An Optimization Approach to the TWPVD Method for Digital Image Steganography

Ismael R. Grajeda-Marín, Héctor A. Montes-Venegas(✉), J. Raymundo Marcial-Romero, J.A. Hernández-Servín, and Guillermo De Ita

Facultad de Ingeniería, Universidad Autónoma del Estado de México, Cerro de Coatepec s/n, Toluca, Edo. de Mexico, Mexico
tione_210@hotmail.com, {hamontesv,jrmarcialr,xoseahernandez}@uaemex.mx, deita@cs.buap.mx

Abstract. In Digital Image Steganography, Pixel-Value Differencing methods commonly use the difference between two consecutive pixel values to determine the amount of data bits that can be inserted in every pixel pair. The advantage of these methods is the overall amount of data that an image can carry. However, these algorithms frequently either overflow or underflow the pixel values resulting in an incorrect output image. To circumvent this issue, either a number of extra steps are added to adjust those values, or simply the pixels are deemed unusable and they are ignored. In this paper, we adopt the Tri-way Pixel-Value Differencing method and find an optimal pixel value for each computed pixel block such that their difference holds the maximum input data and neither underflow or overflow pixels exist.

Keywords: Optimisation · Steganography · Tri-way Pixel-Value Differencing · TWPVD

1 Introduction

Digital steganography is the set of techniques designed to conceal digital data (the payload) within a digital medium or carrier. Unlike related areas, such as cryptography or watermarking, steganography techniques aim to keep the existence of a message undetected and to continuously increase the amount of input data to be embedded [3].

In digital image steganography, the pixel intensities are used to hide the payload data. A common approach, and perhaps the simplest, is to use some form of Least Significant Bit (LSB) insertion method [1]. LSB methods replace b least significant bits of the carrier pixels with the same number of payload data bits. The less bits being replaced, the less altered the carrier image will be, but also the payload will be smaller. Some LSB substitution techniques have implemented an optimal pixel adjustment for data embedding to reduce the disruption of the carrier image [10]. Other steganographic methods include an assortment of transformation as well as masking and filtering techniques. Surveys and reviews of current methods are readily available in the literature [3,7].

J.F. Martínez-Trinidad et al. (Eds.): MCPR 2016, LNCS 9703, pp. 125–134, 2016.
DOI: 10.1007/978-3-319-39393-3_13

With the objective of increasing the amount of data that an image can carry, a set of techniques have been proposed that use the difference between two neighbour pixels to hide input data. This difference can be computed in any neighbouring direction. Wu and Tsai [11] proposed a Pixel Value Differencing (PVD) method that produces a stego-image with considerable payload data and a substancial image quality. Thereafter various approaches based on PVD have been produced [2,4,12].

Ideally, the payload must be recovered using only the resulting pixel values, and all pixels of the original image should be used to embed data in order to achieve a higher payload. However, many PVD methods yield overflow or underflow pixels (i.e., out of the valid range interval) and decide either to ignore or to somehow adjust the resulting pixel values. This, however, may lead to a lower payload or to include additional strategies to retrieve the embedded data [11] that may reveal the existence of a hidden message.

In this paper, we adopt the Tri-way Pixel-Value Differencing method and find an optimal pixel value for each computed pixel block such that their difference holds the maximum input data. Our method reduces the size of the search space and computes a much more smaller set of feasible solutions. In addition, two more strategies are discussed to further increase the size of the embedded payload. The method is designed in such a way that the resulting pixel intensities are never out of the valid interval and it uses all pixel blocks to carry payload data. A series of experimental results show the feasibility of the method.

We begin in Sect. 2 by covering the basics of the Pixel-Value Differencing method. Section 3 presents a detailed description of our two optimisation algorithm approaches. Section 4 presents several experimental results, and Sect. 5 concludes the paper.

2 Pixel-Value Differencing

The PVD method [11] assumes that the payload is a continuos stream of input bits that represent any type of digital data. The PVD embeds data using the intensity difference of two contiguous pixels. The idea is to modify these pixels by adding a decimal conversion of some input data bits in such a way that their value difference is kept to preserve the image quality. Regions in the image with larger differences in pixel intensities can carry more pieces of payload than others. This usually happens in the areas with evident edges and less frequently in smoother regions. The method provides a good embedding capacity but is prone to be detected using statistical based stego-analysis methods [5].

