Documentatie Proiect PP

ANDREI BUDOI – GRUPA 133

Pentru a putea rula, programul citeste calea fisierelor necesare din “fisiere.txt”. Acesta are urmatoarea strucura:

imagine_originala.bmp

imagine_criptata.bmp

imagine_decriptata.bmp

cheie_secreta.txt

sablon0.bmp

sablon1.bmp

sablon2.bmp

sablon3.bmp

sablon4.bmp

sablon5.bmp

sablon6.bmp

sablon7.bmp

sablon8.bmp

sablon9.bmp

imagine_finala.bmp

- Modulul de Criptare / Decriptare

Procesul de criptare incepe prin apelarea functiei *void criptare(char numepoza[ ], char numepoza_poz[ ], char secretKey[ ])*.

Functia *void f_seed(uint32_t *xorseed, uint32_t *SV, char secretKey[ ])* citeste R0 si SV dintr-un fisier.

Functia *uint32_t *XorShift(uint32_t seed,uint32_t W, uint32_t H)* genereaza un vector de dimensiune 2*W*H cu elemente cu caracte pseudo-aleator pornind de la un numar initial. Se returneaza un vector cu numere aleatoare.

*void FileErrorCheck(FILE *f, char *numepoza)* verifica daca un fisier a fost deschis in mod corect.

Functia *pixelBGR *citire_encript(char numepoza[], HeaderInfo *header)* citeste intregul header al unei imagini, ia fiecare pixel din fisierul tip .bmp si il retine intr-un array. Pixelii sunt cititi luand cate 3 octeti, unul pentru intensitatea fiecarui canal (albastru, verde, rosu). In reprezentarea binara a fisierului, matricea pixelilor este inversata (pixelii se citesc normal, de la stanga la dreapta, dar primul rand este ultimul din imaginea propriu zisa). Se returneaza vectorul de pixeli.

Headerul este retinut intr-o variabila de tip struct HeaderInfo, din care sa accesez direct informatii despre imagine precum dimensiune, inaltime si latime, etc.

Pentru memorarea pixelilor am creat o structura ce retine 3 octeti fara semn, fiecare pentru un canal de culoare. Culorile sunt citite folosind tripletul blue, green, red.

Functia *pixelBGR *permutare (HeaderInfo header, pixelBGR *forma_liniara, uint32_t xorseed)* permuta elementele vectorului *forma_liniara* folosind *algoritmul*

de permutare aleatoare a lui Durstenfeld si generatorul de numere cu caracte pseudo-aleator *xorshift32.* Se returneaza vectorul cu elementele permutate.

Functia *pixelBGR *modificarePixeli (HeaderInfo header, pixelBGR *forma_liniara, uint32_t SV, uint32_t xorseed)* modifica valorile pixelilor conform formulei indicate. Operatiile de xor se fac octet cu octet. Se returneaza vectorul cu elementele modificate.

Functia *void afisare_encript (char numepoza_mod[ ], HeaderInfo header, pixelBGR *forma_liniara)* creaza un fisier cu extensia *.bmp* folosind un header si un array de pixeli.

Functia *void chi (HeaderInfo header, pixelBGR *forma_liniara, char *numepoza)* calculeaza testul chi-patrat pentru o imagine folosind vectorul de pixeli. Se afiseaza pe ecran valorile pentru fiecare canal de culoare.

Procesul de decriptare incepe prin apelarea functiei *void decriptare(char numepoza[ ], char numepoza_mod[ ], char secretKey[ ])*

Functia *pixelBGR *permutare_decript(HeaderInfo header, pixelBGR *forma_liniara, uint32_t xorseed)* permuta invers elementele vectorului *forma_liniara* folosind *algoritmul de permutare aleatoare a lui Durstenfeld* si generatorul de numere cu caracte pseudo-aleator *xorshift32.* Se returneaza vectorul cu elementele invers permutate.

Functia *pixelBGR *modificarePixeli_decript (HeaderInfo header, pixelBGR *forma_liniara, uint32_t SV, uint32_t xorseed)* modifica valorile pixelilor conform formulei indicate pentru decriptare. Operatiile de xor se fac octet cu octet. Se returneaza vectorul cu elementele modificate.

- Modulul de recunoastere a pattern-urilor

Procesul de recunoastere incepe prin apelarea functiei

*void recunoastere (char *source, char *source_mod, char *sablon0, char *sablon1, char *sablon2, char *sablon3, char *sablon4, char *sablon5, char *sablon6, char *sablon7, char *sablon8 ,char *sablon9)* ce primeste calea fiecarui fisier ce trebuie folosit.

Functia *pixelBGR *citire_mat(char numepoza[], HeaderInfo *header)* salveaza in memorie headerul si o matrice de pixeli ale imaginii cu calea *numepoza.*

Functia *void greyscale(pixelBGR **matrice,HeaderInfo header)* modifica elementele unei matrice de pixeli, transformand imaginea in grayscale.

Functia *void matching (pixelBGR **source, pixelBGR **s ,pixelBGR **source_mod, HeaderInfo header_source, HeaderInfo header_sablon, detectie *D, uint32_t *k, double pS, uint8_t R, uint8_t G, uint8_t B)* primeste o imagine si un sablon pentru care se va realiza procesul de template matching. Toate detectiile gasite cu corelatia mai mare de *pS* se vor salva in vectorul *D* folosind, pentru desenarea chenarului, culoarea aferenta sablonului.

Un element de tip *detectie* din vectorul *D* retine corelatia, centrul ferestrei si culoarea cu care trebuie desenat chenarul aferent.

Functia *double corr(pixelBGR **source, pixelBGR **s , HeaderInfo header_source, HeaderInfo header_sablon, uint32_t x, uint32_t y)* calculeaza corelatia dintre un sablon *s* si o fereastra cu centru *(x, y)* din imaginea *source* folosind formula din cerinta. Se returneaza o valoare intre -1 si 1 ce reprezinta valoarea corelatiei.

Functia *int cmp(const void* a, const void* b)* este folosita in sortarea tabloului de detectii.

Functia *void eliminare(detectie *D, uint32_t *k, HeaderInfo header)* sterge din tablou detectiile care se suprapun si au o corelatie mai mica. Suprapunerea spatiala a doua ferestre se calculeaza folosind functia *double intersect (uint32_t n, uint32_t L1,uint32_t R1, uint32_t B1 ,uint32_t U1,uint32_t L2,uint32_t R2, uint32_t B2,uint32_t U2)* ce primeste aria unei ferestre si pozitiile laturilor celor doua.

Functia *void incadrare (pixelBGR **source_mod ,HeaderInfo header_source, uint32_t H, uint32_t W,uint32_t x, uint32_t y, uint8_t R, uint8_t G, uint8_t B)* deseneaza un cadran cu culoarea *(R,G,B)* cu centru *(x,y)* si dimensiunea *H*W.*

Functia *void afisare_mat(char numepoza_mod[ ], HeaderInfo header, pixelBGR **matrice_pixeli)* creaza un fisier cu extensia *.bmp* folosind un header si o matrice de pixeli.