UNIVERSIDADE DE SÃO PAULO

INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
IMAGE PROCESSING - SCC0251
PROF. DR. MOACIR PONTI

Project 2: Steganography: Hiding information on image frequencies through DCT

Jaqueline Alvarenga Silveira

2017

1. Objective

The main objective of this study is to develop a program that allows a user to hide and extract information in JPEG images through the discrete cosine transform. The idea is to try to embed the information in the image preferably without any visual degradation and with some resistance to compression.

2. Input Images

In this study, the tests will be performed with JPEG images, because the JPEG format offers reasonable levels of image quality and generates small files compared to other formats. This makes it easier to store and distribute, as it uses image compression. In this format, the higher the level of compression, the smaller the file size, however the image quality will be worse. Therefore, each time an image is saved, it loses quality. An image in the Bitmap format will also be used. The images of this study are taken from [1] and [3] and can be observed in Figure 1a and 1b.

(a) (b)

Figure 1. (a) is the JPEP image used in the tests, and (b) is the Bitmap image.

3. Description of steps

3.1. Pre-Processing

Initially, the user must enter the name of the image and the text that s/he wants to encode. However, I will already leave a demo ready in the github, in order to facilitate this process. The programming language that will be used is Python 2.7 and I will use some OPenCV 2.7 library methods. Thus, the image is read by the cv2.imread method (fileName, 1), and then the text is read. In addition, the dct and idct methods of the scipy.fftpack library are used to calculate the converted cosine and its inverse, respectively.

3.2. Discrete Cosine Transformation (DCT)

DCT method separates the image into parts of differing importance. It transforms a signal or image from the spatial domain to the frequency domain. It can separate the image into high, middle and low frequency components [?]. The general equation for 2D (N by M image) DCT is defined by the following equation:

$$c(u,v) = a(u)a(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} I(x,y)\cos\frac{(2x+1)u\phi}{2N}\cos\frac{(2y+1)v\phi}{2N} \tag{1}$$

where,

$$a(u), v(u) = \begin{cases} \dfrac{1}{\sqrt{n}}, & \text{if } u, v = 0 \\ \dfrac{2}{\sqrt{n}}, & \text{if } u, v = 1, 2, ..., n-1 \end{cases} \tag{2}$$

The Inverse Discrete Cosine Transform (IDCT) is the inverse of DCT which can be obtained by:

$$I(u,v) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} a(u)a(v)I(r,c)\cos\frac{(2x+1)u\phi}{2N}\cos\frac{(2y+1)v\phi}{2N} \tag{3}$$

where u, v varies from 0 to $n-1$.

After the pre-processing, the text should then be hidden in the image. Thus, in this study, the image decomposition concept is used to improve image quality. In the proposed method, the images are decomposed into different DCT frequencies in which the data is hidden.

The biggest issue of hiding the data in the coefficients is that by transforming the coefficients back to the spatial domain of the image. Since the pixels of the image are stored in the full form, rounding errors will be added into the image's spatial domain. Therefore, during the extraction process, DCT coefficients cannot be reversed once they have been modified.

To solve this problem, this paper adapts the technique proposed by [5]. In their study, twenty-two values were randomly selected from each 8×8 block of the DCT coefficients, and of these twenty-two values, two of them were also randomly chosen. Then, the values are compared, for example, if you want to hide bit 1, then $DCT(u_1, v_1) > DCT(u_2, v_2)$. If the condition is satisfied, the values are retained. If not, switch the two coefficients' positions. If you want to hide bit 0, $DCT(u_2, v_2) > DCT(u_1, v_1)$. If the criterion is satisfied the values are retained, otherwise switch their positions.