Chang *et al.* [2] proposed a modified version of the PVD named *Tri-way Pixel-Value Differencing* (TWPVD). Whereas the PVD inserts data in only one pixel pair, the TWPVD uses horizontal, vertical and diagonal diferences (hence its name) in 2×2 pixel blocks to hide input data, thus achieving a higher payload than the PVD in the carrier image. One problem arising in PVD based methods is that they frequently yield overflow/underflow pixel values. These pixels are either adjusted or ignored by the method, thus reducing the number of pixels available to carry data payload [2,6].

2.1 Tri-Way Pixel-Value Differencing

The Tri-way Pixel-Value Differencing (TWPVD) method was designed to get more pixels involved in the data embedding process [2]. The TWPVD divides the carrier image into non-overlapping blocks of 2×2 consecutive pixels. Three difference values are computed in each block from the values of two neighbour pixels in three distinctive directions. The first difference is computed between the pixel in the upper left corner, namely the pivot, and the pixel on its right. The second difference is between the pivot and the pixel in the opposite corner, and the third one is also between the pivot and the pixel below it. Each difference belongs to one of a predefined set of range intervals which, in turn, determines the number of bits to be inserted in every pixel pair. Each range interval R_k has a lower l_k and an upper u_k value listed in the form of a range table. The range table has been designed simply by computing each interval width using a power of two, either to provide large capacity or to provide high imperceptibility [11,12]. Other approaches have designed the range table based on the *perfect square number* [9], or have opted for entirely replacing the range table with a well crafted function based on the floor and ceiling functions [4].

Regardless of how these range intervals are produced, the TWPVD algorithm follows these steps:

1. Compute the differences $d_i = p_i - p_1$ within the pixel block $i \in \{1, 2, 3, 4\}$, where $\begin{array}{|c|c|}\hline p_1 & p_2 \\ \hline p_3 & p_4 \\ \hline \end{array}$
2. Locate for each d_i the range k such that $l_k \leq |d_i| \leq u_k$
3. Compute the amount of input data bits t_i to be inserted in the difference i of the block p_i as follows:

$$t_i = \begin{cases} 0 & \text{if } i = 1 \\ \lfloor \log_2(u_k - l_k + 1) \rfloor & \text{otherwise} \end{cases} \tag{1}$$

4. Compute the decimal representation b_i of the t_i bits
5. A new d'_i is computed for each d_i

$$d'_i = l_{k_i} + b_i \tag{2}$$

6. Later the TWPVD uses each d'_i to compute the values of the resulting pixels p'_i using a well crafted set of rules [2]. We have adopted the TWPVD by replacing these rules with an *optimisation* strategy to determine the best pixel values that hold the maximum payload.

A closer look to this algorithm, reveals that it also produces overflow/underflow pixel values that are simply skipped as data payload carriers. Worse still, TWPVD authors [2] do not seem to discuss how the extraction algorithm knows which pixels are being ignored [4]. This is fundamental to guarantee the integrity of the secret message.

3 An Optimisation Approach to Modify the TWPVD

Any PVD method can be seen as an optimisation problem as follows: Given ${d'}_i$ and p_i, search for a solution ${p'}_i$ subject to the following set of conditions:

1. Overflow/underflow must be prevented subject to $0 \leq {p'}_i \leq 255$
2. Retrieving the payload data is subject to ${d'}_i = |{p'}_i - {p'}_1|$, where p'_i and p'_1 are now variables to be searched as an optimization problem which will define the stego-image.
3. Distortion of the resulting image must be subject to minimize the objective function

$$f(p_i, {p'}_i) = \sum_{i=1}^{4} (p_i - {p'}_i)^2 \tag{3}$$

We know that ${p'}_i = |{d'}_i|$ is a solution, *i.e.* $p_1 = 0$, that fulfills conditions 1 y 2, but does not fulfill condition 3 because it causes a major distortion to the resulting stego-image. Nonetheless, the solution shows that there exist at least one solution for any given input.

