On the use of the discrete Pascal transform in hiding data in images

Eleni E. Varsaki, Vassilis E. Fotopoulos and Athanassios N. Skodras
Digital Systems & Media Computing Laboratory, School of Science and Technology
Hellenic Open University, Patras, Greece
{e.varsaki, vfotop1, skodras}@eap.gr

ABSTRACT

The Discrete Pascal Transform (DPT) has been proved remarkably useful for edge detection, filter design, discrete-time signal interpolation and data hiding. In the present work a new blind fragile data hiding technique for secretly embedding messages into color images, is proposed. The embedding procedure is based on dividing each color image component into even-sized blocks. Information embedding is determined by monitoring the lower-right corner of the DPT coefficient matrix. This particular coefficient suffers the highest change for small pixel modifications. The embedding affects the coefficient's sign. In case that the sign is not the desired one, i.e. negative for a message bit value of '0' and positive for a message bit value of '1', it is changed by repeatedly adding to the block or subtracting from the block the identity matrix. This process is based on the DPT properties and on the sensitivity of the lower-right coefficient in even the smallest pixel changes. The embedding algorithm takes care of the underflows or overflows that may occur during the consecutive additions or subtractions. The method is evaluated in terms of capacity and image distortion. Experiments are conducted using different images and block sizes namely 2x2 / 4x4 / 8x8 / 16x16, and the overall performance of the scheme is quantified. Block size greatly affects capacity and stego-image quality. Comparisons with existing methods prove the superiority of the proposed method.

Keywords: Discrete Pascal Transform, Data Hiding, Steganography

1. INTRODUCTION

During the last decade, the number of works on digital media watermarking, has been exponentially increased[1]. The driving force behind this, is the urgent need for copyright protection since the amounts involved in media industry are really huge. Embedding copyright information and other kind of metadata into the medium, in an invisible/inaudible way has become a reality, even in affordable consumer electronics devices[2]. Data hiding is closely related to watermarking but in principle it is the message to be conveyed, which is of prime importance and not the carrier itself (as it is in watermarking). Such applications include secret communications between agencies, industrial espionage, e-healthcare etc[3].

Robustness[4] is a key characteristic of watermarking for copyright protection but not for other data hiding methods for which capacity is usually more important. If the embedded message cannot be retrieved after simple image processing operations, then the scheme is called *fragile*[5,6,7]. Such methods are commonly used in tamper proofing and are mainly operating in the spatial domain. LSB[8] and histogram[9] manipulation methods are example categories of such techniques. A small level of robustness may be required in the case that the image may sustain some light, non malicious processing (e.g. moderate lossy compression).

In this communication the use of Discrete Pascal Transform (DPT) is proposed in order to achieve data hiding. Aburdene and Goodman[10] are the first who referred to watermarking as an application area of DPT. Kostopoulos et al[11] have suggested an interesting watermarking scheme using Pascal triangles. Wong et al[12] proposed an effective watermarking method in Pascal transformed domain used for medical image authentication by altering the high frequency components according to a code book. In this paper a new data hiding technique is proposed, which uses DPT to invisibly embed secret data into color images. The proposed technique is blind, fragile, and can be used for secret communications, tamper proofing and authentication[13].

Two properties of DPT are used in order to form the proposed data hiding method. Firstly, the DPT is highly sensitive to noise and even the smallest noise can seriously distort the values of the transform coefficients of the signal. If a single

Optics, Photonics, and Digital Technologies for Multimedia Applications, edited by Peter Schelkens, Touradj Ebrahimi, Gabriel Cristóbal, Frédéric Truchetet, Pasi Saarikko, Proc. of SPIE Vol. 7723, 77230L · © 2010 SPIE · CCC code: 0277-786X/10/$18 · doi: 10.1117/12.854220

change of signal *y[n], n=0,1,2,...,N-1* occurs at instant *m*, then all DPT coefficients *Y[k], k=0,1,2,...,N-1* change for $m \le k \le N-1$. Secondly, coefficients in the Pascal transform represent weighted differences of neighbouring data values. That is any sharp changes in the pixel values produce large DPT coefficients. Furthermore the forward and inverse DPT transforms are computationally equivalent[10] and can be efficiently computed[14].

The embedding procedure uses the lower-right corner of the DPT coefficient matrix of even sized blocks of pixels. This is because this particular coefficient rapidly changes for even small pixel modifications. The coefficient's sign can determine whether the bit embedded into the block is *'0'* or *'1'*. If the coefficient's sign doesn't match the message bit, then the identity matrix is added or subtracted in order to change it. Overflows or underflows are taken into account during the embedding procedure.

The rest of the paper is organized as follows. The 2D Discrete Pascal Transform is presented in section 2. Embedding and extracting algorithms are described in sections 3 and 4, respectively. In section 5 experimental results are given and finally in section 6 conclusions are drawn.

