UNIVERSITATEA DIN BUCUREȘTI
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

PROCESAREA SEMNALELOR

Image Watermarking

Echipa LOR:
Codreanu Radu-Ștefan - 461
Albei Liviu-Andrei - 461
Cojocaru Andrei-Laurențiu - 461

2024

Cuprins

I. Problema abordată în proiect

- Eficiența algoritmilor: Alegerea și implementarea de algoritmi eficienți pentru adăugarea și detectarea watermark-ului fără a afecta semnificativ performanța sistemului.
    - **Performanța sistemului:** Algoritmii eficienți permit adăugarea și detectarea rapidă a watermark-ului fără a afecta semnificativ performanța sistemului. Acest aspect devine crucial în aplicații în timp real sau în medii în care resursele sunt limitate, precum dispozitivele mobile.
    - **Scalabilitate:** Cu creșterea volumului de conținut digital, este esențial să se implementeze algoritmi care să poată scala eficient și să gestioneze un număr mare de imagini fără a compromite viteza sau calitatea procesului de watermarking.

- Vizibilitate vs. Invizibilitate: Găsirea unui echilibru între a face watermark-ul vizibil (de exemplu, un logo sau un text) și a îl face invizibil sau dificil de detectat pentru a nu afecta estetica imaginii.

    - **Protecția drepturilor de autor:** Un watermark vizibil poate servi ca o modalitate eficientă de a informa privitorii că imaginea este protejată de drepturi de autor, descurajând utilizarea neautorizată a acesteia.
    - **Estetica imaginii:** Dacă un watermark este prea vizibil, poate afecta estetica și experiența de vizualizare. Prin găsirea unui echilibru între vizibilitate și invizibilitate, se poate menține calitatea estetică a imaginii, fără a compromite protecția drepturilor de autor.

- Adăugarea unui Watermark de Text:

    - Utilizatorul alege opțiunea 1.
    - Imaginea color originală este încărcată.
    - Utilizatorul introduce textul watermark-ului.

- **Se cer detalii suplimentare:**
  - Opacitatea watermark-ului (între 0 și 1).
  - Unghiul de rotație al textului.
  - Poziția watermark-ului (sus, jos, stânga, dreapta, centru).
  - Marginea între text și marginea imaginii.
  - Culoarea textului.
- Funcția **add_watermark** este apelată pentru a adăuga watermark-ul de text.
- Imaginea rezultată este salvată.

- Adăugarea unui Watermark LSB (Least Significant Bit):

  - Utilizatorul alege opțiunea 2.
  - Imaginea color originală și imaginea watermark-ului sunt încărcate.
  - Funcția **add_lsb_watermark** este apelată pentru a adăuga watermark-ul LSB la imaginea originală.
  - Imaginea rezultată este salvată.
  - Watermark-ul este extras.
  - Se calculează și afișează harta de entropie locală a imaginii rezultate (atât pentru opțiunea 1 cât și pentru aceasta).

II. Justificarea problemei abordate

Eficiența algoritmilor:

Selectarea algoritmului de inserare a watermark-ului:

Scop: Identificarea unui algoritm robust pentru încorporarea watermark-ului în imagine.
Descriere matematică: Alegerea unui set de transformări și proceduri matematice care să integreze watermark-ul într-un mod sigur și eficient.

Optimizarea algoritmului de detecție a watermark-ului:

Scop: Dezvoltarea unui algoritm eficient pentru a identifica și extrage watermark-ul dintr-o imagine.
Descriere matematică: Definirea unor criterii și metode matematice pentru identificarea watermark-ului într-un mod precis și rapid.

Vizibilitate vs. Invizibilitate:

Determinarea nivelului de vizibilitate:

Scop: Stabilirea gradului de vizibilitate a watermark-ului în funcție de necesități.
Descriere matematică: Ajustarea parametrilor de inserare a watermark-ului pentru a controla gradul de vizibilitate, utilizând funcții matematice care influențează intensitatea și poziția elementelor de marcaj.

Utilizarea tehnicilor de steganografie:

Scop: Implementarea metodelor de inserare a watermark-ului astfel încât să fie dificil de detectat.
Descriere matematică: Aplicarea tehnicilor de steganografie pentru a ascunde informația watermark-ului într-un mod subtil și dificil de identificat vizual sau prin analize automate.

