PREDICTION-ERROR-ORDERING FOR HIGH-FIDELITY REVERSIBLE DATA HIDING

Ioan Catalin Dragoi and Dinu Coltuc

Faculty of Electrical Engineering, Electronics and Information Technology
Valahia University of Targoviste, Romania
Email: {catalin.dragoi, dinu.coltuc}@valahia.ro

ABSTRACT

Pixel-value-ordering (PVO) appears as an efficient solution for high-fidelity reversible data hiding. State-of-the-art PVO schemes split the host image into blocks, sort pixels within the blocks based on their graylevel and, finally, embed data into some differences between the sorted values. This paper investigates the use of the prediction error instead of the pixel graylevel both for ordering and embedding. The proposed prediction-error-ordering (PEO) scheme also introduces a two-stage procedure by splitting each block in two sets following a chessboard pattern and by processing set by set each block. The pixels of one set are used to classify and predict the ones of the other set and vice-versa. The proposed PEO approach outperforms the state-of-the-art PVO schemes.

Index Terms— Reversible data hiding, pixel-value-ordering, pairwise embedding

1. INTRODUCTION

Reversible data hiding (RDH) deals with data embedding into digital hosts, with the major constraint that both the embedded data and the hosts should be exactly recovered [1]. High-fidelity RDH limits the distortion of the host to maximum one graylevel per pixel.

By a proper selection of the embedding parameters, high-fidelity RDH can be obtained with the classical histogram shifting (HS) or the prediction-error expansion (PEE) approaches. HS makes room for embedding data into a selected bin of a feature histogram (graylevel [2], or prediction error [3], etc.) by shifting with one position the bins situated to the left or the right of the selected bin. A hidden data bit is embedded into each pixel with the selected feature value by keeping it unchanged for "0" and by changing the value with one graylevel in the chosen direction for "1". For a single embedding level, the distortion introduced by HS cannot exceed one graylevel. PEE embeds data into the pixels with prediction errors in a predefined interval around zero, (usually $[-T, T)$, see [4, 5, 6]). The LSB of the prediction error is cleared by two times expansion, making room for hiding one data bit. The pixels with prediction errors outside the selected interval are shifted by $\pm T$ positions. High-fidelity RDH is provided by embedding into the pixels with prediction errors in $[-1, 0)$.

For high fidelity RDH, both HS and PEE are outperformed by the more recent approaches based on pairwise embedding and pixel value ordering (PVO). The pairwise RDH framework of [7] processes the host pixels as pairs. A prediction error (or difference) pair is computed for each pixel pair. Specific embedding equations derived from HS are used to embed data in a more efficient manner. More precisely, instead of embedding two bits of data, i.e. the combinations of bits "00", "01", "10" and "11", the last combination is eliminated, maintaining the embedding distortion within the ± 1 limit. The pairs in [7] are formed by grouping pixel around the diagonal direction. A pairing algorithm that adaptively selects the pairs based on the corresponding prediction errors was proposed in [8].

The PVO based RDH framework introduced in [9] splits the host image into equally sized blocks. The pixels in each block are sorted based on their graylevel value. The first and last pixel in the sorted order serve as potential hosts for the hidden data. A difference value is computed based on a reference pixel for each potential host. A hidden data bit is inserted into a pixel if its corresponding difference value is 1, otherwise its value is shifted outwards by one graylevel. The IPVO approach of [10] improves the difference computation process. PVO was also introduced as a predictor for HS in [11] and PEE in [12]. In [13] PVO was combined with pairwise embedding, an approach that was further refined in [14]. A new PVO approach with variable block sizes based on quadtree partitions was also recently proposed in [15].

The PVO approaches proposed so far sort pixels within blocks based on their graylevel. Furthermore, graylevel differences between the extrema pixels and their references are used to select the candidates for embedding. This paper proposes the use of prediction error instead of pixel graylevel both for ordering and embedding. Since prediction error is Laplacian distributed, the newly proposed prediction-error ordering (PEO) approach is expected to provide more candidates for data embedding. Another original aspect of the proposed PEO is the splitting of each block into cross and dot

This work was supported by UEFISCDI Romania, in the frame of PNIII-P4-IDPCE-2016-0339 and PN-III-P1-1.1-PD-2016-1666 Grants.

sets, as in [4], and a two-stage block processing scheme. The two-stage processing ensures the reversible computation of the prediction error.

