îi trimite persoanei B 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 folosește un algoritm de recunoaștere a cifrelor 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. Tema proiectului constă în implementarea modulelor de criptare/decriptare și de recunoaștere de cifre scrise de mână și apoi integrarea lor într-un program final.

În acest proiect veți lucra cu imagini color, pe care le puteți manipula în limbajul C ca fișiere binare. Imaginele sunt în formatul BMP (bitmap). Spre deosebire de alte formate (JPG, JPEG, PNG, etc.) formatul BMP nu comprimă imaginile, ci stochează 3 octeți per pixel pentru imaginile color. Această caracteristică face formatul BMP adecvat acestui proiect întrucât puteți avea acces în mod explicit la valorile intensităților pixelilor care alcătuiesc imaginea color. Pentru vizualizarea imaginilor pe calculatorul personal trebuie să aveți instalat un program specific (IrfanView, Paint, Gimp, Preview, ImageJ etc.)

Materialele pentru proiect se găsesc în arhiva atașată. În arhivă se găsesc:

- directorul *cod* care conține două fișiere sursă *grayscale.c* și *verificator_criptare.c*. Fișierul sursă *grayscale.c* vă ajută să transformați o imagine color într-o imagine grayscale (în tonuri de gri). Fișierul sursă *verificator_criptare.c* vă ajută să vă verificați că ați realizat criptarea corectă a imaginii *peppers.bmp* furnizate.
- directorul *date* cu subdirectoarele *criptografie* și *recunoasterePatternuri*. Sudirectorul *criptografie* conține imaginea *enc_peppers_ok.bmp* care reprezintă rezultatul criptării imaginii *peppers.bmp* cu cheia secretă dată de fișierul *secret_key.txt*. Fișierul *chi-squared-test-values.txt* conține statistici despre imaginea inițială *peppers.bmp* și cea criptată *enc_peppers_ok.bmp*. Subdirectorul *recunoasterePatternuri* conține imaginile cu care o să lucrăți:

imaginea cu cifre scrise de mâna *test.bmp*, imaginile cu şabloanele pentru cifrele 0-9 *cifra0.bmp*, ..., *cifra9.bmp*.

În cele ce urmează detaliem pe larg proiectul. Prezentăm inițial formatul BMP (bitmap) astfel încât să fie foarte clar cum se reprezintă o imagine în acest format. Prezentăm apoi cele două module de implementat: modulul de criptografie care se ocupă de problema criptării/decriptării unui mesaj și modulul de recunoaștere de pattern-uri care se ocupă de recunoașterea de cifre scrise de mâna într-o imagine folosind metoda de template matching.

Formatul BMP

Formatul BMP (bitmap) este un format de fișier folosit pentru a stoca imagini digitale bidimensionale având lățime, înălțime și rezoluție arbitrară, monocrome sau color. În acest proiect veți lucra numai cu imagini color. Practic, în formatul BMP, imaginea este văzută ca un tablou de pixeli, iar fiecare pixel este o valoare întreagă. Primii 3 octeți din reprezentarea unui pixel reprezintă intensitatea celor 3 canale de culoare R (Red), G(green), B(blue). În consecință, intensitatea fiecărui canal de culoare R, G, B este dată de o valoare naturală cuprinsă între 0 și 255. De exemplu, un pixel cu valorile (0, 0, 0) reprezintă un pixel de culoare neagră, iar un pixel cu valorile (255, 255, 255) reprezintă un pixel de culoare albă. La adresa https://www.rapidtables.com/web/color/RGB_Color.html puteți găsi corespondența dintre triplete RGB și culori.

Formatul BMP este descris aici: https://en.wikipedia.org/wiki/BMP_file_format într-o manieră exhaustivă. 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, numărul de octeți utilizati pentru reprezentarea unui pixel etc.

Dimensiunea imaginii în octeți este specificată în header printr-o valoare întreagă fără semn, deci memorată pe 4 octeți, începând cu octetul cu numărul de ordine 2. Dimensiunea imaginii în pixeli este exprimată sub forma $W \times H$, unde W reprezintă numărul de pixeli pe lățime, iar H reprezintă numărul de pixeli pe 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.

