WAVELET-BASED IMAGE STEGANOGRAPHY

PENTTI SUNILA

We present a wavelet based steganography scheme for embedding hidden image data into a cover image. The performance of the method is measured by the PSNR and SSM metrics.

Contents

Date: June 7, 2021.

1. Introduction

Steganography refers to the practise of hiding signals into a innocuous piece of cover data. The reasons for this include private communications. The difference to cryptography is that the recipient or passer-on of the message will see a cover version that is intended to not cause suspicion, instead of an encrypted ciphertext. Steganography may be combined with cryptography for private communications.

Wavelet analysis is a data processing and analysis methodology that aims to overcome some of the limitations of Fourier analysis by providing both time and frequency views into the data at hand. In practise, a wavelet basis is formed and used to decompose the signal into a wavelet coefficient representation. This representation can be used with the same basis to reconstruct the original signal.

We provide an implementation of a wavelet-based steganography scheme for images, where both the hidden data and the cover data are taken as images. The aim is to hide one image into the other in a reconstructible form, while keeping the result indiscernible from the original cover.

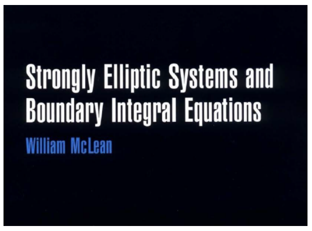
Image based steganography is a common target for steganography. Other methods include the so-called least significant bit steganography, where the least-significant bits of the image data are replaces with the data to be hidden.

The following sections present an implementation of a simple wavelet-based steganography scheme. A literate programming approach is followed, and the programming elements are mixed with the presentation and results of the method. A full program is also given in the appendix of the text.

2. Method

We consider the following scheme for image data. We utilize the following images for the method.

(A) Original cover image


Strongly Elliptic Systems and
Boundary Integral Equations
William McLean
(B) Image to be hidden

We begin by loading the images:

```
% Read in the input data and set script parameters
im = imread("coverimage1024.jpg");
im_stego = imread("stegoimage512.png");

im = im2gray(im);
im_stego = im2gray(im_stego);

% normalize hidden image
im_stego = double(im_stego)/255;
```

We require that the image to be hidden is such that it's largest dimension is at most half that of the smallest dimension of the cover image. We use 1024×1024 for the cover and 512×313 for the hidden image. The images are converted to grayscale to simplify computations.

We compute the two-level Haar wavelet transform of the cover image and extract coefficients:

```
[c,s]=wavedec2(X,2,'haar');

[H1,V1,D1] = detcoef2('all',c,s,1);

[H2,V2,D2] = detcoef2('all',c,s,2);
A2 = appcoef2(c,s,'haar',2);
```

We replace the first-order horizontal coefficients (size $(\frac{n}{2})^2$) by the data to be hidden.

```
H1=zeros(size(H1));
H1(1:size(im_stego,1), 1:size(im_stego,2)) = im_stego;
```

We compute the inverse transform to get back the stegoimage.

```
c2 = [A2(:)' , H2(:)' , V2(:)' , D2(:)' , H1(:)' , V1(:)' , D1(:)'];
im2 = waverec2(c2,s,'haar');
```

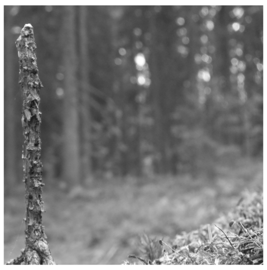
To extract the hidden image from the stegoimage, we may calculate the wavelet transform of the stegoimage:

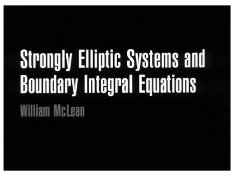
```
% Reconstruct the hidden image
[c,s] = wavedec2(stegoimage,2,'haar');
[H1,V1,D1] = detcoef2('all',c,s,1);
rec = H1(1:size(im_stego,1),1:size(im_stego,2));
```

We may now visualize the results:

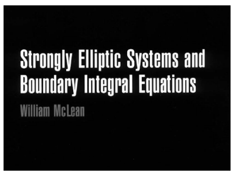
```
figure
imshow(uint8(im))
figure
imshow(uint8(im_stego))
figure
imshow(uint8(im2))
figure
imshow(rec)
```

(A) Original image


(B) Stegoimage


Strongly Elliptic Systems and
Boundary Integral Equations
William McLean

(C) Original hidden image


Strongly Elliptic Systems and
Boundary Integral Equations
William McLean

(D) Reconstructed hidden image

and compute similarity metrics:

```
psnrCover = psnr(im,uint8(stegoimage));
l2errorCover = norm(double(im(:)) - stegoimage(:)) /norm(double(im(:)));
structsimCover = ssim(double(im),stegoimage);
fprintf("Cover Image PSNR=%1.2f, Relative L2 Error %1.2f, SSIM %1.2f \n",...
psnrCover,l2errorCover,structsimCover);

psnrHidden = psnr(rec,im_stego);
```

```
l2errorHidden = norm(im_stego(:) - rec(:)) /norm(im_stego(:));
structsimHidden = ssim(im_stego,rec);
fprintf("Recovered Image PSNR=%1.2f, Relative L2 Error %1.2f, SSIM %1.2f\n",...
psnrHidden,l2errorHidden,structsimHidden);
```