2. DISCRETE PASCAL TRANSFORM

The discrete Pascal transform of a size *MxM* 2D signal **x** is defined as

$$\mathbf{X} = \mathbf{P}_M \ \mathbf{x} \ \mathbf{P}_M^T , \tag{1}$$

and $\mathbf{P}^T$ is the transpose of **P**. The Pascal transform matrix elements are equal to

$$p_{i,j} = \begin{cases} (-1)^j a_{ij}, \textit{for} \quad i \ge j \\ 0, \quad \textit{otherwise} \end{cases}, \tag{2}$$

where *i,j=0,1,2,...,M-1*, and a_{ij} is equal to the binomial coefficient, i.e. $a_{ij} = \binom{i}{j} = \frac{i!}{j!(i-j)!}$. The discrete Pascal transform matrices for *M=2, M=3* and *M=4* are given in (3). Zero elements are denoted by a dot for clarity.

$$\mathbf{P}_2 = \begin{bmatrix} 1 & . \\ 1 & -1 \end{bmatrix}, \ \mathbf{P}_3 = \begin{bmatrix} 1 & . & . \\ 1 & -1 & . \\ 1 & -2 & 1 \end{bmatrix}, \mathbf{P}_4 = \begin{bmatrix} 1 & . & . & . \\ 1 & -1 & . & . \\ 1 & -2 & 1 & . \\ 1 & -3 & 3 & -1 \end{bmatrix}. \tag{3}$$

Pascal transform matrix is *involuntary*, i.e. it is its own inverse. Thus the inverse DPT is calculated by multiplying again with the Pascal transform matrix as in (4):

$$\mathbf{x} = \mathbf{P}_M \ \mathbf{X} \ \mathbf{P}_M^T . \tag{4}$$

The 2D computation is equivalent to performing a one dimensional transform on the rows of the matrix **x**, followed by a one dimensional transform on the columns of the resulting coefficient matrix.

In Aburdene[10] et al it is shown that any sharp change in the pixel values of an image produces large coefficients in the transform matrix. For a constant-value image, all DPT coefficients are equal to 0 except for the *X(1,1)* coefficient, which equals to the pixel value. When a variation occurs, the DPT coefficients change considerably after the variation location. In the following we study how DPT coefficients are affected when the *identity matrix* or *exchange matrix* (i.e. row– or column–reversed version of the identity matrix) is added to 2D data.

For simplicity let us denote matrix **x** by $\mathbf{x}_M$. Then the *2x2* block data matrix $\mathbf{x}_2$ is equal to:

$$\mathbf{x}_2 = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}, \tag{5}$$

and its DPT $\mathbf{X}_2$ calculated according to (1) equals to:

$$\mathbf{X}_2 = \begin{bmatrix} x_{11} & x_{11}-x_{21} \\ x_{11}-x_{12} & x_{11}-x_{21}-x_{12}+x_{22} \end{bmatrix}. \tag{6}$$

Let $\mathbf{I}_2$ be the *2x2* identity matrix. If $\mathbf{x}_2'$ is the *2x2* result of the addition of $\mathbf{x}_2$ and $\mathbf{I}_2$, then the DPT of $\mathbf{x}_2'$ is denoted as $\mathbf{X}_2'$ and is given in (8).

$$\mathbf{x}_2' = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} x_{11}+1 & x_{12} \\ x_{21} & x_{22}+1 \end{bmatrix}. \tag{7}$$

$$\mathbf{X}_2' = \begin{bmatrix} x_{11}+1 & (x_{11}+1)-x_{21} \\ (x_{11}+1)-x_{12} & (x_{11}+1)-x_{21}-x_{12}+(x_{22}+1) \end{bmatrix} \Rightarrow$$

$$\mathbf{X}_2' = \begin{bmatrix} x_{11} & x_{11}-x_{21} \\ x_{11}-x_{12} & x_{11}-x_{21}-x_{12}+x_{22} \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \Rightarrow$$

$$\mathbf{X}_2' = \mathbf{X}_2 + \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}. \tag{8}$$

We observe that the addition of the identity matrix increases all DPT coefficients by one, except for coefficient *$X_2'(2,2)$* which is increased by 2. In fact the *symmetric Pascal matrix* $\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$ that is added to $\mathbf{X}_2$, is the DPT of the identity matrix.

