Laboratorul 10. Prelucrări de imagini - Steganografie

Steganografie

În transmiterea informației e util să putem trimite un mesaj secret, iar pentru aceasta trebuie să folosim diverse tehnici de codificare. Este, de asemenea, util să trimitem informație care aparent să nu conțină un mesaj ascuns. Steganografia presupune ascunderea unui mesaj sub forma de text, imagine, videoclip în cadrul altui mesaj sub forma text, imagine sau videoclip.

În acest laborator vom vedea cum putem transmite o imagine secretă, ascunsă în cadrul altei imagini (denumită imagine de transport).

Pentru a exemplifica tehnica, vom ascunde o imagine gri, de dimensiuni mai mici în cadrul unei imagini color mai mari. Pentru aceasta vom urma metoda descrisă aici: Digital Image Steganography: An FFT Approach

În mare, metoda constă în a calcula spectrul imaginii de trasnsport și a înlocui o parte dintre coeficienții Fourier cu valorile pixelilor imaginii secrete. Vom investiga ce coeficienți sunt mai potriviți pentru a fi înlocuiți, astfel încât imaginea transmisă să nu prezinte zgomot evident și astfel încât imaginea secretă să reziste, să fie invariantă la modificări ușoare ale imaginii transmise.

Exercițiul 1 -- Modulul și faza coeficienților Fourier [5p]

Primul lucru pe care îl vom investiga este importanța modulului și a fazei coeficienților Fourier (complecși). Fiecare coeficient din spectrul Fourier $s \in S$ este un număr complex $s \in \mathbb{C}$ cu o magnitudine $|s|$ și o fază: $e^{i\phi}$ astfel încât: $s = a + i \cdot b = |s| \cdot e^{i\phi}$.

Pentru aceasta:

1. citim o imagine RGB cu 3 canale pentru fiecare culoare, folosind funcția 'imread' din MATLAB/Octave și pentru simplitate o convertim într-o imagine cu un canal cu nivele de gri (folosim funcția 'rgb2gray'). Pentru a fi totul mai simplu vom converti valorile în tipul 'double' și vom scala imaginea din valorile [0,255] în [0,1]. Afișați imaginea folosind funcția 'imshow'.
2. luăm o imagine și încercăm să îi aflăm spectrul folosind Transformata Fourier Discretă implementată în MATLAB/Octave, cu ajutorul funcției 'fft2'.
3. păstrăm doar modulul coeficienților $|S|$ și vom reface imaginea inițială cu Transformata Fourier Inversă, implementată cu funcția 'ifft2'.
4. păstrăm doar $e^{i\phi}$ și refacem imaginea cu 'ifft2'.
5. normalizați imaginea reconstruită (în urma aplicării 'ifft2', atât cea din spectru, cât și cea din fază), prin scăderea minimului și împărțirea la valoarea diferenței dintre maxim și minim, astfel încât valorile finale să fie în intervalul [0,1].
6. verificăm vizual care din cele 2 variante a păstrat mai multă informație. Pentru vizualizarea unei imagini care are valori în [0,1] folosim funcția 'imshow' (alternativ puteți folosi 'imagesc' pentru o imagine nescalată).

În MATLAB folosiți **impixelinfo** pentru a obține informații detaliate despre pixelii unei imagini.

Vom observa că informația este păstrată în principal în una dintre cele 2, așa că putem să schimbăm cealaltă variantă înlocuind-o cu valorile imaginii pe care vrem să o ascundem.

Puteți folosi această imagine pentru acest exercițiu: 

Exercițiul 2 -- Decodificarea unei imagini ascunse [5p]

Descărcați imaginea de aici: imagine. Pentru a decodifica imaginea ascunsă, citiți imaginea cu 'imread' și scrieți o funcție după următorii pași:

1. calculați spectrul Fourier (fft2) al imaginii primite. Vizualizați modulul spectrului folosind funcția 'imagesc'. De exemplu:

   ```
   figure, imagesc(log(abs_S));
   ```

   Logaritmăm pentru a atenua diferențele prea mari din spectru care ne îngreunează vizualizarea.