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. Se realizează acest lucru prin adăugarea unor *octeți de padding* astfel încât

numărul total de octeți de pe o linie să devină multiplu de 4. Numărul de octeți al unui linii este $3 \times latimea$ (câte 3 octeți pentru fiecare pixel de pe o linie). Astfel, dacă o imagine are 11 pixeli în lățime (cum este cazul şabloanelor cu care veți lucra) numărul de octeți de padding este 3 ($3 \times 11 = 33$ octeți pe o linie, deci se vor adăuga la sfârșitul fiecărei linii câte 3 octeți de padding astfel încât să avem $33 + 3 = 36$ multiplu de 4 octeți). De obicei, octetii de padding adăugați au valoarea 0. Fișierul sursă *grayscale.c* vă exemplifică acest lucru.

Deoarece codarea unei imagini BMP într-un fișier binar respectă standardul little-endian, octetii corespunzători celor 3 canale de culoare R, G, B sunt memorati de la dreapta la stânga, adică în ordinea B, G, R.

O imagine color poate fi transformată (Figura 1) într-o imagine în tonuri de gri (grayscale) înlocuind valorile (R, G, B) ale fiecărui pixel cu valorile (R', G', B') definite astfel:

$$R' = G' = B' = [0.299 * R + 0.587 * G + 0.114 * B]$$

unde prin $[x]$ am notat partea întreagă a numărului real x . Imaginea grayscale obținută (folosind funcția din arhivă *grayscale.c*) are proprietatea că utilizează tot 3 octeți pentru reprezentarea culorii fiecărui pixel, toate valorile de pe cele 3 canale fiind egale.

A 2D grid of handwritten digits from 0 to 9 in black on a white background. The digits are written in a cursive style and are of varying sizes and orientations. The grid is approximately 10 columns by 10 rows.

A 2D grid of handwritten digits from 0 to 9 in grayscale on a white background. The digits are represented as grayscale values where darker shades correspond to higher pixel intensities. The grid is approximately 10 columns by 10 rows.

Figura 1. Stânga: imagine color cu cifre scrise de mâna. Dreapta: imagine grayscale cu cifre scrise de mâna.

Modulul de criptare/decriptare

Criptografia este o ramură a matematicii care se ocupă de transmiterea securizată a informației. Unul dintre dezideratele criptografiei îl reprezintă asigurarea *confidențialității datelor*, astfel încât o anumită informație să poată fi accesată doar de către persoanele autorizate. Pentru realizarea acestui deziderat se utilizează un proces de *criptare*, prin care o informație inteligibilă (un text, o imagine, un fișier audio etc.) este transformată într-o neinteligibilă.

Transformarea inversă, prin care o informație criptată este readusă la forma inițială (inteligibilă) se realizează prin intermediul unui proces de *decriptare*.

Un *cifru simetric* constă dintr-o pereche de algoritmi prin intermediul cărora se asigură procesele de criptare și decriptare (Figura 2), precum și o *cheie secretă comună* (cunoscută doar de persoanele care doresc să comunice între ele într-un mod sigur) care controlează cele două procese.

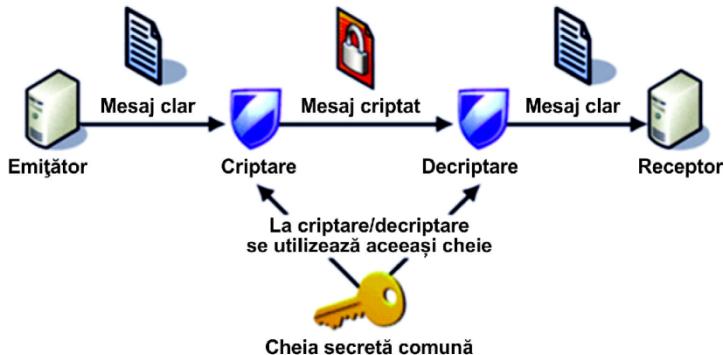


Figura 2. Criptarea și decriptarea folosind un cifru simetric.

Practic, cheia secretă comună reprezintă o dată de intrare pentru algoritmii de criptare și de decriptare, deci necunoașterea valorii sale exacte va conduce la imposibilitatea decriptării mesajului respectiv de către o persoană neautorizată.

