Proiect PP – Documentație

Nume: [redacted]. Larisa

Grupa: 141 - Informatică

Data: 4 Ianuarie 2019

Cuprins:

Noțiuni Introductive:

Pe parcursul proiectului se folosesc următoarele notații:

- ui: unsigned int;
- uc: unsigned char.

Partea I: Criptarea

1) Funția ui *XORSHIFT32(ui w, ui h, ui seed)

Creează tabloul unidimensional cu numere generate aleator de algoritmul lui George Marsaglia din 2003. Parametrii:

- w: lungimea imaginii care se dorește a fi criptată;
- h: înălțimea imaginii care se dorește a fi criptată;
- seed: cheia de criptare.

2) Structura Pixel {uc B, G, R;};

Structura unui pixel cu 3 canale de culoare (RedGreenBlue - RGB). Datorită formatului little-endian, culorile sunt reținute invers în memorie (BGR). Parametrii:

- B / G / R: octeți care rețin valorile culorilor din cele 3 canale.

3) Pixel *liniarizare (char *sursa)

Creează tabloul unidimensional de Pixeli rezultat din liniarizarea imaginii sursă. Parametru:

- sursa: calea imaginii BMP sursă.

4) Funcția void Imagine(char *destinatie, Pixel L[], uc header[], ui h, ui w, ui padding)

Creează o imagine BMP pornind de la tabloul unidimensional de Pixeli liniarizat. Parametrii:

- char *destinatie: calea către imaginea BMP rezultată;
- Pixel L[]: tabloul unidimensional liniarizat de Pixeli (din altă poză rezultat);
- uc header[]: header-ul imaginii originale care trebuie copiat și în poza finală pentru a se crea;
- ui h: înălțimea pozei originale și a pozei finale;
- ui w: lățimea pozei originale și a celei finale;
- ui padding: numărul de octeți de padding care trebuie adăugați în poza finală.

5) Funcția ui *permutare(ui w, ui h, ui R[])

Creează un tablou unidimensional care reține permutarea ce va fi folosită pentru criptarea imaginii, folosindu-se algoritmul lui Durstenfeld. Parametrii:

- ui w: lățimea imaginii ce va fi criptată;
- ui h: înălțimea imaginii ce va fi criptată;

- ui R[]: tabloul unidimensional generat cu ajutorul funcției XORSHIFT32 ce conține numerele random.

6) Uniunea Intreg32

O uniune care permite accesarea fiecărui octet dintr-un int pe 32 de biți (4 octeți). Parametrii:

- ui x: un număr întreg pozitiv;
- uc Octet0 / Octet1 / Octet2 / Octet3: fiecare octet din structura unui int.

7) Pixel P_XOR_P(Pixel P1, Pixel P2)

Întoarce un nou Pixel rezultat din XOR-area a altor 2 Pixeli. Se XOR-ează fiecare canal de culoare. Parametrii:

- Pixel P1 / P2: pixeli ce trebuie XOR-ați (necesar pentru codificare pozei).

8) Pixel P_XOR_Nr(Pixel P1, union Intreg32 SV)

Întoarce un nou Pixel rezultat din XOR-area a unui Pixel și a unui număr întreg. Se XOR-ează fiecare canal de culoare cu octetul corespunzător din int. Parametrii:

- Pixel P1: pixel ce trebuie XOR-at;
- union Intreg32 SV: numărul întreg ce trebuie XOR-at, reținut într-o uniune, pentru a-i putea accesa octeții.

9) Funcția void criptare(char *sursa, char *destinatie, char *fisier_txt)

Funcția criptează o imagine BMP sursă și returnează imaginea criptată. Parametrii:

- char *sursa: calea imaginii BMP inițiale;
- char *destinatie: calea imaginii BMP criptate;
- char *fisier_txt: calea unui fișier text care conține cheia secretă.

10) Funcția void decriptare(char *sursa, char *imag_decriptata, char *fisier_txt)

Funcția decriptează o imagine sursă BMP(criptată) și returnează imaginea originală (de dinainte de criptare), folosindu-ne de cheile secrete oferite de fisierul_txt. Parametrii:

- char *sursa: calea imaginii BMP criptate;
- char *imag_decriptata: calea imaginii BMP decriptate;
- char *fisier_txt: calea unui fișier text care conține cheia secretă.

11) Funcția void histrograma(char *sursa)

Funcția afișează pe ecran valorile testului chi-pătrat pentru fiecare canal de culoare, în funcție de imaginea furnizată. Parametru:

- char *sursa: calea imaginii pe care se va aplica testul.

Partea II: Pattern Recognition

12) Funcția void grayscale(char *sursa, char *destinatie)

Funcția transformă o imagine color, în imagine alb-negru, folosindu-ne de funcții de la Partea I. Parametrii:

- char *sursa: calea imaginii BMP originale, pe care se aplică grayscale;
- char *destinatie: calea imaginii BMP rezultate din cea originală, în urma funcției.

13) Pixel **Matrice(char *sursa, ui wS, ui hS, ui *w, ui *h)

Funcția creează o matrice de Pixeli în care sunt reținuți Pixelii imaginii sursă, dar și Pixeli de contur, care au culoarea neagră. Matricea trebuie să aibă un contur cu dimensiunile Șablonului pentru a putea aplica funcția de template matching. Parametrii:

- char *sursa: imaginea BMP sursă pe care se va aplica funcția de template matching;
- ui wS: lățimea șablonului folosit pentru template matching;
- ui hS: înălțimea șablonului folosit pentru template matching;
- ui *w: lățimea matricei;
- ui *h: înălțimea matricei.