Let $\mathbf{x}_3$ be a *3x3* data block (9) and $\mathbf{x}_3'$ the result of $\mathbf{x}_3 + \mathbf{I}_3$. The DPT of $\mathbf{x}_3'$ results in $\mathbf{X}_3'$ (10), where $\mathbf{X}_3$ is in fact the DPT of $\mathbf{x}_3$ and the *3x3* symmetric Pascal matrix is the DPT of the *3x3* identity matrix $\mathbf{I}_3$.

$$\mathbf{x}_3 = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \tag{9}$$

$$\mathbf{X}_3' = \mathbf{X}_3 + \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 6 \end{bmatrix}. \tag{10}$$

Similarly, for the *4x4* case it is proved that the DPT of the *4x4* 2D data block $\mathbf{x}_4$, augmented by the $\mathbf{I}_4$ matrix, equals to

$$\mathbf{X}_4' = \mathbf{X}_4 + \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 3 & 6 & 10 \\ 1 & 4 & 10 & 20 \end{bmatrix}, \tag{11}$$

where $\mathbf{X}_4$ is the DPT of the original *4x4* 2D data block $\mathbf{x}_4$. Thus we see that the DPT of any odd or even sized data block increased by one along the diagonal, i.e. adding the identity matrix, results to the addition of the individual DPT's (eq. 12).

$$DPT(\mathbf{x}_M + \mathbf{I}_M) = DPT(\mathbf{x}_M) + DPT(\mathbf{I}_M),\ M \in \mathbf{Z} \tag{12}$$

where $DPT(\mathbf{I}_M)$ equals to the symmetric Pascal matrix of size *MxM*. That is to say that the DPT of a *MxM* data block increased by the *MxM* identity matrix is equal to the sum of the DPT of the block and the *MxM* symmetric Pascal matrix. This in fact is a result of the linearity property of the DPT.

Now let $\mathbf{x}_2''$ be the *2x2* block resulted from the addition of the *2x2* $\mathbf{x}_2$ data and $\mathbf{J}_2$, where $\mathbf{J}_2$ is the *2x2 anti-diagonal identity* or *exchange* matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. $\mathbf{X}_2''$ is the DPT of $\mathbf{x}_2''$ and equals to:

$$\mathbf{X}_2'' = \begin{bmatrix} x_{11} & x_{11}-(x_{21}+1) \\ x_{11}-(x_{12}+1) & x_{11}-(x_{21}+1)-(x_{12}+1)+x_{22} \end{bmatrix} \Rightarrow$$

$$\mathbf{X}_2'' = \mathbf{X}_2 + \begin{bmatrix} 0 & -1 \\ -1 & -2 \end{bmatrix}. \tag{13}$$

It is obvious that the $X_2''(1,1)$ coefficient remains unchanged. The rest of the coefficients are decreased by one except for $X_2(2,2)$, which is decreased by 2.

Similarly, the DPT of a *4x4* data block in which the *4x4* anti-diagonal identity matrix $\mathbf{J}_4$ is added, equals to:

$$\mathbf{X}_4'' = \mathbf{X}_4 + \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & -4 \\ 0 & -1 & -4 & -10 \\ -1 & -4 & -10 & -20 \end{bmatrix} \tag{14}$$

In general

$$DPT(\mathbf{x}_M + \mathbf{J}_M) = DPT(\mathbf{x}_M) + DPT(\mathbf{J}_M)\ , for\ M=2k,\ k \in \mathbf{Z} \tag{15}$$

$DPT(\mathbf{J}_M)$ is an even sized triangular matrix, whose elements monotonically decrease towards the *(M,M)* element.

When the odd sized anti-diagonal identity matrix is added to an odd sized matrix the DPT coefficients are increased, as for the 3x3 block below

$$\mathbf{x}_3'' = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & x_{13}+1 \\ x_{21} & x_{22}+1 & x_{23} \\ x_{31}+1 & x_{32} & x_{33} \end{bmatrix} \tag{16}$$

$$\mathbf{X}_3'' = \mathbf{X}_3 + \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 3 \\ 1 & 3 & 6 \end{bmatrix}, \tag{17}$$

where $\mathbf{X}_3''$ is the DPT of the *3x3* data block $\mathbf{x}_3''$ and $\mathbf{X}_3$ is the DPT of the *3x3* data block prior to addition of the exchange matrix. In general

$$DPT(\mathbf{x}_M + \mathbf{J}_M) = DPT(\mathbf{x}_M) + DPT(\mathbf{J}_M)\ , for\ M=2k+1,\ k \in \mathbf{Z} \tag{18}$$

$DPT(\mathbf{J}_M)$ is a triangular odd sized matrix whose elements increase towards the *(M,M)* element.

It has been experimentally observed that the $X_M(M,M)$ coefficient of natural photos has a zero mean narrow distribution, almost symmetric with a considerable tail. The coefficients are positive, negative or zero. It is possible to alter the sign of the coefficients by just adding the identity or the anti-diagonal identity *MxM* matrix to the *MxM* pixel block. This property of the DPT can be exploited in order to invisibly embed data into a block of pixels. The proposed embedding approach is actually based on this particular property.