În continuare, vom descrie un cifru simetric simplu care poate fi utilizat pentru asigurarea confidențialității unei imagini. Mai întâi, vom prezenta câteva notații:

- 1) fie $I = (I_{i,j})_{\substack{0 \leq i < H \\ 0 \leq j < W}}$ 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:

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

- 2) considerând o imagine color I în formă liniară, valoarea fiecărui pixel I_k va fi un triplet de octeți fără semn $I_k = (I_k^R, I_k^G, I_k^B)$;
- 3) vom nota cu \oplus operația sau-exclusiv (XOR) între 2 octeți fără semn;
- 4) pentru 2 pixeli $P_1 = (P_1^R, P_1^G, P_1^B)$ și $P_2 = (P_2^R, P_2^G, P_2^B)$ vom nota cu $P_1 \oplus P_2$ pixelul $(P_1^R \oplus P_2^R, P_1^G \oplus P_2^G, P_1^B \oplus P_2^B)$;
- 5) pentru un pixel $P = (P^R, P^G, P^B)$ și un întreg fără semn X pe 32 de biți format din octetii fără semn (X_3, X_2, X_1, X_0) vom nota cu $P \oplus X$ pixelul

$(P^R \oplus X_2, P^G \oplus X_1, P^B \oplus X_0)$, deci octetul cel mai semnificativ X_3 al lui X nu va fi utilizat;

- 6) 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 (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 (*seed*). De exemplu, 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 (<https://en.wikipedia.org/wiki/Xorshift>).

Folosind notațiile de mai sus, algoritmul de criptare al unei imagini color P (*plain image*) de dimensiune $W \times H$ în formă liniară $P = (P_0, P_1, \dots, P_{W \times H - 1})$ folosind o cheie secretă S este următorul:

- se generează o secvență $R = (R_1, R_2, \dots, R_{2 \times W \times H - 1})$ de numere întregi aleatoare fără semn pe 32 de biți, folosind generatorul XORSHIFT32 inițializat cu o valoare nenulă R_0 ;
- se generează o permutare aleatoare σ de dimensiune $W \times H$, folosind algoritmul lui Durstenfeld – varianta modernă a algoritmului Fisher-Yates (https://en.wikipedia.org/wiki/Fisher%20%20Yates_shuffle) și numerele pseudo-aleatoare $R_1, \dots, R_{W \times H - 1}$;
- se permute pixelii imaginii P conform permutării σ , obținându-se o imagine intermediară P' . De exemplu, considerând imaginea $P = (P_0, P_1, P_2, P_3)$ și permutarea $\sigma = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 3 & 0 & 2 & 1 \end{pmatrix}$ se va obține imaginea $P' = (P_1, P_3, P_2, P_0)$, deoarece o pereche $(k, \sigma(k))$ indică faptul că pixelul aflat pe poziția k în imaginea P va fi mutat pe poziția $\sigma(k)$ în imaginea P' , adică $P'_{\sigma(k)} = P_k$ pentru orice $k \in \{0, 1, \dots, W \times H - 1\}$;
- imaginea criptată $C = (C_0, C_1, \dots, C_{W \times H - 1})$ (*ciphered image*) se obține aplicând asupra fiecărui pixel al imaginii $P' = (P'_0, P'_1, \dots, P'_{W \times H - 1})$ următoarea relație de substituție:

$$C_k = \begin{cases} SV \oplus P'_0 \oplus R_{W \times H}, & k = 0 \\ C_{k-1} \oplus P'_k \oplus R_{W \times H + k}, & k \in \{1, 2, \dots, W \times H - 1\} \end{cases}$$

unde SV (*starting value*) este un număr întreg nenul fără semn pe 32 de biți.

Cheia secretă comună a acestui cifru este compusă din valorile R_0 și SV , ambele numere întregi neneule fără semn pe 32 de biți. Pentru a asigura un nivel ridicat de securitate a acestui cifru, cheia secretă comună trebuie schimbată înaintea fiecărei utilizări a cifrului cu una care nu a mai fost utilizată până atunci!

Cifrul fiind unul simetric, algoritmul de decriptare este complementar celui de criptare. Astfel, algoritmul de decriptare al unei imagini color criptate C (*ciphered image*) de dimensiune $W \times H$ în formă liniară $C = (C_0, C_1, \dots, C_{W \times H - 1})$ folosind o cheie secretă S 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 $C = (C_0, C_1, \dots, C_{W*H-1})$ 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 $D = (D_0, D_1, \dots, D_{W*H-1})$ (*deciphered image*) se obține permutând pixelii imaginii C' conform permutării σ^{-1} , respectiv $D_{\sigma^{-1}(k)} = C'_k$ pentru orice $k \in \{0, 1, \dots, W * H - 1\}$.