The proposal in this study is very similar, however, it is possible to choose more than one fixed value to change in each block and after the switch, to subtract / add a value of $DCT(u_1, v_1)$ and $DCT(u_2, v_2)$, depending on the bit to be coded. This step ensures that the $DCT(u_1, v_1)$ and $DCT(u_2, v_2)$ values will not be modified after the conversion. The 1 that consists of hiding the information and the algorithm 2 that extracts the hidden information is described below. In the algorithm below, only one bit per block is hidden:

3.3. Method validation

In order to verify the compression rates that the proposed method supports and also to determine the number of bits hidden per DCT block for optimal quantity, calculate the

Algorithm 1: Algorithm that hides the information in the DCT frequency values [4]

```
input  : A JPEG image and the text that will be hidden in the image
output: An image with the hidden information
1  Function DCT-Swap $(image, text)$ Check if the image is large enough to
     conceal the information;
2  The image is divided in 8 $\times$ 8 blocks;
3  $k \leftarrow 30$;
4  4 Convert the text in a linear vector named '0' and '1' bit;
5  for $i \leftarrow 0$ to $bit.lenght$ do
6      Calculate the DCT for a block;
7      if $bit[i] \leftarrow 1$ and $DCT(u_1, v_1) < DCT(u_2, v_2)$ then
8          swap(DCT (u$_1, v_1), DCT(u_2, v_2))$
9      if $bit[i] \leftarrow 0$ and $DCT(u_1, v_1) > DCT(u_2, v_2)$ then
10         swap(DCT (u$_1, v_1), DCT(u_2, v_2))$
11     if $DCT(u_1, v_1) > DCT(u_2, v_2)$ then
12         $DCT(u_1, v_1) \leftarrow DCT(u_1, v_1) + k$;
13         $DCT(u_2, v_2) \leftarrow DCT(u_2, v_2) - k$;
14     else
15         $DCT(u_1, v_1) \leftarrow DCT(u_1, v_1) - k$;
16         $DCT(u_2, v_2) \leftarrow DCT(u_2, v_2) + k$;
17 end
18 Merge the blocks again;
19 Apply the DCT inverse to return to the image's spatial domain;
```

Algorithm 2: Algorithm that extracts information of the DCT frequency values

```
input  : An image with the hidden information
output: The hidden information
1  Function DCT-Swap $(image, text)$
2  The image is divided in 8 $\times$ 8-sized blocks;
3  for $i \leftarrow 0$ to $numBlocks$ do
4      Calcule a DCT para um bloco;
5      if $DCT(u_1, v_1) > DCT(u_2, v_2)$ then
6          $bits \leftarrow bits+$ '1';
7      else
8          $bits \leftarrow bits+$ '0';
9      $j \leftarrow j + 1$ if $j = 8$ then
10         $str \leftarrow str + chr(int(bits, 2))$ ;
11         $bits \leftarrow$ '' ;
12         $j \leftarrow 0$ ;
13 end
14 Assemble the string by deleting nonprintable characters;
```

RMSE in relation to the input image (without hidden information) and the output images (with the information hidden). A table showing a range of the number of bits / block versus compression will be presented. Thus, the calculation for each value in the table will be:

$$RMSE = \sqrt{\frac{(f - \hat{f})^2}{N * M}} \tag{4}$$

in which f is an image without hidden information, $\hat{f}$ is an image with hidden information and M e N is the width and height of the image, respectively. The RMSE is a measurement of distance that compares the expected data with the measured data.

3.4. Steganalysis

Steganalysis is the art and science of detecting embedded message based on visual inspection, statistical analysis or other methods. In this work, it is used statistical analysis.