The outline of the paper is as follows. The pairwise IPVO scheme of [14] is discussed in Section 2. The proposed PEO approach is introduced in Section 3. The experimental results are presented in Section 4 and the conclusions are drawn in Section 5.

2. PAIRWISE IPVO

The pairwise IPVO scheme introduced in [14] improves the RDH framework of [13]. The host image is first split into disjoint, equally sized blocks. The complexity of each block is evaluated based on a local complexity value, l_c. Two thresholds, t_1 and t_2, are then used to separate the blocks into three groups: smooth ($l_c \leq t_1$), slightly noisy ($t_1 < l_c \leq t_2$) and noisy ($l_c > t_2$).

The smooth and slightly noisy blocks are processed one by one in raster scan order (from left to right and top to bottom). Let blocks have n pixels. The pixels in each block are sorted in ascending order based on their graylevel value: $x_{\sigma(1)} \leq x_{\sigma(2)} \leq ... \leq x_{\sigma(n)}$. For $x_{\sigma(i)}$, the pixel with the i^{th} largest graylevel in the block, $\sigma(i)$ represents its corresponding position in the non-sorted block. Noisy blocks are not used for data hiding and are skipped by the embedding algorithm.

Before going any further, it is important to mention that the sorting order is only used to identify and process the host pixels. The embedding algorithm does not change the position of pixels in the image.

2.1. Embedding in slightly noisy blocks

Two pixels from each slightly noisy block are selected as potential hosts for the hidden data: $x_{\sigma(1)}$ and $x_{\sigma(n)}$. Two other pixels, $x_{\sigma(2)}$ and $x_{\sigma(n-1)}$, are selected as reference values for the host pixels. Their positions in the non-sorted block are evaluated:

$$\begin{array}{ll} u_1 = \min(\sigma(1), \sigma(2)) & u_2 = \min(\sigma(n-1), \sigma(n)) \\ v_1 = \max(\sigma(1), \sigma(2)) & v_2 = \max(\sigma(n-1), \sigma(n)) \end{array} \quad (1)$$

The differences between the selected host pixels and their closest reference values are computed as:

$$d_l = x_{u_1} - x_{v_1} \quad d_r = x_{u_2} - x_{v_2} \quad (2)$$

Note that d_l is either $x_{\sigma(1)} - x_{\sigma(2)}$ or $x_{\sigma(2)} - x_{\sigma(1)}$ (controlled by (1)).

$x_{\sigma(1)}$ and $x_{\sigma(n)}$ are further used as hosts if their corresponding difference values are either $\{0\}$ or $\{1\}$, otherwise they are shifted outwards by one graylevel:

$$x'_{\sigma(1)} = \begin{cases} x_{\sigma(1)} - w & \text{if } d_l \in \{0,1\} \\ x_{\sigma(1)} - 1 & \text{if } d_l < 0 \text{ or } d_l > 1 \end{cases} \quad (3)$$

$$x'_{\sigma(n)} = \begin{cases} x_{\sigma(n)} + w & \text{if } d_r \in \{0,1\} \\ x_{\sigma(n)} + 1 & \text{if } d_r < 0 \text{ or } d_r > 1 \end{cases} \quad (4)$$

where $w \in \{0,1\}$ is the hidden data bit.

2.2. Embedding in smooth blocks

Four pixels from each smooth block are selected as potential hosts: $x_{\sigma(1)}, x_{\sigma(2)}, x_{\sigma(n-1)}$ and $x_{\sigma(n)}$. Their positions in the non-sorted block are evaluated with (1).