3. THE PROPOSED EMBEDDING ALGORITHM

In DPT, the alteration of even a pixel value affects the subset of coefficients, belonging in the rectangular region defined by the particular pixel location and the lower-right corner of the coefficient matrix[10]. The change is very severe. This property is used in order to embed data into an image, because even small pixel changes can remarkably influence Pascal

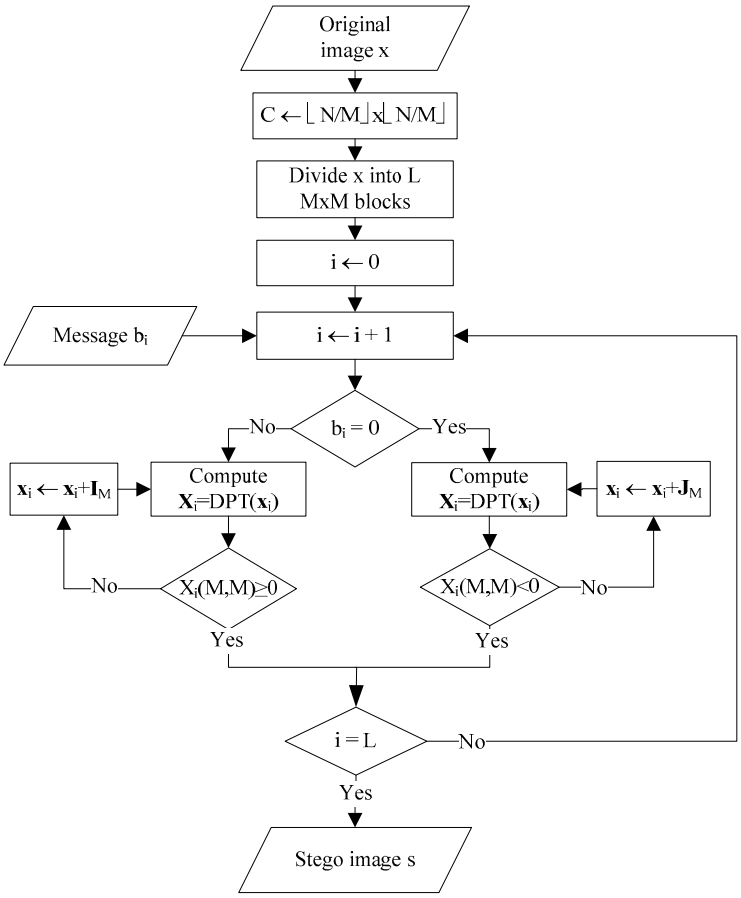

Figure 1. Embedding algorithm

Table 1. Embedding algorithm.

Embedding algorithm
Procedure embedding (image, message, M) 1. Capacity←⌊N/M⌋x⌊N/M⌋ 2. **for** i←1 **to** L **do** 3. **begin** 4. $\mathbf{x}_{MxM}(i)$←$image_{MxM}(i)$ 5. $\mathbf{X}$←DPT($\mathbf{x}_{MxM}(i)$) 6. **if** message(i)=0 **then** {X(M,M) must be negative} 7. **while** X(M,M)≥0 8. $\mathbf{x}_{MxM}(i)$←$\mathbf{x}_{MxM}(i)+\mathbf{J}_{MxM}$ 9. $\mathbf{X}$←DPT($\mathbf{x}_{MxM}(i)$) 10. **endwhile** 11. **else** {X(M,M) must be positive} 12. **while** X(M,M)<0 13. $\mathbf{x}_{MxM}(i)$←$\mathbf{x}_{MxM}(i)+\mathbf{I}_{MxM}$ 14. $\mathbf{X}$←DPT($\mathbf{x}_{MxM}(i)$) 15. **endwhile** 16. **endif** 17. $\mathbf{s}_{MxM}(i)$ ←$\mathbf{x}_{MxM}(i)$ {form the stego image} 18. **endbegin**

coefficients. The proposed embedding procedure deals with $X_M(M,M)$ coefficient, for the even sized blocks. Odd sized blocks are excluded for reasons of overflow manipulation.

Let b_i be the message bit, with $i=1,2,...,L$, where L is the payload. The payload is smaller or equal to the capacity in all cases below. For images of size NxN, the capacity equals to $\lfloor N/M \rfloor x \lfloor N/M \rfloor$ as one bit of information is hidden in each MxM pixel block. The message and the original image are the inputs to the algorithm. The output is the stego-image s, which contains the secret message and is not visible to a human observer. The embedding algorithm can be applied to each of the RGB components to increase the capacity. Other color spaces, such as YUV, could also be used. The flowchart of the embedding procedure is shown in Fig. 1 and its pseudocode is given in Table 1.