2. observați vizual că în spectru apare o imagine ascunsă. De ce apare imaginea oglindită de 3 ori?
3. selectați doar imaginea observată în spectru. Vizualizați imaginea folosind 'imshow'.
4. normalizați doar imaginea observată în spectru prin scăderea minimului și împărțirea la valoarea diferenței dintre maxim și minim astfel încât valorile finale să fie în intervalul [0,1]. Vizualizați imaginea folosind 'imshow'.

Exercițiul 3 -- Codificarea unei imagini ascunse în spectrul unui imagini de transport [Bonus]

Scrieți o funcție care primește două imagini gri (două matrici), o imaginea secretă și o imagine de transport de cel puțin două ori mai mare și codifică imaginea secretă în spectrul imaginii de transport.

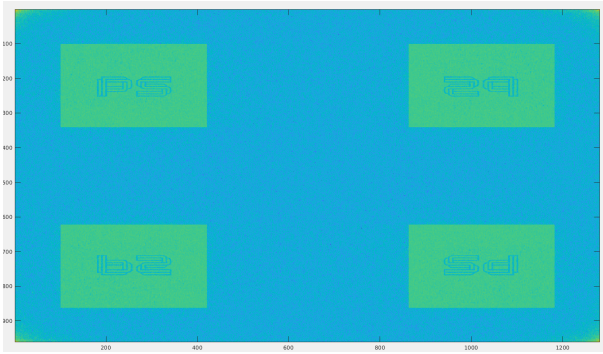
Pentru exercițiile 3 și 4 puteți folosi orice imagini, aveți grijă doar să redimensionați imaginile astfel încât imaginea carrier să fie de cel puțin de 2 ori mai mare decât imaginea secretă.

Urmați următorii pași:

1. calculați spectrul imaginii de transport.
2. selectați o zonă din spectrul imaginii de transport pe care o veți înlocui cu valorile imaginii secrete. Zona va avea colțul din stânga-sus în locația [top,left] și va avea dimensiunea [height,width] = size(secret_image).
3. trebuie să înlocuiți modulul (atenție: doar modulul nu și faza) spectrului din zona selectată cu valorile imaginii secrete. Intâi calculăm media valorilor spectrului din zona dorită, apoi înlocuim valorile din zonă cu valorile imaginii secrete scalate (de 2-10 ori mai mari decât media: ex. inmulțite cu media x 5).
4. vrem să trecem acest spectru înapoi în domeniul timp (de fapt este domeniu spațial, dar îl denumim domeniu timp doar ca să păstrăm terminologia de până acum: domeniu timp –> FFT –> domeniul spectral și invers).
5. pentru a avea sens trecerea din spectrul modificat înapoi în timp trebuie să avem grijă ca spectrul format să corespundă unui semnal real valid, adică spectrul trebuie să fie periodic pe cele 2 axe și trebuie să avem proprietatea |S(-k)| = |S(k)|. Pentru aceasta trebuie să avem un spectru care este

simetric față de jumătatea spectrului atât vertical cât și orizontal. Spectrul final trebuie sa arate asemănător cu spectrul de mai jos.

6. creați spectrul simetric și treceți în domeniul timp (spațial) cu 'ifft2'.
7. afisați imaginea stego - această imaginea ar trebui să fie insesizabil diferită față de imaginea originală.
8. extrageți imaginea secretă folosind funcția scrisă la exercițiul 2.



