PROIECT PROGRAMARE PROCEDURALĂ

Proiectul este împărțit în două fișiere, primul numit criptare.c, în care am scris funcțiile necesare pentru criptarea și decriptarea imaginii date, iar al doilea, numit template.c, în care am scris funcțiile pentru template matching. Am inclus, apoi cele două fișiere într-un main, unde doar am apelat funcțiile necesare.

Algoritmul de criptare se desfășoară astfel: în funcția criptare am citit prima dată seed și R0 din fișierul text dat, apoi am folosit generatorul de numere aletoare XORSHIFT32, inițializat cu valoarea nenulă $R0$ pentru a genera o secvență de 2*W*H-1 numere aletoare întregi fără semn. Se generează o permutare aleatoare σ de dimensiune $W \times H$, folosind algoritmul lui Durstenfeld, după care se permută pixelii imaginii P conform permutării σ, obținându-se o imagine intermediară L_perm. Imaginea criptată C se obține aplicând asupra fiecărui pixel al imaginii L_perm relația de substituție dată. Se extrag octeții 0,1,2 din valorile necesare și se aplică operația xor.

Pentru algoritmul de decriptare se folosește funcția decriptare, în care se generează o secvență R de 2*W*H-1 numere întregi aleatoare fără semn pe 32 de biți, folosind generatorul XORSHIFT32, inițializat cu valoarea nenulă $R0$. Apoi, se generează o permutare aleatoare σ de dimensiune $W \times H$, folosind algoritmul lui Durstenfeld și se calculează inversa sa. se aplică asupra fiecărui pixel din imaginea criptată C inversă relației de substituție folosită în procesul de criptare, obținându-se o imagine intermediară $C_$inter. Imaginea decriptată D se obține permutând pixelii imaginii $C_$inter conform permutării inverse.

La final, se calculează valoarea lui chi pătrat pentru imaginea inițială și pentru cea criptată, folosing 3 vectori de frecvență pentru fiecare canal R,G,B.

Pentru template matching, am pornit prin a extrage dimensiunile imaginii și ale șablonului. Apoi, am apelat funcția template_matching, care citește din fișierul „date_template” numele imaginilor și șabloanelor. Apoi, am salvat imaginea inițială, atât greyscale, cât și color, în două matrici. Pentru fiecare șablon cu cifre de la 0 la 9 am citit denumirea din fișier, am transformat în greyscale și apoi am apelat funcția detecții, care extrage câte o fereastră din matrice, îi calculează corelația după formula dată și dacă este mai mare decât pragul de 0.5, atunci colorează fereastra și salvează coordonatele colțului din stânga sus, corelația și valorile culorii cu care a fost desenat în vectorul d de tip struct detecție. Fereastra este colorată pe prima și ultima linie, prima și ultima coloană, cu culoarea dată, memorată într-un vector C de tip struct pixel. La final se scrie imaginea cu detecțiile și se aplică funcția de eliminare a non-maximelor. Aceasta se realizează prin ordonarea descrescătoare după corelație a valorilor din vectorul d, iar dacă două detecții se suprapun (acest lucru se află cu ajutorul funcției overlapping, care primește coordonatele colțului din stânga sus a două puncte, calculează coordonatele colțului din dreapta jos și calculează suprapunerea spațială a două ferestre), atunci se păstrează cea cu corelația mai mare, care este din nou marcată într-o altă imagine. Se scrie nouă imagine la final.