3.4.1. P value

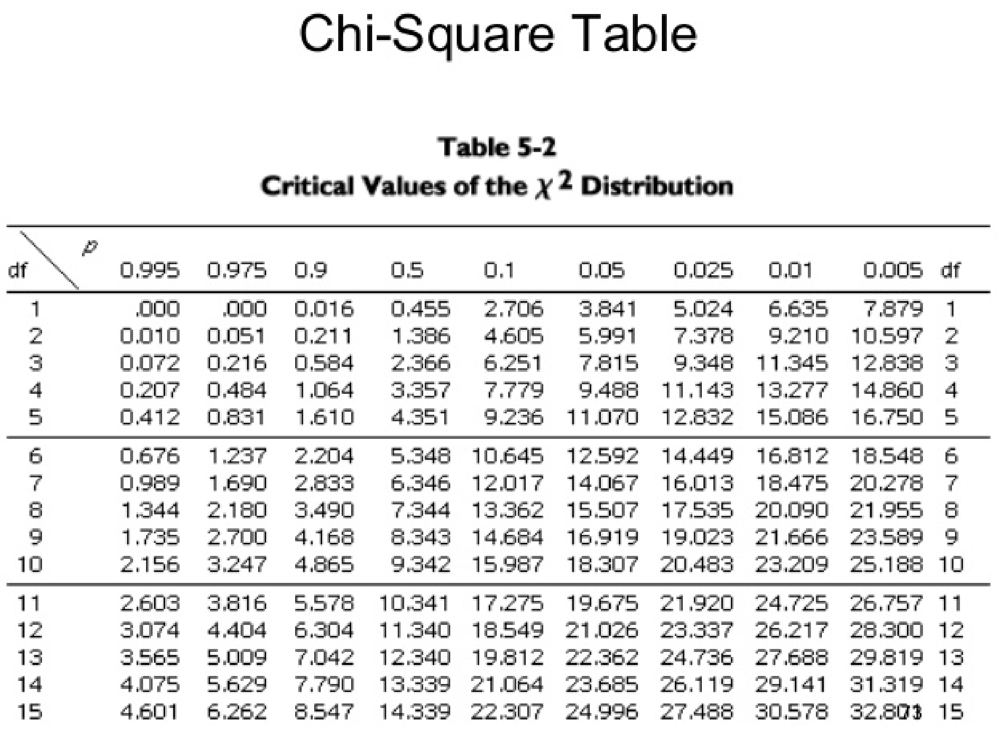
P value is a statistical measure that helps scientists determine whether or not their hypotheses are correct. P values are used to determine whether the results of their experiment are within the normal range of values for the events being observed. Usually, if the P value of a data set is below a certain pre-determined amount scientists will rule out the hypothesis that the variables of their experiment had no meaningful effect on the results. In this work, p values are usually found on a reference Table 2 by first calculating a Chi-square value. The algorithm 3 that consists of Chi-square attack is described below:

Algorithm 3: Chi-squared test

```
input  : A JPEG image without the hidden information
output: An image with the hidden information
1  Function chiSquare ($img1, img2$)
2  $hist1 \leftarrow$ histogram(img1) ;
3  $hist2 \leftarrow$ histogram(img2);
4  $chisqr \leftarrow 0.0$;
5  for $i \leftarrow 0$ to $256$ do
6      if $hist2[i]! = 0$ then
7      $diff \leftarrow (hist1[i] - hist2[i])^2$;
8      $chisqr+ = abs(diff)/hist2[i]$
9  end
10 return chisqr
```

4. Results

The following results were obtained using a JPEG image, hiding only one bit of text per block and with 100% quality and no loss. The image initially used can be seen in Figure 3a and the image generated by the program can be seen in Figure 3b.



Chi-Square Table

Table 5-2

Critical Values of the χ^2 Distribution

df \ p	0.995	0.975	0.9	0.5	0.1	0.05	0.025	0.01	0.005	df
1	.000	.000	0.016	0.455	2.706	3.841	5.024	6.635	7.879	1
2	0.010	0.051	0.211	1.386	4.605	5.991	7.378	9.210	10.597	2
3	0.072	0.216	0.584	2.366	6.251	7.815	9.348	11.345	12.838	3
4	0.207	0.484	1.064	3.357	7.779	9.488	11.143	13.277	14.860	4
5	0.412	0.831	1.610	4.351	9.236	11.070	12.832	15.086	16.750	5
6	0.676	1.237	2.204	5.348	10.645	12.592	14.449	16.812	18.548	6
7	0.989	1.690	2.833	6.346	12.017	14.067	16.013	18.475	20.278	7
8	1.344	2.180	3.490	7.344	13.362	15.507	17.535	20.090	21.955	8
9	1.735	2.700	4.168	8.343	14.684	16.919	19.023	21.666	23.589	9
10	2.156	3.247	4.865	9.342	15.987	18.307	20.483	23.209	25.188	10
11	2.603	3.816	5.578	10.341	17.275	19.675	21.920	24.725	26.757	11
12	3.074	4.404	6.304	11.340	18.549	21.026	23.337	26.217	28.300	12
13	3.565	5.009	7.042	12.340	19.812	22.362	24.736	27.688	29.819	13
14	4.075	5.629	7.790	13.339	21.064	23.685	26.119	29.141	31.319	14
15	4.601	6.262	8.547	14.339	22.307	24.996	27.488	30.578	32.801	15