In some cases the operations in lines 8 and 13 may result in overflow. To avoid such errors, additional control is needed. When overflow occurs and the condition in line 7 is satisfied, then line 8 is modified as follows:

$$\mathbf{x}_{MxM}(i) \leftarrow \mathbf{x}_{MxM}(i) - \mathbf{I}_{MxM}$$

When overflow occurs and the condition in line 12 is satisfied, then line 13 is modified as below:

$$\mathbf{x}_{MxM}(i) \leftarrow \mathbf{x}_{MxM}(i) - \mathbf{J}_{MxM}$$

It is important to mention that this modification cannot be applied to odd sized blocks because of eq. (18). Consequently overflow in odd sized blocks cannot be avoided, thus the embedding algorithm eventually is not used for odd sized blocks. Additionally underflow, which may occur by the subtraction of the anti-diagonal identity matrix, should be taken into account (see Section 5).

4. THE EXTRACTION ALGORITHM

The hidden message may be retrieved by collecting the embedding bit of each MxM block of the **s** stego image. The extraction doesn't need any information about the original image, thus the data hiding scheme is characterized as blind. The procedure simply reads the $S(MxM)$ DPT coefficient of the MxM stego-block in order to decide a '*0*' or '*1*' for the hidden message. Figure 2 depicts the extraction algorithm and its pseudocode is given in Table 2.

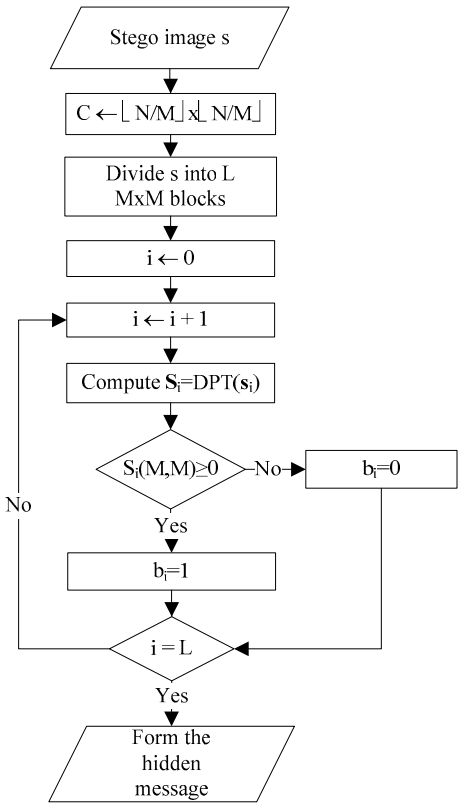

Figure 2. Extraction algorithm

Table 2. Extraction algorithm.

Extraction Algorithm
Procedure extraction (**s**, M) 1. Capacity←⌊N/M⌋x⌊N/M⌋ 2. **for** i←1 **to** L **do** 3. **begin** 4. **S**←DPT($\mathbf{s}_{MxM}$(i)) 5. **if** S(M,M)<0 **then** 6. message(i)=0 7. **else** message(i)=1 8. **endif** 9. **endbegin** 10. **endfor**

5. EXPERIMENTAL RESULTS

The performance of the proposed approach was evaluated on a number of color images for different block sizes. In Section 2 it was discussed how the addition of the identity or anti-diagonal identity matrix to the pixel values affects DPT coefficients. The addition of identity and anti-diagonal identity matrix generates the opposite alterations to the *(M,M)* DPT coefficient. Overflow can be avoided by subtracting the anti-diagonal identity matrix instead of adding the identity matrix. This is referred to in section 3 as modified line 8 and 13 of the embedding algorithm. To ensure that no additional underflow occurs, the distribution of the *(M,M)* DPT coefficients was evaluated for 15 color images and for different block sizes. Table 3 gives the average of the maximum and minimum values of *X(M,M)* for different block sizes.

Table 3. Average upper and lower values of *X(M,M)* for different block sizes.

M	Max X(M,M)	Min X(M,M)
2	135	-149
4	478	-444
8	86352	-92816
16	1740499493	-1296405698

According to Table 3 the minimum *X(2,2)* coefficient equals to -149. Let's take this as the worse case and let the embedding bit be '1'. According to the embedding procedure, *X(2,2)* coefficient must be positive, so the identity matrix should be added to the pixels of that block, until its *X(2,2)* DPT coefficient becomes positive. According to eq. (12) each time the identity matrix is added to the block, *X(2,2)* is increased by 2. Therefore the identity matrix must be added 75 times to produce 75x2=150 increment to the *X(2,2)* coefficient and therefore to achieve its sign change. Thus the pixels belonging to the block's major diagonal will increase by 75. If overflow happens, i.e. pixels are greater than 151, then embedding algorithm's modified line 13 is employed. According to this, the anti-diagonal identity matrix is subtracted until *X(2,2)* exceeds zero. Each time the subtraction takes place, *X(2,2)* is increased by 2. That is again 75 times but in this case it never underflows because two times 75 doesn't fall out of the pixel value range.