Since there are 4 pixels per block in the range [0..255], we can easily estimate the size of the search space to be 2^{32} possible pixel value combinations times the carrier image dimensions divided by 4. These solutions take far too long to be explored efficiently, as shown in Table 1.

Table 1. Comparison between Optimal-TWPVD and a simply Brute Force strategy added to the TWPVD. The time performance advantage is clear.

	bpp		PSNR		Time	
	BFTWPVD	OTWPVD	BFTWPVD	OTWPVD	BFTWPVD	OTWPVD
Barbara	2.54	2.54	36.50	36.50	471.60	9.56
Airplane	2.37	2.37	38.90	38.90	458.35	10.02
Boat	2.41	2.41	38.19	38.19	394.81	8.35
Goldhill	2.38	2.38	38.73	38.73	323.85	17.72
Lena	2.35	2.35	39.34	39.34	366.10	11.80
Average	2.41	2.41	38.33	38.33	402.94	11.49

One alternative is to reduce the size of the search space so that it can be explored in useful times.

Using equation from condition 2 it follows that

$${p'}_i = \pm {d'}_i + {p'}_1 \tag{4}$$

This evidently means that we can compute ${p'}_i$ using the two following variables:

1. $\pm d'_i$ takes the different sign combinations for d'_i. These combinations are 8 because d'_1 is always 0 and d'_2, d'_3, d'_4 only can take 2 different values: one positive and one negative of equal magnitude.
2. p'_1 must be subject to $0 \leq p'_1 \leq 255$. This means that p'_1 only can take 256 distinct values.

This further reduces the size of the search space to 2^{11}. A search space of this size can be readily explored in its entirety. That is, all possible values for p'_1 must be combined with all possible values for $\pm d'_i$.

3.1 An Additional Optimisation Strategy

We now describe an additional optimisation strategy to further increase the payload inserted by the method from Sect. 3. Such strategy is based on the first derivative of the objective function with respect of p'_1 and discards the overflow/underflow solutions.

Using Eqs. 3 and 4, a quadratic function can be produced in terms of p'_1, namely:

$$f(p'_1) = \sum_{i=1}^{4} (\pm d_i + p'_1 - p_i)^2 \tag{5}$$

Eight different quadratic curves can be plotted from the eight different combinations of signs in $\pm d_i$. When computing the first derivative of these functions, a point for each curve can be found for which f is minimum:

$$p'_1 = \frac{1}{4}\sum_{i=1}^{4} p_i - \frac{1}{4}\sum_{i=1}^{4} \pm d_i \tag{6}$$

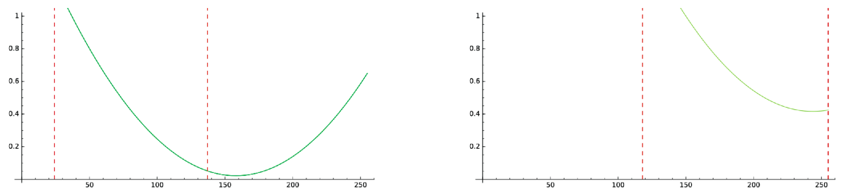
The resulting 8 candidate values for p'_1 can become 16 because Eq. (6) can yield real numbers that need to be converted into integers using both the *ceil* and *floor* functions.

In some cases the optimal point can be off the valid range or can even cause some of the other 3 pixels to be off. It is necessary then to move that point within the proper range as that value is potentially a solution.

Figure 1 shows 2 curves plotted using the objective function. These curves are bounded between a pair of dotted lines representing the upper and lower bounds valid for p'_1. It also shows that the points of minimum value are not always within the valid interval and is necessary to move that point to a valid area.

Equation (4) can yield valid intervals for each curve as $\max(\pm d_i) \leq p'_1 \leq \min(\pm d_i + 255)$. From this equation, we can define the adjustment function:

$$A(p, M, m) = \begin{cases} 0 & if\ M \leq p \leq m \\ M - p & if\ p < M \\ -(p - m) & if\ p > m \end{cases} \tag{7}$$



a) Optimal point *outside* the valid interval b) Optimal point *inside* the valid interval

Fig. 1. Two different objective function graphs

Therefore the optimal point in the valid range would be defined as:

$$p'_1 = p'_1 + A(p'_1, \max(\pm d_i), \min(\pm d_i) + 255) \tag{8}$$

As mentioned before, this point needs to be adjusted using the *ceil* or $floor$ functions. Both functions yield an identical or extremely close value. Because there are 8 curves each with 2 solutions, we end up with a new search space of only 16 potential solutions.

