Programare Procedurala
Documentatie proiect

Avram Adrian-Constantin

Criptarea unei imagini în format BMP

În interiorul funcției alocate și citite tablouri ce vor reprezenta căile către fișierele BMP folosite ce urmează sa fie create.Toate antetele utilizate sunt stocate in header-ul cod.h.După ce au fost alocate și citite fiecare cale de acces, se apelează funcția criptare. In interiorul acesteia se va încărca imaginea sub forma liniarizată prin intermediul funcției citire. Se deschide fișierul ce conține cheia secreta și se citesc cele doua numere reprezentând seed-ul pentru generator R0 și numărul cu care se inițializează pixelii, SV. Se aloca memorie pentru tabloul de numere aleatoare și pentru permutare. Numerele aleatoare vor fi generate prin intermediul funcției xorshift. Funcția se află în fișierul-antet random.h și furnizează valori aleatoare folosind algoritmul XORSHIFT32. Se inițializează vectorul v cu valorile corespunzătoare permutării identice, urmând să se aplice transpoziții utilizând numerele în cadrul algoritmului Durstenfeld. Se aloca un tablou în care se va reține configurația rezultata în urma algoritmului asupra tabloului . Se aplica substituții la nivel de element în cadrul tabloului auxArray, folosind operații XOR la nivel de pixeli și intre pixel și un număr; primul element se va XOR-a cu valoarea SV și numărul aleator de pe poziția pixelNo - 1, următoarele elemente fiind XOR-ate cu valorile anterioare și numărul aleator corespunzător. În final se salvează noua configurație la calea precizată prin al doilea parametru al funcției.

Decriptarea unei imagini în format BMP

În interiorul funcției sunt alocate și citite tablouri ce vor reprezenta căile către fișierele BMP folosite/ ce urmează sa fie create. In interiorul acesteia se va încărca imaginea sub forma liniarizată în tablou prin intermediul funcției citire. Se deschide fișierul ce conține cheia secreta și se citesc cele doua numere reprezentând seed-ul pentru generator R0 și numărul cu care se inițializează substituția pixelilor, SV. Se aloca memorie pentru tabloul de numere aleatoare) și pentru permutare. Numerele aleatoare vor fi generate prin intermediul funcției xorshift32 care furnizează valori aleatoare folosind algoritmul XORSHIFT32. Se aplica substituții la nivel de element în cadrul tabloului în ordine inversă față de algoritmul de criptare (de la ultimul element la primul element) Se inițializează v cu valorile corespunzătoare permutării identice, urmând să se aplice transpoziții utilizând numerele din vector în cadrul algoritmului Durstenfeld. Se calculează inversa permutării a, ce va fi reținută în tabloul a2t, alocat dinamic. Se aloca un tablou decrypt în care se va reține configurația rezultata în urma aplicării permutării asupra tabloului a2. În final se salvează noua configurație la calea precizată prin al doilea parametru al funcției imag_fin .

Testul Chi-Patrat

În main se apelează funcția chi_patrat.În interiorul funcției chi_patrat se încarcă în tabloul imag_ini imaginea aflată la calea transmisă ca parametru, sub formă liniarizată. Se alocă dinamic tablourile r, b si g ce vor stoca frecvența valorilor pixelilor din imag_ini, pe canalul corespunzător. Se parcurge tabloul imag_ini și se actualizează frecventa valorilor de pe fiecare canal. Se calculează media presupusă pe fiecare canal . Se parcurge fiecare tablou de frecvență și se calculează media reală a valorilor si se afișează conform formulei pe ecran cele 3 valori corespunzătoare mediei canalului roșu, canalului verde și canalului albastru ,dupa care se elibereaza memoria .

Template-matching intr-o imagine în format BMP

În interiorul funcției sunt alocate și citite tablouri ce vor reprezenta căile către fișierele BMP folosite/ ce urmează sa fie create. În interiorul antetului template_matching se regaseste ps, reprezentând pragul folosit în cadrul funcției de corelație.Se citește numărul de șabloane care ulterior se pune intr-un vector pe care se va rula algoritmul de matching b[10],udne fiecare sablon e citit din functia citire prin functia sprintf .In fisierul antet se regaseste calea catre imagine , care este ulterior citit din fisier prin intermediul functiei citire. Conversia vectorului se face preluand din functia de conversie oferita in proiect doar partea de transformare in gri a pixelilor.Se parcurge tabloul imag_ini pixel si pixel si Se calculeaza punctul de inceput al unui sablon care e centrat pe un punct "i". Se verifica daca fereastra de marimea sablonului ar incapea in imagine , daca da , atunci va urma parcurgerea algoritmului , astfel va trece la urmatorul pixel.Calculez prima oara intensitatea medie a sablonului si a ferestrei , ulterior calculand deviatia pentru ambele dintre ele.Realizate odata cele doua ,calculez corelatia conform formulei date si verific daca este mai mare decat up prag da .Daca este , atunci acea imagine o introduc intr-un vector aflat si in antet (cor).Urmand apoi sa apelez functia auxiliara colorare aflată în cod.h ce are drept scop colorare portiunii unde s-a regasit fereastra cu pragul mai mare decat ps.Se elibereaza memoria alocata ptr salvarea sablonului aflat la iteratia respectiva ,programul reluandu-se pentru urmatorul sablon.La final , se sorteaza cu ajutorul functiei qsort din libraria stdio.h toate detectiile dupa corelatie , conform cerintei date.

Functiile auxiliare utilizate

Functiile auxiliare au fost folosite in cod.c . Declararea si antetelor au fost facute in cod.h pentru a usura accesul acestora.
unsigned int *xorshift32(unsigned int seed,int n)-functie ce furnizeaza numerele generate aleatoriu
void swap(unsigned int *a,unsigned int *b)-realizeaza interschimbarea a 2 variabile sau structuri
imagine *citire(char *imag_ini)-citeste imaginea conform unui lucru cu un fisier bitmap.Liniile sunt interschimbare , la fel precum si culorile rosu cu albastru.
imagine *citireNormal(char *imag_init)-citeste imaginea normal fara a actiona asupra acesteia
int compar(const void*a,const void*b)-functie folosita in utilizarea algoritmului de sortare quicksort.
De asemenea , in cod.h am introdus si structurile utilizate in rezolvarea proiectului :
-Structul pixel are in componenta trei octeti reprezentand cele 3 canale de culoare (blue,green,red) pe care un pixel dintr-un fisier bitmap le detine.

```
typedef struct {
unsigned char b,g,r;}pixel;
```

-Structul imagine este format din lungimea imaginii(width) , inaltimea sa (height) si un pointer catre un pixel ce este format din cele 3 canale de culoare

```
typedef struct {  pixel *p;
int w,h;
}imagine;
```

-Structul detectie este si acesta format din lungimea , inaltimea si un pointer catre pixeli , insa are inca o valoare "v" ce retine corelatia unei detectii.

```
typedef struct {pixel *p;
int w,h,v;}detectie;
```