14) Pixel **M Sablon(char *sablon, ui *wS, ui *hS)

Funcția creează o matrice de Pixeli în care sunt reținuți Pixelii Șablonului folosit pentru template matching. Parametrii:

- char *sablon: Șablonul BMP folosit pentru template matching;
- ui *wS: vom reține lățimea șablonului;
- ui *hS: vom reține înălțimea șablonului.

15) Funcția double corelatie(Pixel **Imagine, Pixel **Sablon, ui wS, ui hS, ui wM, ui hM, ui l, ui c)

Funcția calculează formula de calcul a corelației dintre un șablon și o fereastră de dimensiunile șablonului din imaginea mare (pe care se aplică template matching-ul). Parametrii:

- Pixel **Imagine: matricea formată din Pixelii Imaginii Originale și conturul de dimensiunile șablonului (funcția 12);
- Pixel **Sablon: matricea formată din Pixelii șablonului;
- ui wS: lățimea **Sablon;
- ui hS: înălțimea **Sablon;
- ui wM: lățimea matricii **Imagine;
- ui hM: înălțimea matricii **Imagine;
- ui l: linia primului Pixel dintr-o fereastră a matricei **Imagine;
- ui c: coloana primului Pixel dintr-o fereastră a matricei **Imagine;

16) Structura fI

Structura fI reprezintă o fereastră (de dimensiunile șablonului folosit pentru template matching) din imaginea originală. Parametrii:

- ui X: linia Pixelului de început al ferestrei din imaginea originală;
- ui Y: coloana Pixelului de început al ferestrei din imaginea originală;
- double cor: corelația corespunzătoare ferestrei;
- Pixel C: culoarea chenarului ferestrei.

17) fI *v_detectie_cifra(char *Imagine_Gri, char *Sablon, double pS, ui *nr, Pixel Culoare, ui *w, ui *h)

Creează un tablou unidimensional de detecții pentru un șablon, imagine și un prag pS. În cadrul funcției, șablonul devine alb-negru și, dacă corelația ferestrelor găsite este mai mare sau egală cu pragul pS, sunt introduse în vector. Parametrii:

- char *Imagine_Gri: calea către imaginea sursă alb-negru, pe care se aplică template matching;
- char *Sablon: calea către șablonul folosit pentru operația de template matching;
- double pS: pragul folosit pentru detectarea ferestrelor;
- ui *nr: numărul de detecții;
- Pixel Culoare: culoarea conturului ferestrelor;
- ui *w: reținem lățimea ferestrei;
- ui *h: reținem înălțimea ferestrei.

18) Pixel Culoare(ui cifra)

Întoarce Pixelul culorii corespunzătoare valorii introduse în parametrul „cifra”.

19) Funcția void desenare(char *Imagine, fI Fereastra, ui wS, ui hS)

Funcția desenează conturul unei ferestre în imaginea sursă. Parametrii:

- char *Imagine: imaginea BMP sursă, unde se face desenarea conturului ferestrei;
- fI Fereastra: fereastra din imaginea sursă care trebuie conturată;
- ui wS: lățimea ferestrei;
- ui hS: înălțimea ferestrei.

20) Funcția int cmp(const void *a, const void *b)

Funcția compară 2 valori, în cazul de față, corelațiile a 2 ferestre și returnează 1 dacă prima valoare este mai mică decât a 2-a, -1 dacă este mai mare și 0 dacă sunt egale, aranjându-le descrescător.

21) fI *sort_desc(fI *v, ui n)

Funcția returnează vectorul de ferestre ordonat descrascător în funcție de corelații. Parametrii:

- fI *v: tabloul original de ferestre;
- ui n: numărul de elemente al vectorului de ferestre.

22) Funcția ui da_nu(fI P1, fI P2, ui wS, ui hS)

Funcția verifică dacă 2 ferestre se suprapun. Parametrii:

- fI P1 / P2: fereastra P1/P2 care conține Pixelul de început al ferestrei;
- ui wS: lățimea ferestrei;
- ui hS: înălțimea ferestrei.

Dimensiunile ferestrei ne ajută să determinăm coordonatele colțurilor ferestrelor pentru a putea verifica dacă acestea se suprapun sau nu. Funcția returnează 0 dacă ferestrel nu se suprapun sau 1, altfel.

23) Funcția double suprapunere(fI P1, fI P2, ui wS, ui hS)

Funcția calculează suprapunerea a doua ferestre. Parametrii:

- fI P1/P2: 2 ferestre;
- ui wS: lățime fereastră;
- ui hS: înălțime fereastră.

Dimensiunile ferestrei ne ajută să determinăm coordonatele colțurilor ferestrelor pentru a putea calcula suprapunerea.

24) fI *detectii(ui nr_sab, char *Imagine, double pS, ui *n, ui *w, ui *h)

Funcția returnează vectorul de detecții final, după ce a fost sortat descrescător, iar suprapunerile au fost eliminate. Parametrii:

- ui nr_sab: numărul de șabloane;
- char *Imagine: calea imaginii BMP originale pe care se execută pattern recognition;
- double pS: pragul de detecții;
- ui *n: reține numărul de detecții;
- ui *w: reține lățimea unei ferestre;
- ui *h: reține înălțimea unei ferestre.