The algorithm follows these steps:

1. Go through steps 1–5 of the algorithm from Sect. 2.1
2. Compute $s_i = \pm d_i + p'_1$ using the *ceil* or $floor$ functions. Overflow/underflow solutions are discarded
3. The optimal solution is given by $p'_i = \min(f(p_i, s_i))$
4. Replace the original 2×2 pixel block with the optimal solution found
5. Repeat for each 2×2 pixel block of the carrier image

To recover the secret message, the inverse process is applied as follows:

1. Divide the carrier image into non-overlapping blocks of 2×2 consecutive pixels
2. Compute the differences $d_i = p_i - p_1$ within the pixel block $i \in \{1, 2, 3, 4\}$
3. For each d_i locate the table range $r_i = k$ such that $l_k \leq |d_i| \leq u_k$
4. Compute the number of inserted bits in each difference
$t_i = \begin{cases} 0 & \text{if } i = 1 \\ \lfloor \log_2(u_{r_i} - l_{r_i} + 1) \rfloor & \text{otherwise} \end{cases}$
5. The entire data payload is recovered by concatenating the binary representation of $b_i = d_i - l_{r_i}$

3.2 Inserting an Extra Bit

The method can insert an additional bit to further increase the secret message inserted in each 2×2 block with a minimal deterioration to the carrier image.

The $floor$ and *ceil* functions yield two consecutive integer numbers that produce very close or even identical Objective Function results. This type of function curves constantly appear and are used as indication for inserting an additional bit of the secret message. This additional bit is called β. If $\beta = 0$, ${p'}_1$ must be even, if $\beta = 1$, ${p'}_1$ must be odd.

To find the optimal we say that $2c = {p'}_1 - \beta$ and modify Eq. 4 as follows:

$$ {p'}_i = \pm d_i + 2c + \beta \tag{9} $$

$$ f(c) = \sum_{i=1}^{4} (\pm d_i + 2c + \beta - p_i)^2 \tag{10} $$

Therefore, the valid interval for the optimisation problem is given by:

$$ c = \frac{1}{8}\sum_{i=1}^{4} p_i - \frac{1}{8}\sum_{i=1}^{4} \pm d_i - \frac{1}{2}\beta \tag{11} $$

$$ c = c + A(c, \max(-\frac{\beta}{2} - \frac{1}{2}(\pm d_i)), \min(-\frac{\beta}{2} - \frac{1}{2} \pm (d_i) + 255)) \tag{12} $$

The algorithm is also modified as follows:

1. Go through steps 1–5 of the algorithm from Sect. 2.1
2. Compute $s_i = \pm d_i + c$ using the *ceil* or *floor* functions. Overflow/underflow solutions are discarded
3. The optimal solution is given by ${p'}_i = \min(f(p_i, s_i))$
4. Replace the original 2×2 pixel block with the optimal solution found
5. Repeat for each pixel block of the carrier image

To recover the message payload, the same steps from Sect. 3.1 are used, and an extra bit 0 is added to the message if p_1 is even or a 1 otherwise.

4 Experimental Results

A set of images were used to test the performance of our algorithms and to compare our results to those previously published in the literature. All carrier images, shown in Fig. 2, are 8-bit grayscale images of size 512×512. These images belong to a larger set that have become a *de facto* standard in Image Processing and Computer Vision experiments for testing new developments. We also have chosen these images to compare our results with previous work by Peng *et al.* [8] and Hernandez-Servin *et al.* [4]. Both authors compared their own results with work previously published. In addition, we also compare the performance of our algorithm with the results of the TWPVD [2].

The peak signal-to-noise ratio (PSNR) is used to measure the difference between the original carrier image and the image with the message payload. The higher the PSNR, the better the quality of the stego image. The number of bits per pixel (bpp) for each test image, is computed simply by dividing the number of bits inserted by the number of pixels in the carrier image.