Figure 2. Reference table by calculating a Chi-square value. [2]

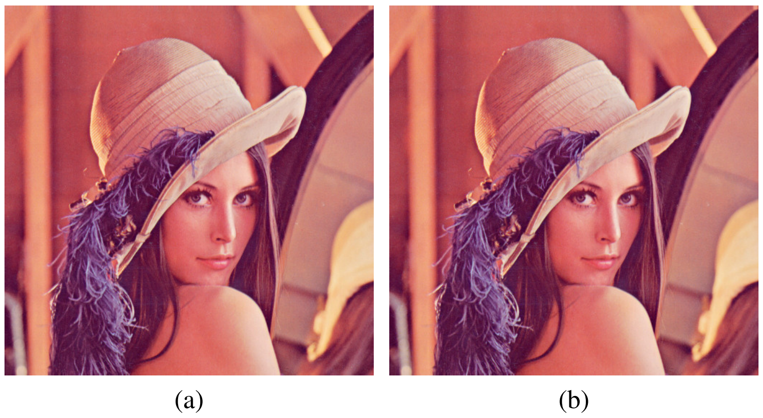
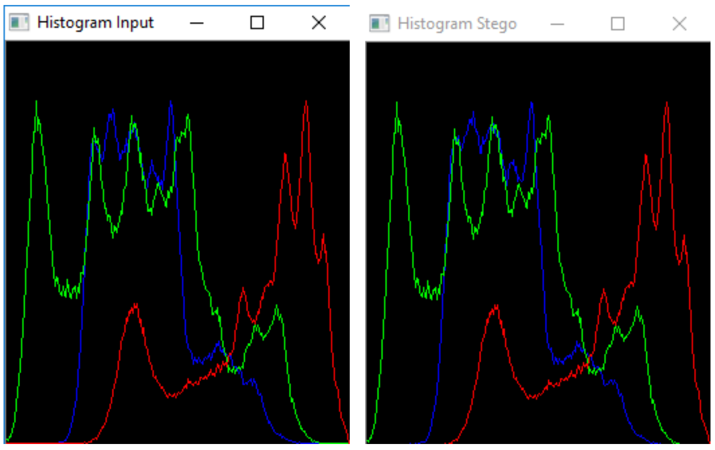


Figure 3. In (a), an image that does not have hidden information is shown. An image that has hidden information is shown in (b).

In Figure 4a and 4b it is possible to examine the histograms of the input and output images respectively, generated by the program developed in this paper. A slight difference can be noted between the histograms of the images.

