Programare procedurală 2018

* prezentare proiect -

Diaconu Rebeca-Mihaela

Grupa 133

**Criptarea unei imagini în format BMP**

Am folosit următoarele:

* am definit o structură de tip ***pixel*** ,cu ajutorul căreia am reținut în formă liniarizată imaginea, aceasta având 3 componente: intensitatea pixelului pentru culoarea roșu, cea pentru albastru și cea pentru verde.
* funcția ***citireDimensiuni***, având ca parametrii calea imaginii sursă și înca 2 variabile, în care vom reține dimensiunile acesteia
* funcția ***liniarizare*** , cu parametrii: de ieșire, ***un vector Liniar***, de tip pixel, ( structura definită mai sus) și, de intrare, calea imaginii sursă; în această funcție am deschis fișierul bitmap, am citit dimensiunile imaginii și am construit tabloul liniarizat pe parcurs ce citeam pixel cu pixel din fișier
* funcția ***xorShift32*** este folosită pentru generarea a 2\*lațimeIMG\*înălțimeIMG-1 numere random, prima jumătate fiind folosită la generarea permutărilor și cea de a doua la criptarea imaginii; această funcție are ca parametrii calea fișierului text din care citim cheia secretă, dimensiunile imaginii, si un vector ***R,*** transmis ca parametru de ieșire, unde am reținut numerele generate
* funcția ***permutariCriptare*** generează o permutare de dimensiuni lățimeIMG\*înălțimeIMG, folosind algortimul lui Durstenfeld și prima jumătate din ***vectorul R***, cu numerele random, iar apoi permută pixelii imaginii sursă, în formă liniarizată, și crează ***un nou vector, Perm, de tip pixel***, cu imaginea permutată; drept parametrii avem: vectorul R, dimensiunile imaginii, ***vectorul Liniar*** ( ce conține imaginea sursă liniarizată) și vectorul Perm, transmis ca parametru de ieșire
* funcția ***XORconstanta***, cu un singur parametru, de intrare: această funcție este folosită pentru ușurarea operației xor dintre un unsigned char și un unsigned int, cu ajutorul unei ***măști*** și al operației ***&*** pentru a genera, pe rând, octeții unui număr
* funcția ***afisare*** are 5 parametrii: vectorul ce conține imaginea în forma liniarizată, dimensiunile acesteia, lațime și înălțime, calea fișierului bitmap ce conține imaginea sursă și calea fițierului bitmap ce conține destinația unde va fi salvată noua imagine
* funția ***criptareImag*** are 3 parametrii: calea imaginii sursă, fișierul bitmap destinație pentru imaginea formată și fișierul text ce conține cheia secretă; în interiorul acestei funcții le apele pe toate cele enumerate mai sus și realizez criptarea imaginii conform algoritmului precizat în enunțul proiectului

**Rularea testului Chi-Squared pentru o imagine în format BMP**

Am folosit o singură funcție pentru rularea acestui test și anume ***chi***, ce are ca parametru calea imaginii în format bitmap pentru care voi face testul.

În interiorul acestei funcții am format 3 vectori de frecvență, inițializați cu 0, câte unul pentru fiecare culoare, roșu, albastru și verde, după care am parcurs imaginea în forma ei liniarizata și am crescut valoarea din vectorul specific culorii pentru fiecare intensitate din fiecare pixel.

Având vectorii completați, am calculat valorile testului chi-squared pentru fiecare culoare, conform enunțului, și le-am afișat pe ecran, cu un mesaj adecvat.

**Decriptarea unei imaginii în format BMP**

Am folosit următoarele:

* funcțiile ***XORconstanta***, ***liniarizare***, ***xorShift32***, ***afisare***, precizate mai sus
* funcția ***ImagInterm,*** în care realizez procedeul invers criptării și formez o imagine nouă; aceasta are 6 parametrii: vectorul ***R***, ce conține numerele random, dimensiunile imaginii, fișierul text din care citesc cheia secretă, imaginea criptată sub forma liniarizată, ***vectorul C*** și vectorul de tip pixel, ***Interm***, ca parametru de ieșire
* funcția ***permutariDecriptare***, cu 5 parametrii: imaginea rezultată din apelul funcției de mai sus, vectorul R, dimensiunile imaginii si vectorul ,de tip pixel, D, ca parametru de ieșire, unde vom reține imaginea decriptată în formă liniarizată; aici generez inaltime\_img \* latime\_img permutari, dupa care calculez inversele lor si formez o imagine noua, decriptata
* funcția ***decriptareImag,*** unde apelez funcțiile precizate mai sus și realizez decriptarea imaginii conform enunțului proiectului

**Modulul de recunoașterea de pattern-uri într-o imagine BMP**

Am folosit următoarele:

* funcția ***grayscale\_image***, cu 2 parametrii; cu ajutorul ei fac imaginea originală gri
* funcțiile ***sablonMediu*** și ***imgMediu***, unde calculez intensitatea medie pentru fiecare șablon și fereastră formată în parte, formule ce vor ajuta la calculul corelației
* funcțiile ***sablonDev*** și ***imgDev***, unde calculez deviația standard pentru fiecare șablon și fereastră formată în parte, formule ce vor ajuta la calculul corelației
* o structură ***templat***, care mă va ajuta să rețin pozițiile de la care formez fiecare fereastră, dimensiunile acesteia, cifra asociată șablonului și corelația dintre fereastră și șablon
* funcția ***vector***, unde calculez pentru fiecare fereastră și șablon în parte corelația, verific dacă este peste pragul admis; în caz afirmativ, rețin pozițiile de la care am format imaginea, corelația acesteia cu șablonul, cifra asociată șablonului și dimensiunile ei într-un vector de tip structură, definită la inceputul programului
* funcția ***contur***, prin intermediul căreia trasez conturul fiecărei detecții, în funcție de cifra asociată șablonului
* funcția ***compareCorr***, care este funcția de comparare folosită la qsort
* funcția ***suprapunere***, care calculează suprapunerea dintre detecțiile gasite în imagine
* funcția ***intersec***, care verifică dacă 2 arii se suprapun
* funcția ***arie***, care, în cazul în care 2 arii se suprapun, calculează aria de intersecție
* funcția ***stergere***, care șterge un element aflat pe poziția k, transmisă ca parametru, din vectorul de detecții
* funcția ***maxime***, unde apelez funcțiile prezentate mai sus și elimin detecțiile, după regula impusă de proiect
* funcția ***templateMatching***, unde apelez funcțiile de mai sus, fac gri imaginea sursă și sabloanele, calculez suprapunerea dintre fiecare fereastră și fiecare șablon, alcătuiesc vectorul de detecții, de tip template, elimin maximele și trasez conturul, obținând astfel imaginea dorită

În funcția ***main*** am format un meniu switch pentru a lăsa la alegerea utilizatorului ordinarea în care va face operațiile de criptare, decriptare, rularea testului chi-squared și recunoașterea pattern-urilor date pe imaginea sursă.  
 De menționat este că numele șabloanelor se vor citi dintr-un fișier text, ***sablon.txt*** regăsit în arhiva proiectului.