Corectitudinea algoritmului de decriptare se demonstrează foarte ușor, folosind proprietățile operației XOR!

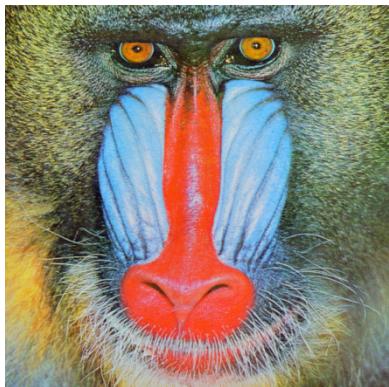
Asupra unei imagini criptate, o persoană neautorizată (un atacator care nu cunoaște cheia secretă) poate efectua mai multe tipuri de atacuri pentru a afla fie cheia secretă, fie imaginea în clar (în multe cazuri, chiar și determinarea doar a unei părți dintr-o imagine poate furniza informații foarte 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. De exemplu, dacă atacatorul intuiște faptul că imaginea în clar conține mai multe nave militare aflate în misiune, atunci el poate presupune faptul că majoritatea pixelilor sunt albaștri (sau nuanțe de albastru), deci au valori RGB de forma (0, 0, 255). În consecință, el va înlocui în toți pixelii valorile cu frecvența maximă de pe canalele de culoare R și G cu 0, iar valoarea cu frecvență maximă de pe canalul de culoare B o va înlocui cu 255. După aceea, poate presupune că restul pixelilor reprezintă navele militare, deci pixelii respectivi pot fi albi (sau o altă culoare specifică navelor de război) și va relua procedeul anterior. Chiar dacă în acest mod nu va reuși să reconstituie imaginea inițială, este posibil să reconstituie parțial contururile navelor, deci va afla câte nave sunt imagine – o informație foarte importantă!

Pentru a rezista în fața unor astfel de atacuri de tip statistic, un cifru trebuie să producă, indiferent de cheia secretă utilizată, imagini criptate cu o distribuție uniformă a valorilor pixelilor din fiecare canal de culoare, pentru a ascunde astfel distribuția neuniformă a valorilor pixelilor din imaginea inițială care poate furniza informații statistice relevante.

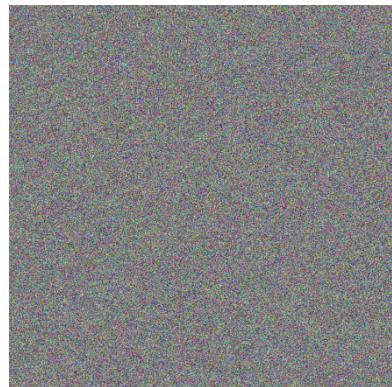
Instrumentul de analiză vizuală cel mai des utilizat pentru a studia distribuția valorilor pixelilor unei imagini color îl constituie *histograma culorilor*, în care

sunt reprezentate grafic frecvențele valorilor pixelilor unei imagini, separat pentru fiecare canal de culoare.

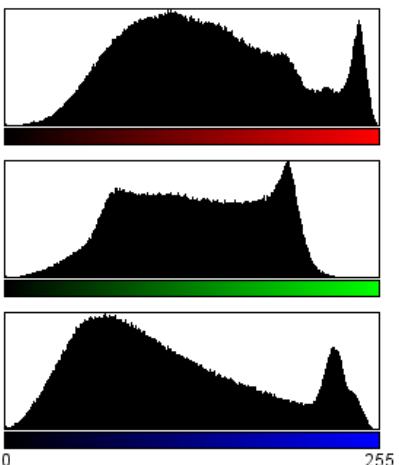
În figura de mai jos puteți observa o imagine în clar și imaginea criptată corespunzătoare, împreună cu histogramele lor:



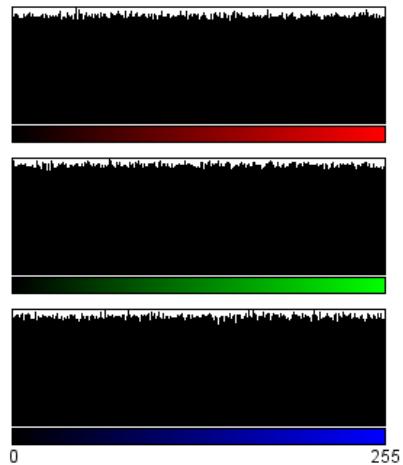
a) Imaginea în clar



b) Imaginea criptată



c) Histograma imaginii în clar



d) Histograma imaginii criptate

Figura 4. Imaginea în clar (a) și imaginea criptată (b) corespunzătoare, împreună cu histogramele corespunzătoare (c) și (d).

