PREZENTAREA PROIECTULUI

In acest PDF voi explica functiile folosite pentru rezolvarea proiectului .

1.unsigned int * xorshift32(unsigned int seed,unsigned int n)

Aceasta functie genereaza n numere intregi pseudo-aleatoare fara semn pe 32 de biti pornind de la o valoare initiala(seed) ,folosind XOR si operatii de deplasare pe biti,astfel incat oricare doua numere din aceasta secventa vor fi diferite.

Vectorul ce contine aceste elemente va fi returnat in urma apelului functiei.

2.void citire_imagine(char*nume_sursa,pixel **L,unsigned char **header)

Functia citire_imagine citeste o imagine bmp stiind calea sa(nume_sursa) si creeaza un vector ce va contine imaginea liniarizata intoarsa.

Liniarizarea va fi de forma pixel,structura ce contine cele 3 canale : R,G,B si va fi retinuta in parametrul L.

De asemenea,voi pastra header-ul intr un vector pentru a-l folosi ulterior in functia de afisare.

3.void afisare(char*nume_sursa,char*nume_destinatie,pixel *L,unsigned char *header)

Dupa efectuarea unor operatii asupra imaginii initiale (nume_sursa) vom avea nevoie de un alt fisier bmp in care vom pastra imaginea modificata (nume_destinatie).

Afisarea se va realiza prin copierea header-ului in noua imagine , a liniarizarii,pe care o vom intoarce din nou si adaugarea padding-ului.

4.pixel* criptare(char*nume_sursa,char*nume_destinatie,char*cheie_secreta)

Aceasta functie apeleaza functia XORSHIFT32 si retine vectorul ,realizeaza permutarea imaginii si criptarea acesteia in functie de valorile din fisierul cheie_secreta.

Permutarea este efectuata conform algoritmului Durstenfeld folosind prima jumatate din vectorul realizat in urma operatiei XORSHIFT32.

Criptarea consta in operatia xor pe urmatoarele cazuri : primul element se va obtine prin operatia xor intre al doilea numar din fisierul cheie_secreta,numarul aflat in permutare la pozitia 0 si primul element din a doua jumatate a vectorului din XORSHIFT32,iar restul elementelor(pozitia i!=0) prin operatia xor intre elementul anterior, numarul aflat in permutare la pozitia i si al i-le termen din a doua jumatate a vectorului din XORSHIFT32.

Functia va intoarce un vector de tip pixel,reprezentand liniarizarea imaginii criptate.

5.pixel* decriptare(char*nume_sursa,char*nume_destinatie,char*cheie_secreta)

Asemenea functiei de criptare,functia de decriptare apeleaza si retine vectorul rezultat in urma functiei XORSHIFT32,creeaza aceeasi permutare si o inverseaza,iar apoi aplica inversa relatiei de substitutie folosita in procesul de criptare. Ca ultim pas,imaginea decriptata se obtine prin permutarea pixelilor(obtinuti in urma aplicarii inversei relatiei de substitutie din criptare) conform permturaii inverse.

Fisierul nume_sursa va contine calea imaginii criptate in functia anterioara,iar nume_destinatie va afisa imaginea decriptata.

6.void chi(char*nume_sursa,char*cheie_secreta)

Functia chi calculeaza,pe o imaginea transmisa prin calea nume_sursa, distributia pixelilor pe fiecare canal de culoare.

Mai intai se calculeaza frecventa fiecarui pixel ,de la 0 la 255,pe fiecare canal de culoare si apoi se aplica formula corespunzatoare.

7.void grayscale_image(char* nume_fisier_sursa,char* nume_fisier_destinatie)

Aceasta functie transforma o imagine transmisa prin nume_fisier_sursa in aceeasi imagine,dar in nuante de gri.

8.double medie_val(unsigned int i,unsigned int j,unsigned int h,unsigned int w,unsigned int hsab,unsigned wsab,pixel*vec)

9.double deviatie_st(unsigned int i,unsigned int j,unsigned int h,unsigned int w,unsigned int hsab,unsigned int wsab,pixel*vec)

Functiile de la punctele 8 si 9 reprezinta calcule intermediare in interiorul calculului corelatiei dintre o pozitie din imaginea pe care se ruleaza operatia de template matching si sabloane.

Functia 8 calculeaza media valorilor intensitatilor grayscale a pixelilor,atat in fereastra curenta,cat si a sablonului curent,iar functia 9 calculeaza deviatia standard a valorilor intensitatilor grayscale a pixelilor, in fereastra si sablonul curent.

Prin indicii i si j va fi transmisa fereastra curenta,iar vec va reprezenta fie liniarizarea imaginii,fie a sablonului.

10.void contur(pixelI,unsigned int cnt,corelatie*D,unsigned char pR,unsigned char pG,unsigned char pB,unsigned int latime_img,unsigned int inaltime_img,unsigned int latime_sablon,unsigned int inaltime_sablon)**

Dupa operatia template matching si eliminarea non-maximelor voi avea un vector de detectii,vector de tip corelatie,iar in functie de cifra detectata,asupra imaginii(ce are liniarizarea I),se va realiza un contur al fiecarei detectii(pR,pB,pG).

11.void ferestre(char*nume_imagine,char*nume_sablon,corelatieD,int*cnt_ferestre,float ps,unsigned int cifra_sablon,unsigned int *wsab,unsigned int *hsab)**

In aceasta functie,pentru un sablon anume(nume_sablon),se va realiza glisare acestuia in imagine (nume_imagine) si se va calcula corelatia dintre sablon si fiecare fereastra.

Orice corelatie ce depaseste un prag ales,se va adauga in vectorul de detectii D.

12.int compar_corelatie(const void*a,const void*b)

13.void sortare(corelatieD,int cnt_ferestre)**

Aceste doua functii sorteaza vectorul de detectii prin functia qsort,descrescator,dupa valorile corelatiilor.

14.double arie_intersectie(int i,int j,corelatie*D,unsigned int w,unsigned int h)

Functia arie_intersectie calculeaza aria intersectiei a doua ferestre din vectorul de detectii,aflate la pozitiile i si j.

15.void suprapunere(corelatieD,int *cnt_ferestre,unsigned int w,unsigned int h)**

In functia suprapunere se realizeaza eliminarea non maximelor prin formula arie_intersectie/arie_reuniune(intre oricare 2 ferestre).

Daca aceasta este mai mare decat 0.2,inseamna ca cele doua ferestre se suprapun si se va elimina cea cu corelatia mai mica.

Pentru a elimina,voi crea un vector de tip corelatie care va fi initial egal cu cel de detectii. De fiecare data cand vreau sa elimin o detectie ii voi schimba valoarea corelatiei in 0.

In final,voi reface vectorul de detectii,transferand din vectorul precedent doar detectiile care au corelatia diferita de 0.

16.pixel* template_matching(char*nume_imagine,char*nume_grayscale)

Functi finala,template_matching,apeleaza,pentru toate cele 10 sabloane,functiile grayscale si ferestre.

Astfel,voi avea un vector de detectii de tipul corelatie pe care il voi sorta si asupra caruia voi aplica functia suprapunere.

In final,voi aplica functia contur detectiilor ramase.