In the *4x4* block case, the average maximum *X(4,4)* equals to 478, and is modified by 20 each time the identity or anti-diagonal identity is added to the image block. The average change of pixel values equals to 23 (since *23x20=460*), which is not big enough to cause overflow. Finally in the *8x8* block case, the *X(8,8)* is changed by 3432 every time the addition is made, so 27 times (since *27x3432=92664*) are enough to change the *X(8,8)* coefficient's sign. Consequently for bigger block size, fewer additions are required.

Since the values of Table 3 correspond to the average of the $X_M(M,M)$ extrema over a collection of fifteen natural images, it is possible (although rare) for certain blocks in certain images, to exceed these values. In such a case, the following three conditions must hold in order for overflow or underflow to happen. First condition is the message's bit doesn't match with the coefficients sign, therefore algorithm's line 8 or 13 is applied. Second condition is the pixel value

belonging to the block's major diagonal to be relatively big or small so that it overflows. Last, when modified line 8 or 13 is executed, pixels belonging to the minor diagonal overflow. If all three conditions are satisfied, overflowed value is set to 0 or 256. In fact none of the pictures tested in this paper satisfy all these three conditions and for all pictures the message is correctly extracted.

Capacity in each color component of size N_1xN_2 is defined by:

$$C = \lfloor N_1 / M \rfloor x \lfloor N_2 / M \rfloor \tag{18}$$

where M is the block size used in the embedding algorithm. Since the embedding is performed in each color component independently, the total capacity is multiplied by three. Thus, the method's capacity depends only on the image and block sizes. Table 4 gives the capacity of the proposed method for different block sizes.

Table 4. Capacity of the proposed method for color images of different block sizes.

M	Capacity (bpp)
2	0.75
4	0.19
8	0.047
16	0.012

The embedding algorithm for a message of size equal to the capacity was applied to various images, leading to the experimental results in Table 5, for different block sizes. Perceptual distortion of the proposed scheme was evaluated by means of the peak signal to noise ratio (PSNR), which is defined as follows:

$$PSNR = 10 \log_{10} \frac{255^2}{MSE} \tag{19}$$

$$MSE = \frac{1}{3 \times N_1 \times N_2} \sum_{m=0}^{N_1-1} \sum_{n=0}^{N_2-1} \left[\left(x_r(m,n) - s_r(m,n) \right)^2 + \left(x_g(m,n) - s_g(m,n) \right)^2 + \left(x_b(m,n) - s_b(m,n) \right)^2 \right], \tag{20}$$

where $x_r(m,n)$, $x_g(m,n)$ and $x_b(m,n)$ are red, green and blue components of the original image and $s_r(m,n)$, $s_g(m,n)$ and $s_b(m,n)$ are the corresponding components of the stego-image.

The quality of stego-image deteriorates as message is embedded in smaller block sizes. It seems quite obvious that this is to happen since the smaller the block size used, the more data is embedded. The proposed scheme is based on DPT for deciding how much the diagonal pixels of the block should be changed in order for the required information to be hidden. When $X_M(M,M)$ coefficient value is high (block containing details), more additions are required and therefore the image block is severely distorted. Smooth blocks produce coefficients with lower magnitude and are therefore easily changed.

Three resized versions of Lenna image were used to shape Table 6. The embedding algorithm was applied to each version for different block sizes, measuring PSNR and capacity. It is obvious from Tables 5 and 6 how image quality drops as capacity increases. It seems that for larger block size there is less image degradation, but this is related to the image size and content. For large DPT frequencies more identity matrix additions are needed, so image quality drops. Figure 3 depicts the original *256x256* version of Lenna image and the stego–Lenna respectively for hiding data in different block sizes.

It is obvious that the distortion of a single block depends on its size. Indeed, block size selection of the embedding algorithm is determined by the total capacity; however, quality is the main priority. In other words when good quality is needed, the embedding algorithm must be applied on large block sizes. When capacity is important, small block sizes should be used.

Of great interest is the comparison between the proposed data hiding technique with the Wong et al method[12]. Wong et al proposed a fragile embedding algorithm which modifies the high frequency DPT coefficients according to a code book. Their method is applied to grayscale biomedical images for authentication purposes. It is implemented by dividing the image into *4x4* non-overlapping blocks and embedding one bit per block. That gives a capacity of 0.0625bpp. Implementing our method in grayscale images using *4x4* block sizes gives the same capacity as the Wong et al algorithm.

Table 5. Image distortion for different color images for embedding a message equal to the image capacity.