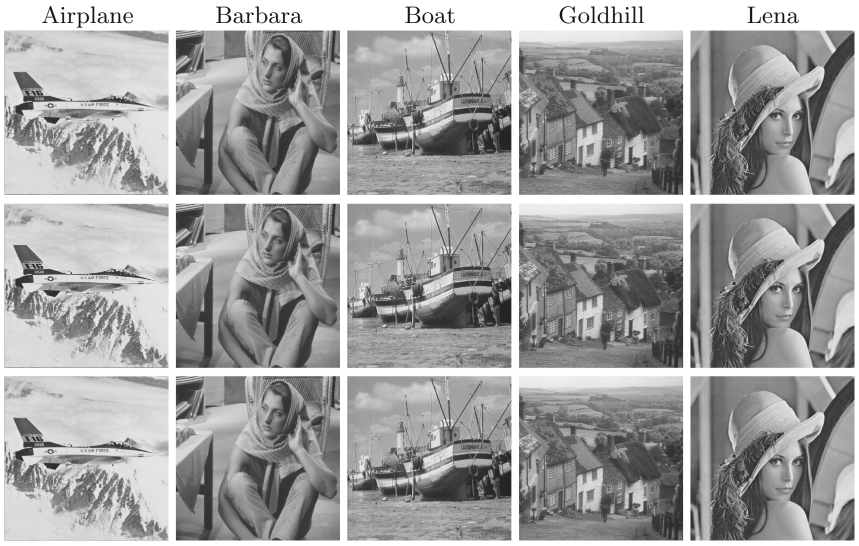


Fig. 2. Original images (first row). Resulting stego images using the Optimal-TWPVD (second row). Resulting stego images using the OTWPVD and Extra Bit Insertion (third row)

Table 2 shows a comparison between the Optimal-TWPVD and the Extra Bit Insertion algorithms. While the former shows a better performance than previous work (as shown in Table 4), the latter further increases the overall results in terms of both the amount of data payload (the *bpp*) inserted, and the image distortion measured with the PSNR in all images tested. This might seem expected as both the Optimal TWPVD and the Extra Bit Insertion strategies use every 2×2 block to carry data payload. No pixel block is ignored and no pixel overflow/underflow occurred.

Table 2. Comparison between the Optimal-TWPVD and the Optimal-TWPVD with Extra Bit Insertion

	bpp		PSNR		Time	
	OTWPVD	EOTWPVD	OTWPVD	EOTWPVD	OTWPVD	EOTWPVD
Barbara	2.54	2.79	36.50	36.43	9.56	16.21
Airplane	2.37	2.62	38.90	38.76	10.02	17.99
Boat	2.41	2.66	38.19	38.09	8.35	20.24
Goldhill	2.38	2.63	38.73	38.64	17.72	16.53
Lena	2.35	2.60	39.34	39.17	11.80	15.65
Average	2.41	2.66	38.33	38.22	11.49	17.32

We also compare our results with those from the TWPVD [2] in Table 3. Since our algorithms search for the optimal pixel values for each block, the results are superior in terms of both *bpp* and PSNR. The general notion is that less data embedded should result in less distortion of the carrier image, which is not observed by comparing the PSNR values of our experiments.

Table 3. Comparison between the TWPVD and a our Optimal-TWPVD

	bpp		PSNR		Time	
	TWPVD	OTWPVD	TWPVD	OTWPVD	TWPVD	OTWPVD
Barbara	2.54	2.54	36.38	36.50	1.55	9.56
Airplane	2.37	2.37	38.23	38.90	1.90	10.02
Boat	2.40	2.41	37.72	38.19	1.88	8.35
Goldhill	2.38	2.38	38.09	38.73	1.84	17.72
Lena	2.35	2.35	38.61	39.34	2.19	11.80
Average	2.41	2.41	37.81	38.33	1.87	11.49

A similar comparison with recent results by Peng *et al.* [8] and Hernandez-Servin *et al.* [4] is shown in Table 4. This table also shows favorable results in terms of both data payload carried and stego image quality.