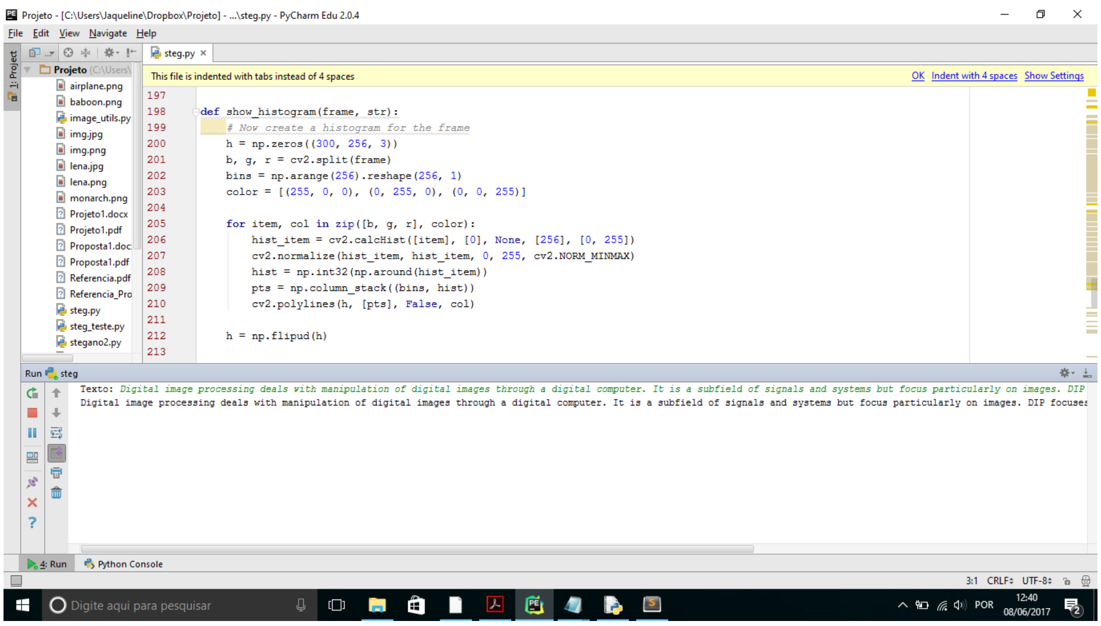
In Figure 5 , an image with the message extracted by the program is shown. It is important to note that the message was extracted with 100% similarity when the image is stored at 100% quality.

In order to verify how much compression versus the number of bits / blocks that the method of this paper upholds, a test with the Lena.jpg image was carried out. Thus,



(a) (b)

Figure 4. In (a) the histogram of the image without hidden information can be seen. Whereas in (b), the histogram of the input image that has hidden information is presented.


Projeto - [C:\Users\Jaqueline\Dropbox\Projeto] - ...\steg.py - PyCharm Edu 2.0.4
File Edit View Navigate Help
steg.py
This file is indented with tabs instead of 4 spaces
OK Indent with 4 spaces Show Settings
Run steg
Run
Python Console
Digite aqui para pesquisar

```python
def show_histogram(frame, str):
	# Now create a histogram for the frame
	h = np.zeros((300, 256, 3))
	b, g, r = cv2.split(frame)
	bins = np.arange(256).reshape(256, 1)
	color = [(255, 0, 0), (0, 255, 0), (0, 0, 255)]

	for item, col in zip([b, g, r], color):
		hist_item = cv2.calcHist([item], [0], None, [256], [0, 255])
		cv2.normalize(hist_item, hist_item, 0, 255, cv2.NORM_MINMAX)
		hist = np.int32(np.around(hist_item))
		pts = np.column_stack((bins, hist))
		cv2.polylines(h, [pts], False, col)

	h = np.flipud(h)
```

```
Texto: Digital image processing deals with manipulation of digital images through a digital computer. It is a subfield of signals and systems but focus particularly on images. DIP
Digital image processing deals with manipulation of digital images through a digital computer. It is a subfield of signals and systems but focus particularly on images. DIP focuses
```

Figure 5. Message extracted from the image.

the number of bits / block from one to four varies, just as the quality of the image varies (without compression), at 100%, 95%, 90% and 85%. In Table 1, the number of bits / block in relation to the compression quality in which the image with the information was saved can be observed. Note that the best configuration is when storing 3 bits / block, because as the RMSE is smaller, it is possible to recover 100% of the information as can be seen in Table 2. It is also possible to use more bits per block, increasing the amount of words that can be stored in the image. If the user needs more compression, according to Tables 1 and 2, it is best to choose up to 2 bits / block with a maximum of 90% compression.

Additionally, in order to verify how much steganalisys attack that the method of

Table 1. The RMSE calculation of the image after hiding the information and saving it to the computer. In the first row of the table, the relation to the number of bits / block varies. Whereas in the first column of the table, it is the relation to the compression quality in which the image was saved that varies.

	1	**2**	**3**	**4**
w/o comp.	1.38	1.57	1.24	1.12
100%	2.78	2.82	2.72	2.68
95%	3.27	3.27	3.18	3.13
90%	3.44	3.52	3.44	3.41
85%	3.87	3.92	3.91	3.89

Table 2. Calculation of the information retrieval rate. In the first row of the table, the rate varies in relation to the number of bits / block. In the first column of the table, the rate varies in relation to the compression quality at which the image has been saved.