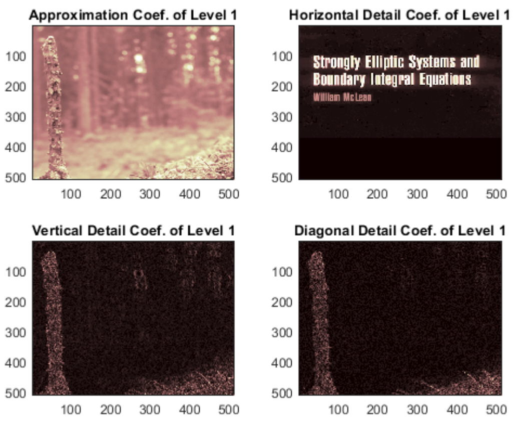
With the output:

```
Cover Image PSNR=38.92, Relative L2 Error 0.02, SSIM 0.92
Recovered Hidden Image PSNR=278.01, Relative L2 Error 0.00, SSIM 1.00
```

We have high metric values for the cover image, which is desirable. It is noted that the hidden image is reconstructed exactly, since it is not compressed or modified at all by the method.

We may visualize the method followingly:

```
% Visualize the method by checking the wavelet coefficients of the
% acquired stegoimage
[H1,V1,D1] = detcoef2('all',c,s,1);
A1 = appcoef2(c,s,'haar',1);

V1img = wcodemat(V1,255,'mat',1);
H1img = wcodemat(H1,255,'mat',1);
D1img = wcodemat(D1,255,'mat',1);
A1img = wcodemat(A1,255,'mat',1);

figure;
subplot(2,2,1)
imagesc(A1img)
colormap pink(255)
title('Approximation Coef. of Level 1')

subplot(2,2,2)
imagesc(H1img)
title('Horizontal Detail Coef. of Level 1')

subplot(2,2,3)
imagesc(V1img)
title('Vertical Detail Coef. of Level 1')

subplot(2,2,4)
imagesc(D1img)
title('Diagonal Detail Coef. of Level 1')
```



And note that even though the stegoimage is visually indistinguishable from the original cover, the hidden image is clearly visible from the wavelet decomposition of the stegoimage.

The key trick used for the method is the normalization of the hidden image into a floating point range between 0 and 1.0, instead of integer values between 0 and 255. This guaranteees that the wavelet coefficients themselves are small and close to zero compared to integer values:

```
>> max(max(D1))
ans =
44.5000

>> max(max(H1))
ans =
1.0000
```

It is noted that all the first level coefficients of the image may be put to zero without visually affecting the image:

```
H1 = zeros(size(H1));
V1 = zeros(size(H1));
D1 = zeros(size(H1));
c2 = [A2(:)' , H2(:)' , V2(:)' , D2(:)' , H1(:)' , V1(:)' , D1(:)'];
im2 = waverec2(c2,s,'haar');
figure
imshow(im)
figure
imshow(uint8(im2))
```

(A) Original image

(B) Image with zeroed coefficients

This fact, combined with the normalization for the hidden image allow the method to function successfully. The method described above is implemented by the Matlab script files waveletstego.m and waveletSteganography.m, distributed with this text and also given in the appendix. By modifying these files, we may compare the results when used with other choices of parameters.

For example, choosing the Daubechies 2 wavelet family instead of the Haar results in the following metrics.

```
>> waveletstego

wvlet =

'db2'

Cover Image PSNR=41.26, Relative L2 Error 0.02, SSIM 0.94
Recovered Hidden Image PSNR=33.40, Relative L2 Error 0.08, SSIM 0.98
```

We note that the hidden image is no longer recovered exactly. This is likely due to the fact that the previously used Haar wavelets are discrete by nature and thus function well for discrete data such as images. Choosing a continuous wavelet such as the Daubechies 2 means that rounding errors will play a part in the computations.

The resulting images are still visually indistinguishable and will not be printed here.

Using more scales for the method does not change the results. This is because the information is still hidden in the first-order coefficients. It is, however, conceivable that additional information could be hidden using the higher order coefficients, but since their size is halved at each iteration, the original hidden image must stay within the first-order coefficients. A modified multi-scale version of the file waveletSteganography.m is given along this text.

