Universitatea din Bucureşti
Facultatea de Matematică şi Informatică
Alexiu Adrian-Ştefan

Criptare/Decriptare
TemplateMatching

Universitatea din Bucureşti
Facultatea de Matematică şi Informatică
Alexiu Adrian-Ştefan

Criptare/Decriptare

void citire(char **numeImagine)

Citeste de la tastatura numele fisierului si il transmite prin intermediul parametrului “numeImagine”.

void scriere(unsigned int *p, int w, int h, char *numeImagineNoua, unsigned char *s)

Creeaza o noua imagine in format “.bmp”, avand ca parametrii un tablou unidimensional (“p”), reprezentand o imagine liniarizata, latimea (“w”), inaltimea (“h”), numele imaginii noi si header-ul (“s”).

void formaLiniarizata1(unsigned int **p, int *w, int *h, char *numeImagine)

Incarca in memorie imaginea “numeImagine” folosind un tablou unidimensional “p”. De asemenea, functia furnizeaza si latimea (“w”) si inaltimea (“h”) imaginii.

void XORSHIFT32(unsigned int **p, int n, int seed)

Geneareaza intr-un tablou unidimensional “p”, “n” numere pseudo-aleatoare pornind de la un “seed”.

void permutare(unsigned int **p, int n, unsigned int *r)

Genereaza o permutare de ordin “n” in tabloul unidimensional “p”, pe baza numerelor pseudo-aleatoare generate anterior (“r”).

void permutareInversa(unsigned int *p, unsigned int **q, int n)

Calculeaza in tabloul unidimensional “q”, permutarea inversa a permutarii “p”, “n” fiind ordinul permutarii.

void seed(char *numeFisierCheie, int *r0, int *sv)

Citeste din fisierul care contine cheia secreta, valorile “r0” si “sv”, ce vor fi folosite atat la criptare, cat si la decriptare.

void criptare1(char *numeImagine, char *numeImagineSemiCriptata, char *numeFisierCheie)

1. deschide imaginea initiala (“numeImagine”)
2. extrage header-ul
3. liniarizeaza imaginea
4. apeleaza functia “seed” pentru a extrage numerele “r0” si “sv”
5. apeleaza “XORSHIFT” pentru a genera numerele pseudo-aleatoare
6. genereaza permutarea pe baza numerelor generate anterior
7. tabloul q se obtine permutand elementele din tabloul initial (cel ce contine imaginea liniarizata) conform permutarii obtinute
8. creeaza imaginea auxiliara (“numeImagineSemiCriptata”)
9. elibereaza memoria alocata
10. inchide fisierul

void criptare2(char *numeImagineSemiCriptata, char *numeImagineCriptata, char *numeFisierCheie)

1. deschide imaginea auxiliara (“numeImagineSemiCriptata”)
2. extrage header-ul
3. liniarizeaza imaginea
4. apeleaza functia “seed” pentru a extrage numerele “r0” si “sv”
5. apeleaza “XORSHIFT” pentru a genera numerele pseudo-aleatoare
6. tabloul q se obtine folosind operatia “xor” conform algoritmului prezentat
7. creeaza imaginea criptata (“numeImagineCriptata”)
8. elibereaza memoria alocata
9. inchide fisierul

void decriptare1(char *numeImagineCr, char *numeImagineSemiDecr, char *numeFisierCheie)

1. deschide imaginea criptata (“numeImagineCr”)
2. extrage header-ul
3. liniarizeaza imaginea
4. apeleaza functia “seed” pentru a extrage numerele “r0” si “sv”

5. apeleaza “XORSHIFT” pentru a genera numerele pseudo-aleatoare
6. tabloul q se obtine folosind operatia “xor” conform algoritmului prezentat
7. creeaza imaginea auxiliara (“numeImagineSemiDecr”)
8. elibereaza memoria alocata
9. inchide fisierul

void decriptare2(char *numeImagineSemiDecr, char *numeImagineDecr, char *numeFisierCheie)

1. deschide imaginea auxiliara (“numeImagineSemiDecr”)
2. extrage header-ul
3. liniarizeaza imaginea
4. apeleaza functia “seed” pentru a extrage numerele “r0” si “sv”
5. apeleaza “XORSHIFT” pentru a genera numerele pseudo-aleatoare
6. se genereaza permutarea pe baza numerelor generate anterior
7. se calculeaza permutarea inversa
8. tabloul “q” se obtine aplicand permutarea inversa tabloului ce contine imaginea auxiliara (“p”), in forma liniarizata
9. creeaza imaginea decriptata
10. elibereaza memoria alocata
11. inchide fisierul

void chipatrat(char *numeImagine)

1. deschide fisierul (“numeImagine”)
2. liniarizeaza imaginea
3. calculeaza frecventa medie (“fmed”)
4. numara frecventa fiecarei culori (“nr1”, “nr2”, “nr3”), culorile obtinandu-se folosind operatii de shiftare
5. calculeaza chi-patrat pentru fiecare canal de culoare in parte (“chi1” – rosu, “chi2” – verde, “chi3” – albastru)
6. afiseaza valorile testului pentru fiecare culoare
7. elibereaza memoria alocata
8. inchide fisierul

Universitatea din Bucureşti
Facultatea de Matematică şi Informatică
Alexiu Adrian-Ştefan

TemplateMatching

```
typedef struct
{
   int i, j;
   int culoare;
   float scor;
} Detectie;
```