III. Cum este abordată problema tehnic

Adăugarea unui Watermark de Text:

Funcția add_watermark:

Această funcție adaugă un watermark de text pe o imagine. Începe prin a crea o imagine goală (transparentă) pentru watermark, apoi adaugă textul specificat într-o anumită poziție și cu anumite proprietăți, cum ar fi opacitate, rotație și culoare. Imaginea de watermark este apoi aplicată peste imaginea originală, luând în considerare transparența.

Adăugarea unui Watermark LSB (Least Significant Bit):

Ce este LSB (Least Significant Bit)?

Cel mai puțin semnificativ bit (LSB) este cel mai dreapta bit în reprezentarea binară a unui număr. Într-un byte (8 biți), LSB este ultimul bit. Într-o imagine color, fiecare canal de culoare (roșu, verde, albastru) al unui pixel poate fi reprezentat printr-un byte (8 biți). Tehnica LSB înseamnă că informația (cum ar fi un watermark) este ascunsă prin înlocuirea acestui bit cel mai puțin semnificativ cu informația pe care dorim să o ascundem.

Funcția add_lsb_watermark:

Această funcție adaugă un watermark folosind tehnica LSB (Least Significant Bit). Se transformă imaginea watermark în tonuri de gri și se redimensionează pentru a se potrivi dimensiunilor imaginii originale. Apoi, fiecare bit al imaginii de watermark este înglobat în cel mai puțin semnificativ bit al fiecărui pixel al imaginii originale.
Cum se Face Înglobarea în LSB:
Binarizare:

În cazul watermark-ului, prima etapă este să convertim imaginea watermark în imagine binară. Acest lucru se face adesea prin aplicarea unui prag: dacă valoarea unui pixel este sub prag, este setat la 0; altfel, este setat la 1.

Redimensionarea:

Watermark-ul binar este redimensionat pentru a avea aceeași dimensiune cu imaginea originală.

Înglobarea în LSB:

Începem să parcurgem fiecare pixel al imaginii originale și, pentru fiecare canal de culoare, înlocuim LSB-ul cu valoarea corespunzătoare din imaginea binară watermark.

Rezultatul:

Rezultatul este o imagine modificată, care va arăta la fel ca imaginea originală în ochii unei persoane, dar care conține informația ascunsă în LSB-urile pixelilor.

Exemplu:

Luăm un pixel în imaginea originală, cu valoarea RGB: (100, 150, 200). În binar, aceasta ar fi:

R (Roșu): 01100100
G (Verde): 10010110
B (Albastru): 11001000

Fie acum un pixel în imaginea binară watermark: 1.
Pentru a îngloba acest pixel în LSB, vom înlocui ultimul bit al fiecărui canal de culoare:

R (Roșu): 01100101
G (Verde): 10010111
B (Albastru): 11001001
Pixelul înglobat arată similar cu pixelul original, dar conține acum informația din imaginea binară watermark.
Aceasta este esența tehnicilor LSB în adăugarea watermark-ului în imagini.

Funcția extract_lsb_watermark:

Această funcție extrage watermark-ul dintr-o imagine care a fost supusă tehnicii LSB. Se extrage cel mai puțin semnificativ bit al fiecărui pixel, apoi se aplică egalizarea histogramei și un prag pentru a face watermark-ul mai vizibil.

Funcția calculate_entropy:

Această funcție calculează entropia unui semnal. În contextul acestei aplicații, entropia reprezintă măsura diversității sau dezordinei semnalului (zgomot).

Funcția local_entropy:

Această funcție calculează entropia locală a unei imagini. Pentru fiecare pixel din imagine, se formează o vecinătate și se calculează entropia locală în acea zonă. Acest lucru oferă o măsură a variației texturii în diferite părți ale imaginii.

IV. Tehnologiile folosite:

Limbaj de Programare: Python

Justificare: Python oferă o sintaxă concisă și ușor de înțeles, facilitând dezvoltarea și testarea algoritmului. Este o alegere ideală pentru astfel de proiecte, rapid și capabil de dezvoltare eficientă.