Table 4. Comparison between our proposals Optimal-TWPVD & Extra Bit OTW-PVD, and Hernadez-Servin *et al.* [4] and Peng *et al.* [8]

bpp					PSNR			
	OTWPVD	EOTWPVD	[4]	[8]	OTWPVD	EOTWPVD	[4]	[8]
Barbara	2.54	2.79	1.38	1.20	36.50	36.43	36.04	30.75
Airplane	2.37	2.62	1.30	1.20	38.90	38.76	36.09	33.45
Boat	2.41	2.66	1.80	1.20	38.19	38.09	34.56	26.66
Goldhill	2.38	2.63	1.66	1.20	38.73	38.64	37.03	30.70
Lena	2.35	2.60	1.60	1.20	39.34	39.17	37.55	26.89
Average	2.41	2.66	1.55	1.20	38.33	38.22	36.25	29.69

5 Conclusions

This work designs an optimisation algorithm that modifies and improves the TWPVD [2] steganographic method. It is favourably compared against the TWPVD and also against recent results by Peng *et al.* [8] and Hernandez-Servin *et al.* [4].

Our results show improvements in several important aspects, namely, (1) Number of Bits per pixel inserted, (2) Better stego image quality measured with the PSNR, (3) No overflow/underflow pixels are produced, and (4) No blocks of pixels are skipped or ignored as data carriers.

The major merit of our algorithms is to reduce the feasible set of possible pixel values for each block such that the search for the best solution in terms of both data payload and stego image quality can be efficiently conducted.

There are a couple directions in which this work may proceed. The first logical next step is to use the method to hide data payload into color images. The obvious result would be to achieve a very large payload insertion, but the effects in color and general image distortion may either require to adapt or entirely change the algorithm.

References

1. Chan, C.K., Cheng, L.M.: Hiding data in images by simple LSB substitution. Pattern Recogn. **37**(3), 469–474 (2004)
2. Chang, K.C., Chang, C.P., Huang, P.S., Tu, T.M.: A novel image steganographic method using Tri-way Pixel-Value Differencing. J. Multimedia **3**(2), 37–44 (2008)
3. Cheddad, A., Condell, J., Curran, K., Kevitt, P.M.: Digital image steganography: survey and analysis of current methods. Sig. Process. **90**(3), 727–752 (2010). http://www.sciencedirect.com/science/article/pii/S0165168409003648
4. Hernández-Servin, J.A., Marcial-Romero, J.R., Jiménez, V.M., Montes-Venegas, H.A.: A modification of the TPVD algorithm for data embedding. In: Carrasco-Ochoa, J.A., Martí-nez-Trinidad, J.F., Sossa-Azuela, J.H., Olvera López, J.A., Famili, F. (eds.) MCPR 2015. LNCS, vol. 9116, pp. 74–83. Springer, Heidelberg (2015)
5. Mahajan, M., Kaur, N.: Adaptive steganography: a survey of recent statistical aware steganography techniques. Int. J. Comput. Netw. Inf. Secur. (IJCNIS) **4**(10), 76–92 (2012)
6. Mandal, J.K., Das, D.: Steganography using adaptive pixel value differencing (APVD) of gray images through exclusion of overflow/underflow (2012). CoRR arXiv:1205.6775
7. Mishra, M., Mishra, P., Adhikary, M.C.: Digital image data hiding techniques: a comparative study. ANSVESA **7**(2), 105–115 (2012)
8. Peng, F., Li, X., Yang, B.: Adaptive reversible data hiding scheme based on integer transform. Sig. Process. **92**(1), 54–62 (2012)
9. Tseng, H.W., Leng, H.S.: A steganographic method based on pixel-value differencing and the perfect square number. J. Appl. Math. **2013**(1), 1–8 (2013)
10. Wang, R.Z., Lin, C.F., Lin, J.C.: Image hiding by optimal LSB substitution and genetic algorithm. Pattern Recogn. **34**(3), 671–683 (2001)
11. Wu, D.C., Tsai, W.H.: A steganographic method for images by pixel-value differencing. Pattern Recogn. Lett. **24**(9), 1613–1626 (2003)
12. Wu, H.C., Wu, N.I., Tsai, C.S., Hwang, M.S.: Image steganographic scheme based on pixel-value differencing and lsb replacement methods. IEE Proc. Vis. Image Sig. Process. **152**(5), 611–615 (2005)