The $x_{\sigma(3)}$ and $x_{\sigma(n-2)}$ pixels are selected as references. Their positions are also evaluated with respect to the host pixel positions:

$$\begin{array}{ll} u_{11} = \min(u_1, \sigma(3)) & u_{21} = \min(u_2, \sigma(n-2)) \\ v_{11} = \max(u_1, \sigma(3)) & v_{22} = \max(u_2, \sigma(n-2)) \\ u_{12} = \min(v_1, \sigma(3)) & u_{22} = \min(v_2, \sigma(n-2)) \\ v_{12} = \max(v_1, \sigma(3)) & v_{22} = \max(v_2, \sigma(n-2)) \end{array} \quad (5)$$

Four difference values are then computed:

$$\begin{array}{ll} d_{11} = x_{u_{11}} - x_{v_{11}} & d_{21} = x_{u_{21}} - x_{v_{21}} \\ d_{12} = x_{u_{12}} - x_{v_{12}} & d_{22} = x_{u_{22}} - x_{v_{22}} \end{array} \quad (6)$$

The pixels are processed as pairs, namely (x_{u_1}, x_{v_1}) and (x_{u_2}, x_{v_2}). These pairs are classified into groups based on the $\{0\}$ and $\{1\}$ selected differences:

$$(x_{u_i}, x_{v_i}) \in \begin{cases} A & \text{if } d_{i1} \in \{0,1\} \text{ and } d_{i2} \in \{0,1\} \\ B & \text{if } d_{i1} \in \{-1,2\} \text{ and } d_{i2} \in \{-1,2\} \\ C & \text{if } d_{i1} \in \{0,1\} \text{ and } d_{i2} \notin \{0,1\} \\ D & \text{if } d_{i1} \notin \{0,1\} \text{ and } d_{i2} \in \{0,1\} \\ E & \text{otherwise} \end{cases} \quad (7)$$

with $i \in \{1,2\}$.

Based on their group, the pairs are either embedded with data or shifted:

$$(x'_{u_i}, x'_{v_i}) = \begin{cases} (x_{u_i}, x_{v_i}) - (w_1, w_2) & \text{if } (x_{u_i}, x_{v_i}) \in A \\ (x_{u_i}, x_{v_i}) - (w, w) & \text{if } (x_{u_i}, x_{v_i}) \in B \\ (x_{u_i}, x_{v_i}) - (w, 1) & \text{if } (x_{u_i}, x_{v_i}) \in C \\ (x_{u_i}, x_{v_i}) - (1, w) & \text{if } (x_{u_i}, x_{v_i}) \in D \\ (x_{u_i}, x_{v_i}) - (1, 1) & \text{if } (x_{u1i}, x_{v_i}) \in E \end{cases} \quad (8)$$

where $(w_1, w_2) \in \{(0,0), (0,1), (1,0)\}$ and $w \in \{0,1\}$.

2.3. Embedding parameters and data extraction

The embedding parameters (t_1, t_2 and the size of the pixel blocks) are stored in a reserved area of the image by LSB substitution. The substituted LSBs are appended to the hidden data before the embedding stage. The optimal embedding parameters for a given payload size can be determined by exhaustive search or by using a linear programing model (see [14]). The overflow/underflow at embedding or shifting is solved by using a location map: overflow/underflow pixels are left unchanged and their locations are stored in the map. After the embedding stage is concluded, the map is compressed and stored in the reserved area.

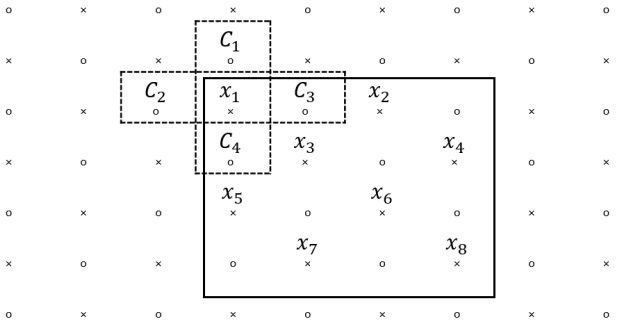

Fig. 1. The $X_\times$ and X_o sets, the eight $x_i \in X_\times$ pixels of a 4×4 block and the prediction context for x_1.