Manipularea Imaginilor: Pillow

Justificare: Deși obiectivul este de a dezvolta algoritmul de watermarking fără biblioteci, utilizarea Pillow pentru manipularea imaginilor este justificată pentru operațiuni precum încărcarea, salvarea și afișarea acestora. Aceasta oferă un set simplu de instrumente pentru manipularea pixelilor din imagini.

Utilizare cod:

- Pentru a încărca imaginea originală și a o converti la tonuri de gri.
- Pentru a realiza operații de conversie între modurile de culoare ale imaginilor.

Interfața Utilizator: Matplotlib

Justificare: Pentru vizualizarea rezultatelor și analiză imagini într-un mod interactiv, Matplotlib oferă funcționalități utile pentru afișarea imaginilor în cadrul mediului de dezvoltare Python.

Utilizare cod:

- Pentru a vizualiza imaginile și rezultatele procesării acestora, inclusiv afișarea imaginii originale, a imaginii cu watermark și hărții de entropie.

Manipularea Datelor: Numpy

Justificare: NumPy este o bibliotecă fundamentală pentru calculul științific în Python. Aceasta oferă suport pentru lucrul cu array-uri și matrice multidimensionale, esențial pentru manipularea datelor de imagine.

Utilizare cod:

- În mai multe locuri din cod pentru manipularea datelor sub formă de array-uri, inclusiv în procesarea imaginilor și calculul entropiei.

Prelucrarea Imaginilor: OpenCV (cv2)

Justificare: OpenCV (Open Source Computer Vision Library) este o bibliotecă open-source specializată în prelucrarea imaginilor și a vizionării computerizate. Este utilizată pentru încărcarea, manipularea și procesarea imaginilor în aplicație.

Utilizare cod:

- În funcția **add_watermark** pentru a desena textul watermark pe o imagine, a efectua rotații și a aplica transparența.
- În funcția **add_lsb_watermark** pentru a redimensiona imaginea watermark la dimensiunile imaginii originale și a efectua operații pe canalele de culoare ale imaginii.
- În funcția **extract_lsb_watermark** pentru manipularea și extragerea bitului cel mai puțin semnificativ al fiecărui pixel din imagine.

V. Rezultate

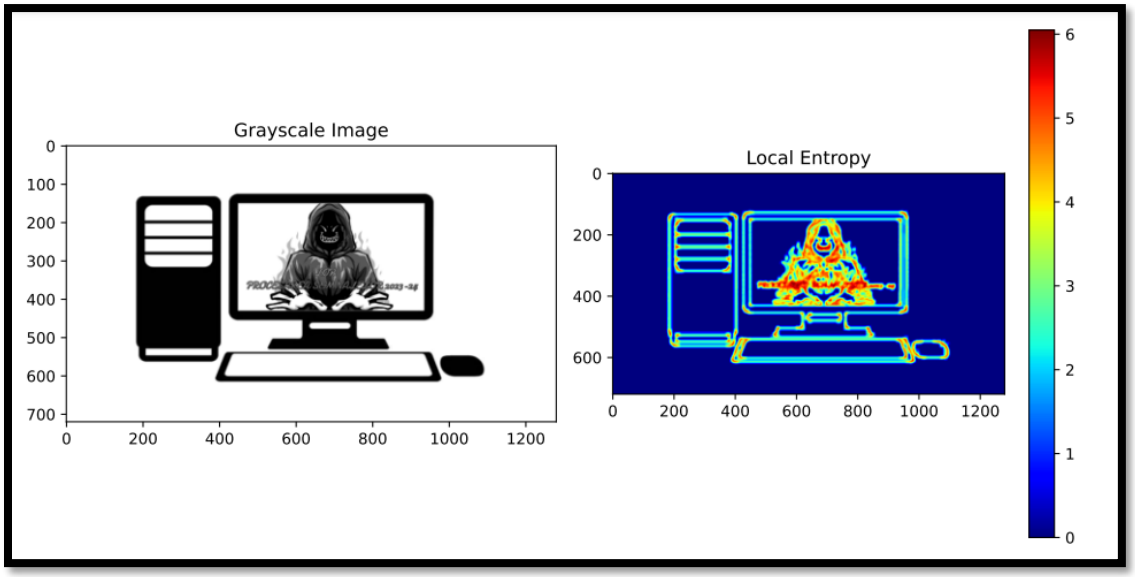
În urma rulării codului am obținut următoarele rezultate:

