Facultatea de Matematica si Informatica

Universitatea Bucuresti

Proiect Programare Procedurala 2018

Ilicea Anca-Stefania

Grupa 143

Proiectul contine doua module: modulul de criptare si decriptare respectiv modulul de recunoastere de patternuri.

Modulul „Template Matching”

Modulul de recunoastere de patternuri intr-o imagine

Contine 8 functii:

- incarcareImg2
- Grayscale
- writeBmpOut2
- corelatie
- eliminareMaxime
- initializareCulori
- desenareDrepunghiuri
- compararеDetectii

2 strucruri px,fi

functia main

Prezentarea functiilor

void incarcareImg2(char *bmpIn, unsigned int *H, unsigned int *W, struct px ***liniarizare, unsigned char **header)

- incarca headerul in memorie

- incarca in memoria interna, respectiv in matricea linearizare pixelii din imaginea test.bmp

void writeBmpOut2(struct px **liniarizare, char *bmpOut1, char *bmpIn, unsigned char *header, unsigned int latime_img,unsigned int inaltime_img)

- deschide fisierul binar in care "vom scrie imaginea"

- salveaza in memoria externa imaginea(scrie initial headerul,apoi vectorul linearizare)

Functia corelatie

```
void corelatie(struct px **liniarizare, struct px **cifra, double ps, struct fi **D, struct fi *fiCurent,
               unsigned int H, unsigned int W, unsigned int H1, unsigned int W1, unsigned int *numarDetectii,unsigned int *q) {
```

Parametrii:

- linearizare-matricea de pixeli ai imaginii "test.bmp"
- cifra - matricea de pixeli ai sablonului folosit
- ps - pragul minim
- D - vectorul in care vor fi retinute toate detectiile
- fiCurent - retine detectia curenta
- H - inaltimea imaginii "test.bmp"
- W-latimea imaginii "test.bmp"
- H1-inaltimea sablonului
- W1-latimea sablonului
- numarDetectii-retine numarul total de detectii
- q-este un indice care va ajuta la detectarea culorii(q=0 reprezinta culoarea specifica pentru desenarea dreptuncghiurilor cifrei 0,q=1 reprezinta culoarea specifica pentru desenarea dreptuncghiurilor cifrei 1 etc),fiecare detectie va avea un camp special in structura intitulat culoare care va retine culoarea cu care trebuie desenat dreptunghiul,aceasta va fi accesata prin intermediul vectorului de structuri **CULOARE**)

Functia corelatie: gliseaza sablonul apelat peste imaginea principala “test.bmp” avand ca punct de start pentru fiecare fereastra punctul din stanga sus de coordonate u(x,y). In ceea ce urmeaza se calculeaza valorile pentru fi,pentru fiecare pixel, iar mai apoi se face suma tuturor acestor valori in s mediu, fi mediu, deviatia standard s si deviatia standard fi. Apoi se calculeaza corelatia pentru fereastra conform formulei parcurgand din nou matricea sablonului:(agaiga formula)

- daca corelatia ferestrei este mai mare decat pragul standard(variabila ps cu valoarea 0.5)
- ferastra este detectie prin urmare
- adaugam intr.o variabila de tip structura principalele specificatii ale detectiei(coordonatele coltului din stanga sus(fiCurent.x,fiCurent.y),valoarea corelatiei,latimea si inaltimea ferestrei, o variabila semafor care va retine true deoarece fereastra este detectie si variabila q care va retine indexul culorii.
- fiCurent este adaugat in vectorul cu detectii pe pozitia numarDetectii care va creste

Functia initializareCulori

```
void initializareCulori(struct px **culoare) {
```

Parametrii folositi:

- vectorul de structuri px care va retine cele 10 culori cu care vor fi desenate contururile detectiilor

- initializeaza canalele r,g,b cu valorile potrivite pentru fiecare structura din vector

Functia desenareDreptunghiuri:

```
void desenareDreptunghiuri(struct px ***liniarizare, struct fi D, struct px culoare, unsigned int numarDetectii) {...}
```

Parametrii:

- matricea linearizare care se ca modifica pe parcursul functiei(am folosit *)

- D, vectorul cu detectii

- Numarul de detectii

- Vectorul de culori

Functia :

-salveaza initial coordonatele punctului din stanga sus

-parcurge cu un for latura de sus si latira de jos(adaug 15 deoarece aceasta este inaltimea imaginii), si coloreaza toti pixelii cu o culoare precizata

-parcurge cu al doilea for marginile din stanga si din dreapta (adauga 11 deoarece aceasta este latimea imaginii si am nevoie de y+11 pixeli pentru a ajunge la marginea din dreapta)

compararеDetectii:

Functia foloseste 2 structuri si le compara corelatiile respectiv: daca corelatia detectiei 1 este mai mare decat corelatia detectiei 2 atunci returneaza -1, daca subt egale returneaza 0 si daca corelatia detectiei 1 este mai mica decat corelatia detectiei 2 atunci returneaza 1

Functia va fi folosita in cadrul qsort

Functia eliminareMaxime

```
void eliminareMaxime(struct fi *D, struct px ***liniarizare, struct px *culoare, unsigned int numarDetectii, unsigned int H1,
                     unsigned int W1,unsigned int *contorSuprapuneri,struct fi **Dsuprapuneri) {...}
```

Parametrii:

- matricea linearizare care se ca modifica pe parcursul functiei(am folosit *)

- D, vectorul cu detectii

- Numarul de detectii

- Vectorul de culori

- inaltimea imaginii salbon

- latimea imaginii sablon

- contorSuprapuneri, indexul vectorului care va retine ferestrele cu corelatii macime

- Dsuprapuneri retine ferestrele cu cofelatii maxime

- parcurge cu 2 foruri vctorul D

- retine ariile celor 2 detectii

- Pentru cele 2 dreptunghiuri afla coordonatele coltului maxim (cu x maxim si y maxim) din dreapta.