3. Conclusion, further research

Some obvious directions for future work include compression or encryption of the hidden image to allow for larger hidden images, and to better secure the data transmitted. Combining further (possibly wavelet-based) compression to the stegoimage could allow larger images to be considered as stegoimages. The steganography step could be combined with an encryption scheme such as public key encryption to allow only the intended recipient to be able to receive the stegoimage.

Support for color images would be straightforward to implement by considering the method for each color separately. Another way would be to consider the luminosity representation of the image and apply the method componentwise there. The method could also be expended from a Matlab script to a reusable command-line tool. The author is planning an expanded implementation of the presented method with the Python programming language.

4. Appendix: implementation

File: waveletstego.m

```
% Wavelet steganography scheme
close all;
clear all;

%% 1
% Read in the input data and set script parameters
im = imread("coverimage1024.jpg");
im_stego = imread("stegoimage512.png");

im = im2gray(im);
im_stego = im2gray(im_stego);

% normalize hidden image
im_stego = double(im_stego)/255;

%% 2
% Call the waveletSteganography() routine to process the image data
stegoimage = waveletSteganography(im,im_stego);
imshow(uint8(stegoimage));

%% 3
% Reconstruct the hidden image
[c,s] = wavedec2(stegoimage,2,'haar');
[H1,V1,D1] = detcoef2('all',c,s,1);
rec = H1(1:size(im_stego,1),1:size(im_stego,2));

figure;
imshow(rec)

%% 4
% Do comparisons
% Print and visualize results
psnrCover = psnr(im,uint8(stegoimage));
l2errorCover = norm(double(im(:)) - stegoimage(:)) /norm(double(im(:)));
structsimCover = ssim(double(im),stegoimage);
fprintf("Cover Image PSNR=%1.2f, Relative L2 Error %1.2f, SSIM %1.2f \n",...
psnrCover,l2errorCover,structsimCover);

psnrHidden = psnr(rec,im_stego);
l2errorHidden = norm(im_stego(:) - rec(:)) /norm(im_stego(:));
```

```
structsimHidden = ssim(im_stego,rec);
fprintf("Recovered Image PSNR=%1.2f, Relative L2 Error %1.2f, SSIM %1.2f\n",...
psnrHidden,l2errorHidden,structsimHidden);

%% 5
% Visualize the method by checking the wavelet coefficients of the
% acquired stegoimage
[H1,V1,D1] = detcoef2('all',c,s,1);
A1 = appcoef2(c,s,'haar',1);

V1img = wcodemat(V1,255,'mat',1);
H1img = wcodemat(H1,255,'mat',1);
D1img = wcodemat(D1,255,'mat',1);
A1img = wcodemat(A1,255,'mat',1);

figure;

subplot(2,2,1)
imagesc(A1img)
colormap pink(255)
title('Approximation Coef. of Level 1')

subplot(2,2,2)
imagesc(H1img)
title('Horizontal Detail Coef. of Level 1')

subplot(2,2,3)
imagesc(V1img)
title('Vertical Detail Coef. of Level 1')

subplot(2,2,4)
imagesc(D1img)
title('Diagonal Detail Coef. of Level 1')
```

File: waveletSteganography.m

```
function imSteg = waveletSteganography(im,im_stego)

X = im;
[c,s]=wavedec2(X,2,'haar');

[H1,V1,D1] = detcoef2('all',c,s,1);
[H2,V2,D2] = detcoef2('all',c,s,2);
A2 = appcoef2(c,s,'haar',2);

H1=zeros(size(H1));
H1(1:size(im_stego,1), 1:size(im_stego,2)) = im_stego;

c2 = [A2(:)' , H2(:)' , V2(:)' , D2(:)' , H1(:)' , V1(:)' , D1(:)'];
im2 = waverec2(c2,s,'haar');
imSteg = im2;

end
```

And a modified version for a 4-scale analysis.

```
function imSteg = waveletSteganography(im,im_stego,wvlet)

X = im;
[c,s]=wavedec2(X,4,wvlet);

[H1,V1,D1] = detcoef2('all',c,s,1);
[H2,V2,D2] = detcoef2('all',c,s,2);
[H3,V3,D3] = detcoef2('all',c,s,3);
[H4,V4,D4] = detcoef2('all',c,s,4);
A4 = appcoef2(c,s,wvlet,4);

H1=zeros(size(H1));
H1(1:size(im_stego,1), 1:size(im_stego,2)) = im_stego;

c2 = [A4(:)' , H4(:)' , V4(:)' , D4(:)' , H3(:)' , V3(:)' , D3(:)',...
H2(:)' , V2(:)' , D2(:)' , H1(:)' , V1(:)' , D1(:)'];

%c2 = [A2(:)' , H2(:)' , V2(:)' , D2(:)' , H1(:)' , V1(:)' , D1(:)'];
im2 = waverec2(c2,s,wvlet);
imSteg = im2;

end
```