	PSNR (dB)			
Color Image	M=2	M=4	M=8	M=16
Lenna(512x512)	39.70	44.67	48.18	51.99
Babbon (512x512)	30.69	35.93	40.44	44.35
Peppers (512x512)	37.40	39.62	41.39	44.01
Tiffany (512x512)	39.37	44.02	47.05	50.65
F16 (512x512)	41.99	48.99	52.83	56.32
Sailboat (512x512)	35.28	38.73	42.35	47.17
Barbara (576x720)	35.45	44.78	50.95	54.75
Goldhill (576x720)	40.64	44.98	49.80	53.04
House (256x256)	41.40	47.47	51.74	54.53
Girl (256x256)	39.20	44.20	47.86	50.62
Jelly_beans (256x256)	43.36	51.46	55.14	58.79
Cornfield (480x512)	38.20	47.80	53.45	58.33
Flower (480x512)	44.32	50.34	54.67	59.14
Fruits (480x512)	41.84	50.23	56.11	61.69
Pens (480x512)	42.83	49.46	53.88	58.31

Table 6. PSNR and capacity of four resized versions of Lenna image.

M	512x512		256x256		128x128		64x64	
	PSNR (dB)	Capacity (bits)	PSNR (dB)	Capacity (bits)	PSNR (dB)	Capacity (bits)	PSNR (dB)	Capacity (bits)
2	39.07	196609	35.10	49152	31.24	12288	28.66	3072
4	44.67	49152	39.10	12288	35.65	3072	32.23	768
8	48.18	12288	43.76	3072	39.04	768	35.99	192
16	51.99	3072	46.07	768	42.10	192	42.84	48

Original | Stego (M=16, PSNR=46.07 dB) | Stego (M=8, PSNR=43.76 dB)

Stego (M=4, PSNR=39.10 dB) | Stego (M=2, PSNR=35.10 dB)

Figure 3. Original and stego – Lenna (*256x256*) for different block size message embedding.

A comparison of the image distortion for the two methods is given in Table 7. It can be noticed that the proposed method doesn't give a uniform PSNR as Wong et al method does. For different images, PSNR can vary from 37.24dB for the *baboon* image up to 53.31dB for the *jelly_beans* image. This is because every image has different *X(4,4)* distribution, therefore the message demands larger or smaller changes in the pixel values. The overall performance shows that the proposed data hiding scheme leads to higher image quality.

Table 7. PSNR comparison between the proposed and the Wong et al method for *V=4*.

	PSNR (dB)	
Grayscale Image	**Wong et al[12]**	**Proposed**
Lenna (512x512)	32.18	47.22
Baboon (512x512)	31.97	37.24
Peppers(512x512)	31.93	41.80
Tiffany (512x512)	31.93	45.40
F16 (512x512)	31.94	50.15
Sailboat (512x512)	32.08	40.89
Barbara (576x720)	31.94	45.28
Goldhill (576x720)	31.97	46.03
House (256x256)	32.03	50.06
Girl (256x256)	31.88	47.96
Jelly_beans(256x256)	32.26	53.31
Cornfield (480x512)	32.04	49.74
Flower (480x512)	31.77	52.64
Fruits (480x512)	31.82	52.85
Pens (480x512)	31.90	51.70

Table 8. PSNR comparisons between the proposed method and that of Ni et al.

		Ni et al[9]	Proposed M=2	Proposed M=4	Proposed M=8
Image	**Capacity (bpp)**	**PSNR (dB)**	**PSNR (dB)**	**PSNR (dB)**	**PSNR (dB)**
Lenna (512x512)	0.0105	53.79	59.38	56.07	52.74
Babbon (512x512)	0.0105	50.42	40.81	41.36	42.58
Peppers (512x512)	0.0105	50.08	53.19	53.23	49.36
Tiffany (512x512)	0.0167	69.89	56.32	52.47	48.69
F16 (512x512)	0.0317	53.99	55.08	59.43	54.56
Sailboat (512x512)	0.0140	53.69	48.62	48.63	45.23
Barbara (576x720)	0.0096	50.45	53.47	56.58	54.24
Goldhill (576x720)	0.0251	48.19	55.53	59.54	50.03
House (256x256)	0.0744	55.18	51.85	51.85	54.50
Girl (256x256)	0.0048	51.09	53.65	52.22	51.37
Jelly_beans (256x256)	0.0570	56.57	59.55	54.37	56.58
Cornfield (480x512)	0.0160	51.35	61.46	60.63	55.09
Flower (480x512)	0.0155	48.35	60.12	59.70	56.22
Fruits (480x512)	0.0121	52.23	59.84	60.95	59.31
Pens (480x512)	0.0102	48.34	59.14	59.13	57.66
Average	**0.0212**	**52.90**	**55.20**	**55.07**	**52.54**