- X maxim este reprezentat de R1Top si y maxim de R2Top.

- afla coordonatele coltului minim (cu x minim si y minim) din stanga

- Calculeaza aria intersectiei cu ajutorul formulei:

$$suprapunere(d_i, d_j) = \frac{aria(d_i \cap d_j)}{aria(d_i \cup d_j)} = \frac{aria(d_i \cap d_j)}{aria(d_i) + aria(d_j) - aria(d_i \cap d_j)}$$

- Afla daca cele doua dreptunghiuri se intersecteaza prin intermediul formulei:(insereaza formula)

- Calculeaza suprapunerea dupa ce verifica daca nu se efectueaza impartirea la 0

- Daca suprapunerea este mai mare decat 0.2 atunci detectia este adaugata in vectorul Dsuprapuneri

Modulul de criptare si decriptare

In ceea ce urmeaza se vor prezenta pe rand functiile folosite:

Functia CitireLinearizare

```
void CitireLiniarizare(char *bmpIn, unsigned char **header, struct px **liniarizare, unsigned int *latime_img,
                       unsigned int *inaltime_img) {...}
```

Parametrii

- bmpIn-Calea imaginii originale "peppers.bmp"
- H-inaltimea imaginii "peppers.bmp" care este preluata cu "*" deoarece valoarea acesteia se va modifica pe parcursul functiei si se va pastra si in main
- W-latimea imaginii "peppers.bmp" care este preluata cu "*" deoarece valoarea acesteia se va modifica pe parcursul functiei si se va pastra si in main
- liniarizare-retine matricea linearizata
- header-retine primii 54 de octeti ai fisierului binar "peppers.bmp"

- efectueaza urmatoarele operatii

  - deschide fisierul binar, valideaza deschiderea corecta

  - citeste headerul respectiv H si W

  - citeste pixelii de la octetul 54 si ii adauga in memoria interna

Functia writeBmpOut2

void writeBmpOut2(struct px **liniarizare, char *bmpOut1, char *bmpIn, unsigned char *header, unsigned int latime_img,unsigned int inaltime_img)

-deschide fisierul binar in care "vom scrie imaginea"

-salveaza in memoria externa imaginea(scrie initial headerul,apoi vectorul linearizare)

Functia xorshift32

```
unsigned int xorshift32(unsigned int seed) {...}
```

Parametrii:seed, valoarea din cadrul fisierului secret_key.txt cu care va incepe generarea numerelor

Functia genereaza numere pseudoaleatoare pe baza xorarii si shiftarii(algoritm conceput de George Marsaglia)

Functia criptare

```
void criptare(char *bmpIn, char *bmpOut, char *secretKey, struct px **liniarizare, unsigned int latime_img,
              unsigned int inaltime_img, unsigned int *R0, unsigned int SV) {...}
```

Parametrii:

- bmpIn: calea imaginii originale
- bmpOut: calea fisierului in care se va scrie imaginea criptata
- Secretkey: calea fisierului cu R0 si SV
- Liniarizare:vectorul ce contine matricea de pixeli linearizata
- Latimea si inaltimea imaginii
- R0 si SV

Functia:

- Genereaza o un sir de dimR numere aleatoare
- Initializeaza permutarea cu valori de la 0 la dim R
- Reinitializeaza vectorul permutare in functie de valorile generate de xorshift32
- Efectueaza permutarea prin intermediul unei permutari auxiliare
- R0 este reinitializat pentru a putea merge mai departe cu noua valoare
- Functia preia pe rand pixelii vectorului linearizare pentru a-i xora
- Este calculat initial primul element prin xorarea valorii de start (SV) cu linearizare [i]
- Foloseste o copie a valorii SV
- Este descompus SV (intreg pe 32 de biti) folosindu-se pe rand ultimii 8 octeti pentru a xora canalul blue, urmatorii 8 pentru a xora canalul g si urmatorii 8 pentru a xora canalul r, ultimii 8 octeti nu sunt utilizati
- Analog pentru xorarea valorii obtinute cu R0

In acest punct xorarea primului element a fost terminata si, deoarece formula pentru xorarea celorlalte elemente este diferita asadar se va accesa “else”. Deoarece este vorba despre xorarea a doi pixeli nu va mai putea fi folosita aceeasi metoda ca anterior.

$$C_k = \begin{cases} SV \oplus P_0' \oplus R_{W*H}, & k = 0 \\ C_{k-1} \oplus P_k' \oplus R_{W*H+k}, & k \in \{1,2,\dots,W*H-1\} \end{cases}$$

In acest caz sunt xorate pe rand valorile de pe fiecare canal de culoare a pixelului curent cu valorile pixelului precedent.

In contiruare valorile de pe canalele de culoare ale rezultatului vor fi xorate pe rand cu cate 8 biti din R0

R0 este reinitializat pentru a putea merge mai departe

Functia decriptare:

Functia functioneaza in mod invers fata de functia de criptare generand si permutarea inversa

Formula:

$$C_k' = \begin{cases} SV \oplus C_0 \oplus R_{W*H}, & k = 0 \\ C_{k-1} \oplus C_k \oplus R_{W*H+k}, & k \in \{1,2,\dots,W*H-1\} \end{cases}$$

Functia chiPatrat:

Functia realizeaza o analiza a frecventelor culorilor pe fiecare canal de culoare

Initial este calculata frecventa estimata reoretic (frecv2)

Pentru canalul de culoare B:

- Este construit un vector de frecventa care retine frecventa fiecarei valori intre 0 si 255
- Este parcurs vectorul de frecventa si se realizeaza suma conform formulei:

Analog pentru canalele G si R