Imaginea inițială:

Imagine rezultată în urma inputului următor:

```
PS C:\Users\radus\OneDrive\Desktop\psProject> python -u "c:\Users\radus\OneDrive\Desktop\psProject\project.py"
Choose watermarking method: 1 for Text Watermark, 2 for LSB Watermark: 1
Overall Entropy of the grayscale image: 1.8001008986906921
Enter the watermark text: Output
Enter the opacity (0 to 1): 0.65
Enter the rotation angle (in degrees): 90
Enter the watermark position (top, bottom, left, right, center): left
Enter the margin (in pixels) for the watermark position: 5
Enter the text color (red, blue, white, black, yellow, green): red
Watermarked image saved at: result_image.jpg
PS C:\Users\radus\OneDrive\Desktop\psProject>
```

Entropia:



Rezultatul calculului:

```
Overall Entropy of the grayscale image: 1.8001008986906921
```

În urma rulării programului cu choice = 2 (LSB Watermarking) imaginea rezultată este asemanătoare cu ochiul liber, watermark-ul fiind ascuns în ultimul bit al fiecărui pixel al pozei. În urma folosirii imaginii inițiale și a celei de watermark (Figura 1), rezultatul decriptării este următorul. (Figura 2)

Figura 1

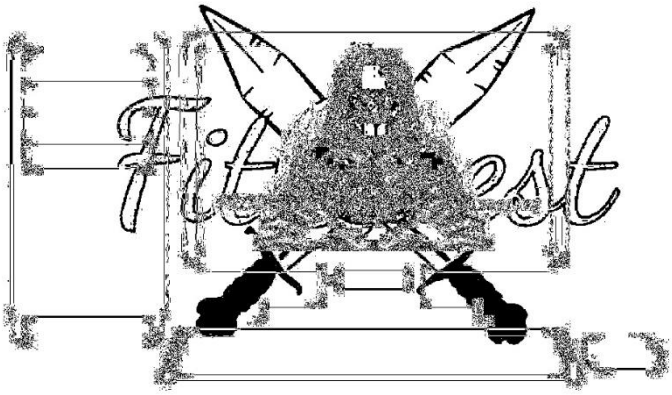


Figura 2

VI. Concluzii

Watermarking-ul este un proces esențial în domeniul gestionării drepturilor digitale și al securității informațiilor. El permite inserarea unei mărci invizibile sau greu de detectat într-un conținut digital, cum ar fi o imagine, un videoclip sau un fișier audio, pentru a indica proprietatea, originea sau alte informații relevante.

Astfel, dintre motivele importante amintim:

- Protecția Drepturilor de Autor
- Integritate
- Robusteză:
- Simplicitate și Eficiență

În concluzie, watermarking-ul, în special prin metoda LSB, este un instrument valoros pentru protecția, autentificarea și gestionarea drepturilor digitale. Oferă un mod eficient și discret de a încorpora informații suplimentare în conținutul digital, menținând în același timp integritatea și calitatea originală a acestuia.

VII. Bibliografie

- https://stats.stackexchange.com/questions/235270/entropy-of-an-image
- https://www.hdm-stuttgart.de/~maucher/Python/MMCodecs/html/basicFunctions.html
- https://en.wikipedia.org/wiki/Steganography
- https://wiki.bi0s.in/forensics/lsb/
- https://medium.com/@renantkn/lsb-steganography-hiding-a-message-in-the-pixels-of-an-image-4722a8567046

Link-uri repo github:

- Codreanu Radu-Ștefan - 461: https://github.com/radustefan2311/461-TemeProcesareaSemnalelorCTI (aici)
- Albei Liviu-Andrei - 461: https://github.com/xSuly/ProcesareaSemnalelor (aici)
- Cojocaru Andrei-Laurențiu - 461 : https://github.com/AndreiLaurentiu/TemeProcSemCTI (aici)