Se observă faptul că în urma procesului de criptare a imaginii în clar, având o distribuție a culorilor puternic neuniformă, s-a obținut o imagine criptată cu o distribuție uniformă a valorilor pixelilor pentru fiecare canal de culoare, deci un posibil atacator nu va putea extrage informații de natură statistică referitoare la imaginea inițială sau la cheia secretă utilizată.

Analiza vizuală a uniformității distribuției 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 și de calitatea grafică a histogramei. Dezavantajele analizei vizuale pot fi eliminate prin utilizarea unui instrument statistic de evaluare a uniformității distribuției valorilor dintr-un sir, respectiv testul χ^2 (*testul chi-pătrat*).

Valoarea testului χ^2 corespunzătoare unei imagini de dimensiune $m \times n$ se calculează folosind formula următoare:

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

unde f_i este frecvența valorii i ($0 \leq i \leq 255$) pe un canal de culoare al imaginii, iar \bar{f} este frecvența estimată teoretic a oricărei valori i , respectiv $\bar{f} = \frac{m \times n}{256}$. O valoare a testului mai mică sau egală decât 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).

Din valorile numerice anterioare se poate observa faptul că imaginea în clar are distribuții ale valorilor pixelilor puternic neuniforme pe toate cele canale de culoare (fapt ușor de observat și din analiza vizuală a histogramei sale), în timp ce imaginea criptată are o distribuție uniformă a valorilor pixelilor pe canalele R și G, dar prezintă o ușoară neuniformitate a distribuției pe canalul B (fapt aproape imposibil de detectat prin analiza vizuală!).

Puteți să vă verificați corectitudinea criptării imaginii *peppers.bmp* rulând programul scris în sursa *verificator_criptare.c* ce compară imaginea voastră criptată cu imaginea criptată *enc_peppers_ok.bmp* obținută de noi prin criptare folosind cheia secretă dată de fișierul *secret_key.txt*. Fișierul *chi-squared-test-values.txt* conține statistici legate de valorile testului χ^2 pe canalele de culoare RGB pentru imaginea inițială *peppers.bmp* și cea criptată *enc_peppers_ok.bmp*.

Cerințe:

- 1) Implementați generatorul de numere pseudo-aleatoare XORSHIFT32 sub forma unei funcții sau a unei macrodefiniții. (0.5 puncte)
- 2) Scrieți o funcție care să încarce o imagine BMP în memoria internă în formă liniarizată. Funcția va avea ca parametru calea imaginii BMP. (0.5 puncte)
- 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. (0.5 puncte)

- 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. (1 punct)
- 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. (1 punct)
- 6) Scrieți o funcție care să afișeze valorile testului χ^2 pentru o imagine BMP pe fiecare canal de culoare. Funcția va avea ca parametru calea imaginii. (0.75 puncte)

Modulul de recunoașterea de pattern-uri într-o imagine

Procesarea digitală a imaginilor este o ramură a informaticii care utilizează algoritmi pentru prelucrarea imaginilor în format digital (și nu analogic) cu scopul de a extrage informații din imagini utile pentru rezolvarea unor probleme complexe. O astfel de problemă complexă o reprezintă recunoașterea de pattern-uri, particularizată în acest proiect la problema de recunoaștere a cifrelor scrise de mâna într-o imagine (Figura 5 stânga).



0	1	2	3	4
5	6	7	8	9

Figura 5. Stânga: imaginea cu cifre scrise de mâna. Dreapta: şabloane pentru cifrele 0-9.

Problema recunoașterii cifrelor scrise de mâna este o problemă fundamentală în procesarea digitală a imaginilor. Problema este grea încât cifrele diferă ca mărime, înfățișare, culoare, înclinare etc. Algoritmii cei mai complecși folosesc noțiuni de învățare automată (machine learning) și învață un model pentru fiecare cifră în parte din zeci de mii de imagini. Performanța lor pentru această problemă este comparabilă cu cea a oamenilor.