Exercițiul 4 -- Codificarea unei imagini secrete, gri în spectrul unui imagini color [Bonus]

Vrem să codificăm într-o imagine color o imagine gri sau mai multe. Pentru aceasta am putea codifica în fiecare dintre cele 3 canale RGB o imagine secretă. Pentru a avea totuși o imagine stego rezultată fară prea mult zgomot cauzat de encodarea imaginilor secrete vom avea altă abordare. Vom converti imaginea RGB în alt spațiu de culoare L*a*b, un spațiu de culoare cu 3 canale: L pentru luminanță care reprezintă majoritatea informației (echivalent cu imaginile gri cu care am lucrat până acum) și 2 canale de culoare a și b. Dacă modificăm doar canalele de culoare imaginea finală se modifică mult mai puțin sesizabil pentru un om, față de cazul în care modificăm canalele RGB sau Luminanța. De aceea vom encoda o singură imagine secretă (aceeași) în canalele a și b, lucrând cu ele individual ca și cum ar fi poze gri.

Folosind scheletul următor și funcțiile scrise precedent codificați o imagine gri într-o imagine color.

lab10_ex4.m

```
close all
clear all

height0  = 240;
width0   = 320;

height  = 1 * height0;
width   = 1 * width0;
%% citire
% citim imaginea de transport si o convertim din spatiul de culoare RGB in
% spatiul L*a*b*

img = imread('sky.jpg');
img = imresize(img, [4 * height0 + 1, 4 * width0 + 1]);
h = figure , image(img)
title('imagine originala')
print(h, '-dpng', 'steganography_imagine_originala.png');
img_lab = double(rgb2lab(img)) / 255;

h = figure , image(img_lab)
title('imagine originala in spatiul l*a*b')
%citim imaginea secreta si o convertim in gri
hid_img = imread('dog.jpeg');
hid_img = imresize(hid_img, [height,width]);
hid_img = double(rgb2gray(hid_img)) / 255;

h = figure, imshow(hid_img);
title('imagine secreta')
%% encodare
stego_a = encode_hidden_image(img_lab(:,:,2), hid_img);
stego_b = encode_hidden_image(img_lab(:,:,3), hid_img);
```

```
figure, imagesc(stego_a);
title('stego a')

stego = img_lab;
stego(:,:,2) = stego_a;
stego(:,:,3) = stego_b;

stego_rgb = lab2rgb(stego * 255);
savefile = 'steno_img.bmp'
% imwrite(stego_rgb,'stego_rgn.jpeg','jpeg','Mode','lossless','BitDepth',12);
imwrite(stego_rgb,savefile,'bmp');

%% decodare
stego_rgb_read = imread(savefile);
h = figure , image(stego_rgb_read);
title('imagine stego')

stego_jpeg = double(rgb2lab(stego_rgb_read))/255;
figure, imagesc(log(abs(fft2(stego_jpeg(:,:,2)))));

hid1 = decode_stegano_image(stego_jpeg(:,:,2));
hid2 = decode_stegano_image(stego_jpeg(:,:,3));
hid = (hid1 + hid2) / 2;

figure, imshow(hid);impixelinfo
title('imagine secretă recuperată')
```

Exercițiul 5 [Bonus] -- Verificare Robustețe

Vrem să verificăm cât de robustă este metoda noastră de codificare. Pentru aceasta obținem imaginea stego prin codificarea imaginii secrete în imaginea de transport, efectuăm diverse transformări pe aceasta și verificăm dacă imaginea secretă este încă prezentă.

Verificați următoarele transformări pe imaginea stego:

1. tăiați o parte din imaginea stego: faceți negru o parte din aceasta. Cu cât tăiați mai mult cu atât ar trebui ca imaginea secretă să se degradeze(fără să lipsească vreo parte).
2. redimensionați imaginea stego: redimensionați la 0.5 (imresize), apoi reveniți la rezoluția inițială.
3. salvați imaginea stego sub diverse formate de compresie (de ex: jpeg, png), citiți imaginea rezultă și recuperați imaginea secretă. Ar trebui să observați degradarea imaginii secrete.

- Responsabil: Andrei Nicolicioiu [mailto:andrei.nicolicioiu@gmail.com]

ps/labs/10.txt · Last modified: 2020/12/16 10:35 by ionut.gorgos