The proposed method is also compared with another spatial domain data hiding technique. Ni et al[9] suggested embedding the secret message by modifying the image's histogram. The capacity of such a method equals to the histogram's maximum value. For each image tested, capacity is calculated according to Ni et al scheme and the message's size embedded for both methods equals to that of capacity. Results are presented in Table 8. The proposed method is applied by using different block sizes. There exist cases that Ni et al has better image quality. There is no practical rule derived as to for which image the proposed method has better performance. It depends on the $X_M(M,M)$ DPT coefficient, the message bit and the payload. For example smooth *Lenna* image has a better quality, in contrast to the noisy *baboon* image. That is because in noisy images the $X_M(M,M)$ coefficient is relatively higher than in smooth images. Thus in order to change this sign more additions of the identity matrix must be performed. On the other hand, in the smooth *house* image, a bigger message is embedded and consequently PSNR drops. The overall performance shows that the proposed method gives in average better image quality when using smaller block size. For big block size, the $X_M(M,M)$ coefficient

is higher, so it changes difficultly. In any case the proposed method with a capacity up to 0.25bpp for M=2 is superior to the Ni et al method.

6. CONCLUSIONS

The proposed data hiding method is blind, fragile algorithm that takes advantage of DPT properties. In fact it uses the DPT for deciding, in which way and how much to change some of the pixel date in order to hide the desired information. Specifically, image is divided in MxM blocks and the sign of $X_M(M,M)$ coefficient is examined. Negative or positive DPT coefficients, denote message bits of '0' or '1' respectively. If pixel values don't satisfy this condition, the identity or anti-diagonal identity matrix is added to the pixel block. Experimental results show that the proposed data-hiding scheme is able to embed relatively large amounts of data with low image degradation. This method can be applied to color images for tamper proofing, image authentication or secret message communication.

ACKNOWLEDGEMENT

This work was supported in part by the Greek State Scholarship Foundation (IKY) and the European Union – European Social Fund (75%), the Greek Government – Ministry of Development – General Secretariat of Research and Technology (25%) and the Private Sector in the frames of the European Competitiveness Program (Third Community Support Framework – Measure 8.3 program PENED – contact no. 03EΔ832).

REFERENCES

[1] Fridrich, J., "Steganography in Digital Media: Principles, Algorithms, and Applications", Cambridge University Press (2010).
[2] Morikowa, G., Tokura, G, "Picture Taking Apparatus and Method of Controlling Same", US Patent and Trademark Office, 2008/0025574 A1, (2008).
[3] Cox, I., Miller, M., Bloom, J.A., Fridrich, J., Kalker, T., "Digital Watermarking and Steganography", Morgan Kaufmann (2008).
[4] Balado, F., Perez-Gonzalez, F., "Provably or Probably Robust Data Hiding?", IEEE International Conference on Multimedia & Expo, (ICME2002), Lausanne, Switzerland, (2002).
[5] Tian, J., "Reversible data embedding using a difference expansion", IEEE Trans on Circuits and Systems for Video Technology Vol. 13, No. 8, 890-896 (2003).
[6] Alattar, A.M., "Reversible Watermark Using the Difference Expansion of a Generalized Integer Transform", IEEE Transactions on Image Processing Vol. 13, No. 8, 1147-1156 (2004).
[7] Hu, Y., Lee, H.-K., Li, J., "DE-Based Reversible Data Hiding With Improved Overflow Location Map", IEEE Transactions On Circuits And Systems For Video Technology Vol. 19, No. 2, February, 250-260 (2009).
[8] Hassan, M.H., Gilani, S.A.M., "A Fragile Watermarking Scheme for Color Image Authentication", Proc. World Academy of Science, Engineering and Technology Vol. 13, 312-316 (2006).
[9] Ni, Z., Shi, Y. – Q., Ansari, N., Su, W., "Reversible Data Hiding", IEEE Transactions on Circuits and Systems for Video Technology Vol. 16, No. 3, (2006).
[10] Aburdene, M.F., Goodman, T.J., "The Discrete Pascal Transform", IEEE Signal Processing Letters Vol. 12, No. 7, 493-495 (2005).
[11] Kostopoulos, H., Kandiliotis, S., Kostopoulos, I., Xenos, M., "A Digital Image Watermarking Technique using Modulated Pascal's Triangles", IASTED International Conf. Signal Processing, Pattern Recognition & Applications, Rhodes, Greece, 82-86 (2003).
[12] Wong, M.L.D, Goh, A.W.-T., Chua, H.S., "Medical Image Authentication Using DPT Watermarking: A Preliminary Attempt", Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications, Information and Multimedia Vol. 8, 41-53 (2009).
[13] Varsaki, E.E., Fotopoulos, V., Skodras, A.N., "Self-Authentication of Natural Color Images in Pascal Transform Domain", 2009 16th Int. Conference on Digital Signal Processing (DSP 2009), Santorini, Greece, (2009).
[14] Skodras, A.N., "Fast Discrete Pascal Transform", Electronics Letters Vol. 42, No. 23, 1367-1368 (2006).