	1	**2**	**3**	**4**
w/o comp.	100	100	100	75
100%	95	95	78	55
95%	95	96	74	55
90%	81	94	74	55
85%	61	0	0	0

this paper upholds, a chi-square test with the Lena.jpg image was carried out. Since there is not value greater than 3.4 value in Table 3, the method this work is very robust against steganalisys attack.

Looking on Table 3, we can to observe that the max chi-square was 0.05. So, let's use the chi square distribution table2 above to find an approximate p value. Since we know our experiment has only 1 degree of freedom (degree of freedom = n - 1, n is the number of images), we'll start in the highest row. We'll go from left to right along this row until we find a value higher than 0.05 - our chi square value. The first one we encounter is 2.706. Looking to the top of this column, we see that the corresponding p value is 0.10. This means that our p value is between 0.10 and 0.90 (the next-biggest p value on the table).

Since we have found an approximate p value for our experiment, we can decide whether or not to reject the hypothesis of our experiment (as a reminder, this is the hypothesis that the experimental variables we manipulated did not affect the results we observed). If our p value is lower than our significance value, we've shown that our experimental results would be highly unlikely to occur if there was no real connection between the variables we manipulated and the effect we observed. If our p value is higher than our significance value, we can't confidently make that claim. Our p value is smaller than 0.10, we can reject our hypothesis in all cases.

Another test with Bitmap images was also carried out. However, for this type of image it was not possible to use compression since the message is irretrievable. Yet without compression, it is possible to recover 100% of the information. In Figure 6a, the original image is shown and in Figure 6b the image with the hidden information is

Table 3. The Chi-square calculation of the image after hiding the information and saving it to the computer. In the first row of the table, the relation to the number of bits / block varies. Whereas in the first column of the table, it is the relation to the compression quality in which the image was saved that varies.

	1	**2**	**3**	**4**
w/o comp.	0.003	0.003	0.002	0.001
100%	0.036	0.031	0.006	0.005
95%	0.056	0.044	0.012	0.014
90%	0.058	0.052	0.015	0.006
85%	0.037	0.055	0.001	0.009

shown. It is important to note that all images to be used must be of good quality (without undergoing many compressions), as tests have been performed with poor quality images and the information could not be retrieved.

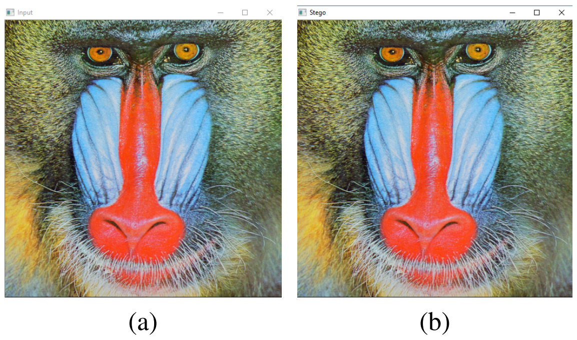


Figure 6. The original image without hidden information can be observed in (a). In (b), the original image with hidden information is shown.

References

[1] `https://www.cosy.sbg.ac.at/~pmeerw/Watermarking/lena.html`, Access: 05-06-2017.

[2] `http://www.wikilectures.eu/index.php/Hypothesis_Testing`, Access:07-06-2017.

[3] *Wallpapers*, `http://www.wallpapersonview.com/resolution/1920x1200-page-1.html`, Access: 05-06-2017.

[4] Lin Chia-Chen and Shiu Pei-Feng, *High capacity data hiding scheme for dct-based images*, Journal of Information Hiding and Multimedia Signal Processing **1** (2010), no. 3, 2073–4212.

[5] Alisha Parnami, Ankit Gupta, and Gagan Parnami, *Article: A robust dct based digital image watermarking using random mid-band coefficient exchange scheme for gray scale images*, International Journal of Computer Applications **100** (2014), no. 8, 6–10, Full text available.