Pentru acest proiect vom folosi o metodă simplă în recunoașterea cifrelor scrise de mâna într-o imagine și anume găsirea de şabloane (template matching

https://en.wikipedia.org/wiki/Template_matching). Această metodă constă în găsirea de părți mici ale unei imagini care se potrivesc cu o imagine şablon. Pentru problema recunoașterii cifrelor scrise de mâna vom folosi pe post de şabloane imagini cu 10 cifre (Figura 5 dreapta). Ele se regăsesc în arhivă ca fișierele *0.bmp*, *1.bmp*, ..., *9.bmp*. Fiecare şablon este o imagine color (are 3 canale) cu dimensiunile 11×15 pixeli (11 pixeli în lățime, 15 pixeli în înălțime). Algoritmul de template matching aplicat pentru o imagine color I și un şablon S (o imagine color ce reprezintă o cifră) funcționează astfel:

- se transformă imaginea color I și şablonul S în imagini grayscale. Un punct de pornire este sursa *grayscale.c* din arhivă;
- se glisează şablonul curent pe imaginea I centrând şablonul 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 şablonul 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 şablonul S și fereastra f_I este următoarea:

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

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

La marginile imaginii I , şablonul S iese din imaginea I . Puteți implementa acest caz limită prin adăugarea unor pixeli cu valoarea 0 (pixeli de culoare neagră) în

afara imaginii I. O variantă mai puțin elegantă este să considerați numai pozițiile (x, y) din imaginea I pentru care şablonul S încape în imaginea I.

Corelația dintre şablonul S și fereastra corespunzătoare f_I este mare la pozițiile (x, y) din imaginea I unde conținutul imaginii se potrivește cu conținutul şablonului S . Corelația este un număr real între -1 și 1, o valoare a corelației de 1 indică o corelație perfectă cu şablonul S . Pentru vizualizarea ferestrelor f_I cu corelație mare cu şablonul S (numite în cele ce urmează *detectii*) se realizează următoarele:

- c) se păstrează ca detectii ferestrele cu corelație mai mare decât un prag p_S . Valori mici ale pragului p_S conduc la un număr mare de detectii f_I , valori mari ale pragului p_S conduc la un număr mic de detectii f_I .
- d) pentru vizualizarea detectiilor f_I putem colora cu o numită culoare conturul ferestrei f_I în imagine.

Figura 6 prezintă rezultatele obținute pentru şabloanele cu cifrele 0 și 5 și pragurile 0.35 și 0.70.

