Documentatie Proiect

# Structuri folosite

1. Pixel, o uniune din o structură ce conține 3 câmpuri R, G, B un tablou de unsigned char colors și un char gray pentru a putea accesa în mai multe moduri octeții pixelilor: pe culori, ca tabloul de 3 octeți, ca intensitate de gri aca imaginea e grayscale
2. Image, structură ce conține lățimea, lungimea, mărimea, header, pointer spre tablou liniaraziat și pointer dublu, care reprezintă matricea. Matricea și tabloul liniarizat arată spre acceași valori, ce se schimbă în unul se schimbă și în celălalt.
3. Point, punct, conține coordonatele x,y.
4. Window, conține un punct de mijloc, mărimea ferestrei, și pointer spre întreaga imagine
5. Detection, detecție, conține punctul de mijloc, din stânga sus și dreapta jos, tipul – cifra detectată, acuratețea între -1 și 1, și câmp good care semnifică dacă e maximă sau nu.
6. DetectionArr, tablou de detecții ce conține și dimensiunea lui.

# Funcții folosite

1. void deconstructImage(Image\_ptr ); Eliberează memoria folosită de o imagine în RAM
2. Image\_ptr loadBMPImage(char\* path); Incarcă imagine în memorie, setează toate câmpurile din structura Image și întoarce un pointer la aceasta
3. void saveBMPImage(Image\_ptr img, char \*path); Salvează imaginea bmp din memorie RAM în memoria externă
4. void loadSecretKeys(char \*keyPath, uint\* key\_1, uint\* key\_2); Citește cheile din fișier și le returnează ca parametri
5. uint \*xorShift32(uint seed, uint size); Returnează un tablou de mărimea site t cu numere aleatoriu generate
6. uint \*getRandomPermutation(uint\* randomArr, uint size); Construiește și returnează o permutare aleatorie după algoritmul lui Durtnsfield și numere aleatorii deja generate
7. void permutePixels(Image\_ptr img, uint\* randomPermutation); Permută pixelii dintr-o imagine conform permutării primite ca parametru
8. void dePermutePixels(Image\_ptr img, uint\* randomPermutation); Face permutare inversă a imaginii conform permutării primite ca parametru
9. Pixel numberToPixel(uint nr); Convetește un număr unsigned int de 32 biți în pixel astfel încât primul octet va fi culoare B, apoi R apoi G iar ultimul nu este folosit
10. void encryptImage(char \*initial\_img\_path, char \*encrypted\_img\_path, char \*secret\_keys\_path); Criptează imaginea inițială și o salvează în calea propusă
11. void xorEncryptImage(Image\_ptr img, uint\* randArr, uint key); Face xor la pixelii din imagine conform formulei
12. void xorDecryptImage(Image\_ptr img, uint\* randArr, uint key); Face xor invers la pixelii din imagine conform formulei
13. void showPixel(Pixel p); Funcție de debug, afisează valorile unui pixel la consola
14. double chiSquaredTest(Image\_ptr img, uchar channel); Face testul chi squared pentru canalul channel și returnează valoarea
15. void chiTest(char \*path); Face testul chi squared pe toate 3 canale a unei imagini și afisează la consolă
16. void toGrayscale(Image\_ptr img); Convertește imaginea în grayscale
17. double avgGrayIntensity(Window window); Calculează intensitatea medie de gri
18. void buildMatrixImg(Image\_ptr img); Nu este folosită. Construia o matrice de pixeli folosind o copie a valorilor din tablou liniarizat
19. void buildSmartMatrixImg(Image\_ptr img); Construiește o matrice de pixeli compactă după cum a fost descris la struct Image
20. double standardDeviation(Window window, double avg); Calculează deviata standard pentru o anumită fereastră din imagine după formulă și returnează valoarea
21. double crossCorrelation(Window template, Window target); Calculează cross correlation între template și sablon și returnează o valoare între -1 și 1
22. void drawDetection(Pixel \*\*M, Detection detection); Deseanează conturul la o detective
23. Pixel getColor(int number); Returnează o culoare pentru fiecare cifră, după cum e descris în proiect
24. Detection convertDetection(Window targetWindow, int nr, double accuracy); Convertește o fereastră într-o detective calculând toate campurile și inițializând good cu 1
25. DetectionArr getDetections(Image\_ptr target, char\* path\_to\_template, int nr, double threshold); Returnează o structură ce conține un tablou alocat dynamic de detecții între target și sablonul din path\_to\_template socotind ca fiind cifra nr și le selectează doar pe cele cu acuratețe minimă de accuracy
26. void copyArray(DetectionArr \*destination, DetectionArr \*source); Unește 2 tablouri dynamic în destination și eliberează memorie din source
27. DetectionArr getAllDetections(Image\_ptr img, char paths[][BUFFER\_SIZE], int nr); Returnează un tablou cu toate detecțiile dintre imaginea img și imaginile aflate în paths care urmează a fi încarcate în memorie
28. double overlap(Detection a, Detection b); Calculează raportul de suprapunere dintre 2 ferestre detectate.
29. int intersect(Detection a, Detection b); Returnează 1 dacă ferestrele se intersectează, 0 altfel
30. DetectionArr removeOverlapping(DetectionArr detArr); Sterge detecțiile non maxime din tablou și îl returnează

**Continuarea in README.md**