Racoviță Andra-Georgiana
Grupa 132

Documentație proiect

Tema proitectului

- În prima parte este Criptarea și Decriptarea unei imagini
- În a doua parte este Recunoașterea unor pattern-uri într-o imagine

În acest proiect se folosesc doar imagini in format BMP(bitmap). Formatul BMP nu comprimă imaginile, ci le stochează sub formă de pixeli pe 3 octeți.

I. Criptarea și decriptarea unei imagini bmp

Criptarea este folosită pentru a asigura confidențialitatea datelor, astfel încat imaginea să poată fi accesată doar de către persoanele autorizate. Prin procesul de criptare imaginea devine una imposibil de înțeles. Avand un algoritm simetric, procesul de criptare și decriptare se realizează cu ajutorul unei **chei secrete comune**.
Pentru a realiza prima parte am folosit următoarele funcții și structuri:

- **Dimensiuni_imagine** are ca parametrii calea imaginii si furnizează lungimea, lățimea și padding-ul
- Pentru a prelucra datele imaginii am folosit o structură numita **Pixel,** care conține canalele **RGB**
- **XORSHIFT32** creează un vector de lungime 2*W*H cu numere pseudo-aleatorii având ca seed primul număr din fișierul text secret_key.txt
- **Algoritmul_lui_Durstenfeld** generează, cu ajutorului vectorului obținut anterior, permutari aleatoare de lungime W*H
- **Liniarizare** citește pixelii dintr-o imagine și îi pune într-un vector
- **Salvare_imagine_forma_liniarizata** folosește funcția Liniarizare si face o copie a imaginii inițiale
- **Permutare_pixelii_imaginii** creează o nouă imagine care are pixelii permutați. Pentru a obține noua imagine, se folosește Algoritmul lui Durstenfeld, dar și vectorul obținut prin Liniarizare. Pixelii din imaginea inițială sunt poziționați după permutarea obținută.
- **Criptare** utilizează imaginea obținută prin permutare schimbând valoarea pixelilor cu ajutorul funcțiilor:
    - **P_XOR_X** o funcție care aplica xor intre un pixel P și cei mai semnificativi octeți ai numărului X
    - **Pi_XOR_Pj** o funcție care aplica xor intre doi pixeli Pi și Pj
- **Decriptare** incepem prin a aduce Pixelii la valoriile originale utilizand XOR și după folosim permutarea inversă pentru a replasa pixelii la pozițiile inițiale
- **Testul_chi_patrat** calculeaza frecvența valorilor pe fiecare canal de culoare

II. Recunoașterea unor pattern-uri într-o imagine

Problema de recunoaștere a cifrelor scrise de mână într-o imagine este tema acestei părți. Pentru a realiza acest lucru vom folosi metoda glisării de șabloane pe imaginea inițială și vom găsi părți mici ale unei imagini care se potrivesc cu imaginea șablon.

Racoviță Andra-Georgiana
Grupa 132

Pentru a realiza a doua parte am folosit următoarele funcții și structuri:

- Pentru a salva datele pe care la obțin din prelucrarea imaginilor, folosesc o structură numită **Coordonate** care conține Ox și Oy(coordonate), corelația și template (ce reține numele șablonului)
- **Matrice_pixeli** face o matrice cu pixelii unei imagini dupa citire
- **Matrice_pixeli_inversa** face o matrice cu pixelii unei imagini, însă aceasta are pixelii așezați conform imaginii: ultimul pixel citit este primul în matrice
- **Corelatie** returnează valoarea corelației dintre un șablon și o fereastră a imaginii
- **Grayscale_image** transformă o imagine color în una alb-negru
- **Template_matching** tranformă inițial imaginile în imagini alb-negru. După, pentru fiecare pereche de coordonate x și y calculează corelația și o verifică cu pragul ps=0,5. Valorile care au corelația mai mare decât pragul ps sunt salvate într-un vector.
- **Colorare_fereastra** bordează în imaginea I o fereastră de dimensiunea șablunui S
- **Creare_vector_Detectii** citesc imaginea și șabloanele, iar după rulez template-matching pentru toate șabloanele
- **Cmp** o folosesc pentru a sorta vectorul descrescător cu qsort
- **Aria** calculează aria
- **Suprapuner** returnează procentul de suprapunere a două ferestre
- **Eliminare_non_maxime** din vectorul cu detecții le eliminăm pe cele care se suprapun cu alte detectii(cu suprapunera mai mare sau egala cu 0.2) și au corelația mai mică