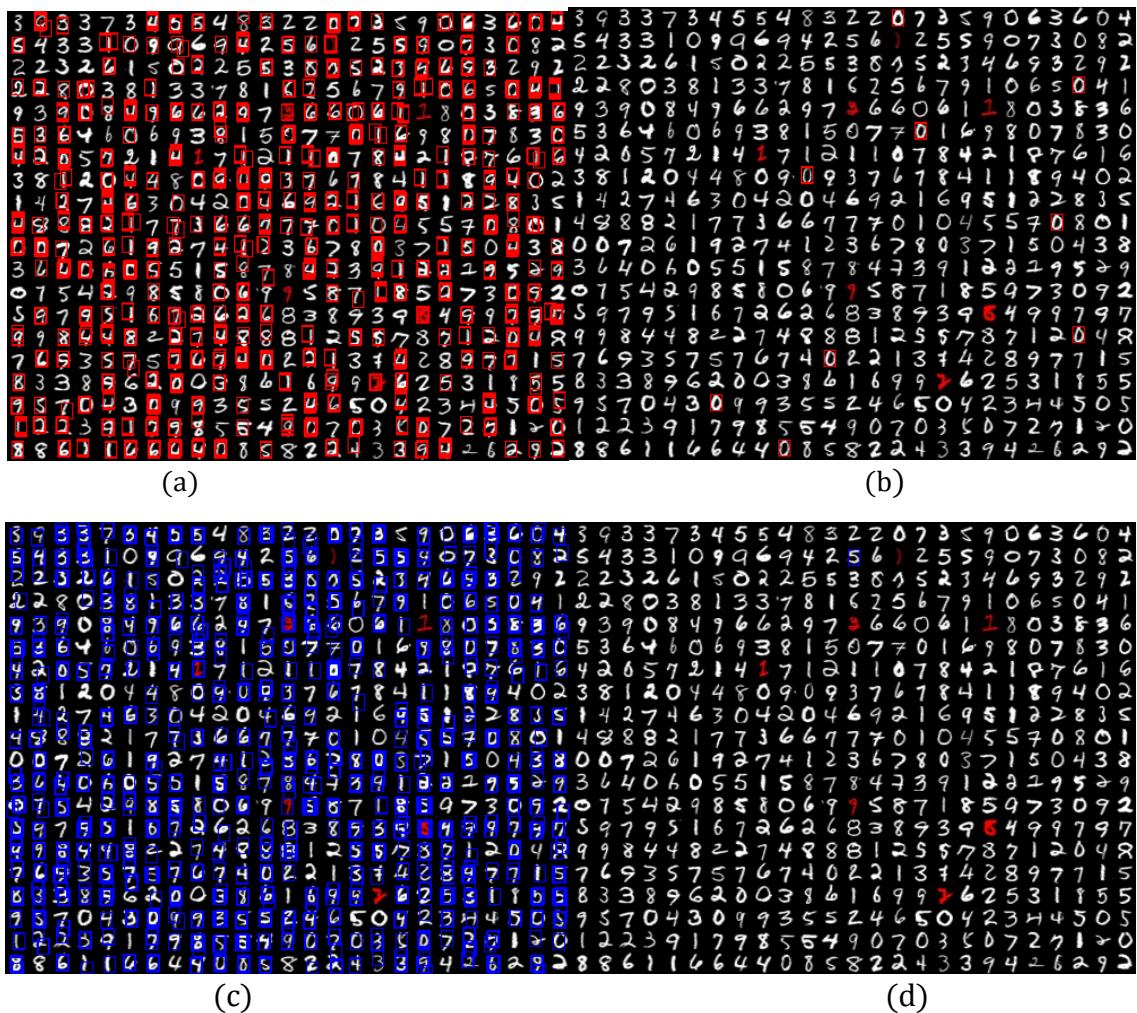


Figura 6. Rezultatele obținute pentru template matching (a) folosind şablonul pentru cifra 0 și pragul 0.35 (b) folosind şablonul pentru cifra 0 și pragul 0.70; (c) folosind şablonul pentru cifra 5 și pragul 0.35 (b) folosind şablonul pentru cifra 5 și pragul 0.70;

Rularea tuturor celor 10 şabloane pe imaginea I pentru diverse valori va conduce la rezultate asemănătoare, în funcție de pragul ales. În momentul în care reunim detectiile tuturor cifrelor vom vedea că ele se suprapun foarte mult, ca în Figura 7 stânga (pragul ales este 0.5):

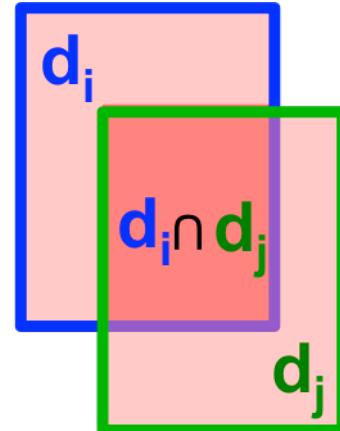
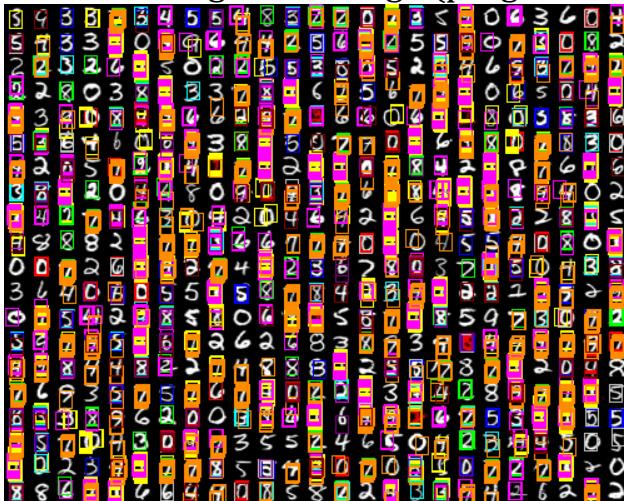


Figura 7. Stânga: Rezultatele obținute pentru template matching folosind toate şabloanele și un pragul de 0.50 pentru toate cifrele. Dreapta: suprapunerea a două ferestre d_i și d_j .