Retine coordonatele primului punct al unei detectii, culoarea specifica sablonului cu care s-a potrivit si scorul corelatiei.

void citire(char **numeImagine)

Citeste de la tastatura numele fisierului si il transmite prin intermediul parametrului “numeImagine”.

int cond(const void *a, const void *b)

Ajuta la apelarea functiei “qsort()”, comparand campul “scor” pentru doua variabile de tip “Detectie”. Functia va returna -1 daca scorul primei variabile este mai mare decat celei de-a doua variabila, 0 daca sunt egale si 1 altfel.

void scriere1(unsigned int **p, int w, int h, char *numeImagineNoua, unsigned char *s)

Creeaza o noua imagine in format “.bmp”, avand ca parametrii un tablou unidimensional (“p”), reprezentand o imagine liniarizata, latimea (“w”), inaltimea (“h”), numele imaginii noi si header-ul (“s”).

void grayScale(char* numeImagineColor, char* numeImagineGrayscale)

Converteste o imagine color intr-una in nuante de gri.

void formaLiniarizata(unsigned int ***p, int *w, int *h, char *numeImagine)

Incarca in memorie imaginea “numeImagine” folosind un tablou bidimensional “p”. De asemenea, functia furnizeaza si latimea (“w”) si inaltimea (“h”) imaginii.

double intersectie(int i, int j, int k, int l, int w, int h)

Calculeaza aria intersectiei a doua detectii. Parametrii “i” si “j” reprezinta coordonatele primului punct al primei detectii, iar “k” si “l” coordonatele primului punct al celei de-a doua detectie. Parametrii “w” si “h” reprezinta latimea, respectiv inaltimea unei detectii (dimensiuni care corespund cu cele ale unui sablon).

double suprapunere(Detectie a, Detectie b, int w, int h)

Calculeaza si returneaza suprapunerea dintre doua detectii “a” si “b”. Parametrii “w” si “h” reprezinta latimea, respectiv inaltimea unei detectii (dimensiuni care corespund cu cele ale unui sablon).

void eliminare(Detectie **a, unsigned int *nr, int k)

Elimina elementul de pe pozitia “k” din tabloul de detectii “a”, “nr” fiind numarul total de detectii din tablou, care se modifica la executarea functiei.

unsigned int color(unsigned int r, unsigned int g, unsigned int b)

Calculeaza si returneaza sub forma de intreg culoarea folosind 3 valori intregi: “r”, “g”, “b”.

void patrat(unsigned int ***p, int i, int j, int w, int h, unsigned int culoare)

Coloreaza o fereastra dintr-un tablou bidimensional “p” intr-o anumita culoare, incepand cu elementul de pe pozitia (i, j). Parametrii “w” si “h” reprezinta latimea, respectiv inaltimea unei ferestre (dimensiuni care corespund cu cele ale unui sablon).

unsigned int alegereCuloare(char *numeSablon)

Returneaza culoarea respectiva fiecarui sablon, in functie de nume.

float deviatiaStandardFereastra(unsigned int **p, int w, int h, int i, int j)

Calculeaza deviatia standard conform formulei pentru o fereastra dintr-un tablou bidimensional “p”, ce reprezinta o imagine in forma liniarizata, incepand cu elementul de pe pozitia (i, j). Parametrii “w” si “h” reprezinta latimea, respectiv inaltimea unei ferestre (dimensiuni care corespund cu cele ale unui sablon).

float deviatiaStandardSablon(unsigned int **p, int w, int h)

Calculeaza deviatia standard conform formulei pentru un tablou bidimensional “p”, ce reprezinta un sablon in forma liniarizata. Parametrii “w” si “h” reprezinta latimea, respectiv inaltimea unui sablon.

float corr(unsigned int **pimg, unsigned int **psbl, int i, int j, int w, int h)

Calculeaza si returneaza corelatia dintre o fereastra dintr-un tablou bidimensional (“pimg”), ce reprezinta o imagine in forma liniarizata, incepand cu elementul de pe pozitia (i, j) si un tablou bidimensional (“psbl”) ce reprezinta un sablon in forma liniarizata. Parametrii “w” si “h” reprezinta latimea, respectiv inaltimea unei ferestre (dimensiuni care corespund cu cele ale unui sablon).

void eliminareaNonMaximelor(Detectie **D, unsigned int **pimg, int wimg, int himg, int wsbl, int hsbl, float prag, unsigned int *nr)

Urmeaza algoritmul pentru eliminarea non-maximelor din tabloul “D” de detectii cu “nr” elemente. Parametrii “pimg”, “wimg”, “himg”, “wsbl”, “hsbl” reprezinta o imagine in forma liniarizata, latimea, si inaltimea acesteia, respectiv latimea si inaltimea unui sablon. Parametrul “prag” reprezinta pragul minim pe care trebuie sa il atinga calculul corelatiei unei ferestre pentru a fi adaugata in tabloul “D”.

void templateMatch(char *numeImagineGrayscale, char *numeImagine, float prag, char *numeImagineRezultat)

1. liniarizeaza atat imaginea grayscale formata (“numeImagineGrayscale”) cat si imaginea color (“numeImagine”)
2. numara ferestrele al caror corelatie depaseste pragul impus
3. apeleaza functia de eliminare a non-maximelor
4. alege culoarea si, in functie de culoare, se contruieste o fereastra in imaginea color
5. apeleaza functia scriere pentru a creea imaginea nou-formata (“numeImagineRezultat”)
6. se elibereaza memoria pentru tablourile alocate dinamic