După cum se poate observa în Figura 7, multe detectii (provenite din același şablon sau şabloane diferite) se suprapun.

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

- se pun într-un tablou unidimensional D toate detectiile obținute pentru toate cele 10 şabloane;
- se sortează tabloul D descrescător în funcție de scorul de corelație al fiecarei detectii;
- se procesează tabloul D sortat de la stânga (detectii cu scor foarte mare) la dreapta (detectii cu scor foarte mic) astfel: toate detectiile d_j care se suprapun spațial cu detectia curentă d_i , cu $i < j$ și deci și $\text{scor}(d_i) > \text{scor}(d_j)$ se elimină.

Suprapunerea spațială dintre două detectii d_i și d_j (Figura 7, dreapta) se măsoară ca raportul dintre intersecția și reuniunea ariilor celor două ferestre.

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

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



Figura 8. Stânga: rezultatele obținute pentru template matching folosind toate şabloanele și un pragul de 0.50 pentru toate cifrele. Dreapta: rezultatele obținut aplicând eliminarea non-maximelor pentru detectiile din imaginea din stânga.

După aplicarea algoritmului de eliminare a non-maximelor eliminăm toate detectiile care se suprapun (Figura 8, stânga). Detectiile rămase (Figura 8, dreapta) oferă o performanță mai bună pentru algoritmul de recunoaștere a cifrelor scris de mâna folosind template matching.

Cerințe:

- 7) Scrieți o funcție care să implementeze operație de template matching dintre o imagine I și un şablon S . Funcția va avea ca parametri imaginea I , şablonul 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 detectii în ordinea descrescătoare a corelațiilor detectiilor. (**0.25 puncte**)
- 10) Scrieți o funcție care implementează algoritmul de eliminare a non-maximelor. (**1.5 puncte**)
- 11) Scrieți un program care să realizeze următoarele operații: (**0.5 puncte**)
 - criptează o imagine color BMP și salvează imaginea criptată în memoria externă (cările ambelor imagini și a fișierului text care conține cheia secretă se vor citi de la tastatură sau dintr-un fișier text);
 - decriptează o imagine color BMP criptată și salvează imaginea decriptată în memoria externă (cările ambelor imagini și a fișierului text care conține cheia secretă se vor citi de la tastatură sau dintr-un fișier text);
 - afișează pe ecran valorile testului χ^2 pentru imaginea inițială și imaginea criptată, pe fiecare canal de culoare.
 - rulează operația de template matching pentru o imagine color BMP și o colecție de şabloane color BMP. Folosiți întotdeauna un prag $p_S = 0.5$

indiferent de şablonul ales. Reuniți toate detectiile rezultate pentru fiecare şablon în parte într-un singur tablou unidimensional D. Căile imaginii și a şabloanelor BMP se vor citi de la tastatură sau dintr-un fișier text.

- rulați funcția de eliminare a non-maximelor pe tabloul D și desenați în imagine detectiile rămase folosind o culoare specifică pentru fiecare şablon.

Folosiți imaginea *test.bmp* și şabloanele *0.bmp*, *1.bmp*, ..., *9.bmp* furnizate. La desenarea ferestrelor corespunzătoare, folosiți următoarele culori (https://www.rapidtables.com/web/color/RGB_Color.html):

Pentru cifra 0 – culoarea roșu : (255, 0, 0)
Pentru cifra 1 – culoarea galben: (255, 255, 0)
Pentru cifra 2 – culoarea verde : (0, 255, 0)
Pentru cifra 3 – culoarea cyan: (0, 255, 255)
Pentru cifra 4 – culoarea magenta : (255, 0, 255)
Pentru cifra 5 – culoarea albastru: (0, 0, 255)
Pentru cifra 6 – culoarea argintiu : (192,192, 192)
Pentru cifra 7 – culoarea albastru: (255, 140, 0)
Pentru cifra 8 – culoarea magenta : (128, 0, 128)
Pentru cifra 9 – culoarea albastru: (128, 0, 0)

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 predă proiectul lui Bogdan (adresa de email: bogdan.alex@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 funcția 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 restricțiile indicate (rezolvarea unor cerințe fără a folosi funcții, utilizarea unor variabile globale, neutilizarea funcției qsort pentru sortarea unui tablou etc.) vor primi punctaje parțiale.
6. Se acordă 1 punct din oficiu.