At the decoding stage, the embedding parameters and the overflow map are first extracted from the reserved area. The local complexity for each block is computed and the blocks are separated into the three groups. The original blocks are restored by reverting the corresponding operations (either (3), (4) or (8)). After the entire hidden content was extracted, the original reserved area LSBs are removed from the extracted bit-stream and are used to restore the reserved area.

3. THE PROPOSED PEO APPROACH

The host image is first divided into $m_b \times n_b$ pixel blocks, where both m_b and n_b have even values. Similarly to [4], the pixels are also split into two sets: $X_\times$ and X_o (cross and dot), forming a chessboard pattern (Fig. 1). A pixel belonging to $X_\times$ has to its left, right, top and bottom pixels belonging to X_o and vice-versa.

The embedding of the data proceeds in two stages, corresponding to $X_\times$ and X_o, respectively. Each stage provides half of the required embedding capacity. Since both m_b and n_b are even, the smallest block size for the proposed approach is 4×4. For [13] and [14] this size is 2×3. The difference between the minimum block sizes of our approach and the ones of [13, 14] is attenuated by the splitting of the blocks in two sets and the separate embedding of $X_\times$ and X_o pixels that doubles the possible host pixels per block.

The $X_\times$ pixels are processed first. For each block, the local complexity is evaluated using the pixels from the same block that are part of X_o:

$$l_c = \frac{2}{m_b n_b} \sum_{i=1}^{m_b n_b/2} |x_i - \mu_c|, \; x_i \in X_o \tag{9}$$

where $|a| = \begin{cases} a & \text{if } a > 0 \\ -a & \text{if } a \leq 0 \end{cases}$ and $\mu_c = \frac{2}{m_b n_b} \sum_{i=1}^{m_b n_b/2} x_i$. Based on their corresponding l_c value, the blocks are classified into smooth, slightly noisy and noisy groups (as described in the previous section).

Fig. 2. The six classic test images and the Kodak set.

The pixels of X_o are also used to predict the ones of $X_\times$. Let $\hat{x}$ be the predicted value of x, $\hat{x}$ is computed using the rhombus average of [4]:

$$\hat{x} = \left\lfloor \frac{c_1 + c_2 + c_3 + c_4}{4} + \frac{1}{2} \right\rfloor \tag{10}$$

where c_1, c_2, c_3, c_4 is the prediction context of x (shown in Fig. 1) and $\lfloor a \rfloor$ represents the greatest integer less than or equal to a. The prediction error is then determined for each $x \in X_\times$ in the current block:

$$e = x - \hat{x} \tag{11}$$

The $X_\times$ pixels from smooth and slightly noisy blocks can now be sorted based on their prediction errors. Let $e_{\sigma(1)} \leq e_{\sigma(2)} \leq ... \leq e_{\sigma(n)}$ be the derived ordering.

For slightly noisy blocks, equation (1) is used to evaluate the original pixel positions. The difference values used for embedding are then computed as:

$$d_l = e_{u_1} - e_{v_1} \quad d_r = e_{u_2} - e_{v_2} \tag{12}$$

These difference values are then used in equations (3) and (4) to insert hidden bits in the corresponding pixels. Note that because e is computed based on the pixel value x (equation (11)), any modification upon x will equally affect e.

A similar approach is used for smooth blocks. With equations (1) and (5), the difference values follows as:

$$\begin{array}{ll} d_{11} = e_{u_{11}} - e_{v_{11}} & d_{21} = e_{u_{21}} - e_{v_{21}} \\ d_{12} = e_{u_{12}} - e_{v_{12}} & d_{22} = e_{u_{22}} - e_{v_{22}} \end{array} \tag{13}$$

The differences are then used in equation (8) to insert the hidden data.

After the $X_\times$ set was used for data hiding, the embedding process is repeated for the X_o set. The modified values in $X_\times$ are used to predict the pixels in X_o and to evaluate the local complexity of the blocks.

The data extraction starts with the recovery of the parameters from the reserved area. The extraction stages are performed in reverse order, starting with the X_o set. The $X_\times$ set is processed after the X_o pixels are restored. It should also be mentioned that the reserved area together with the first/last pixel columns and rows are not considered for data hiding.

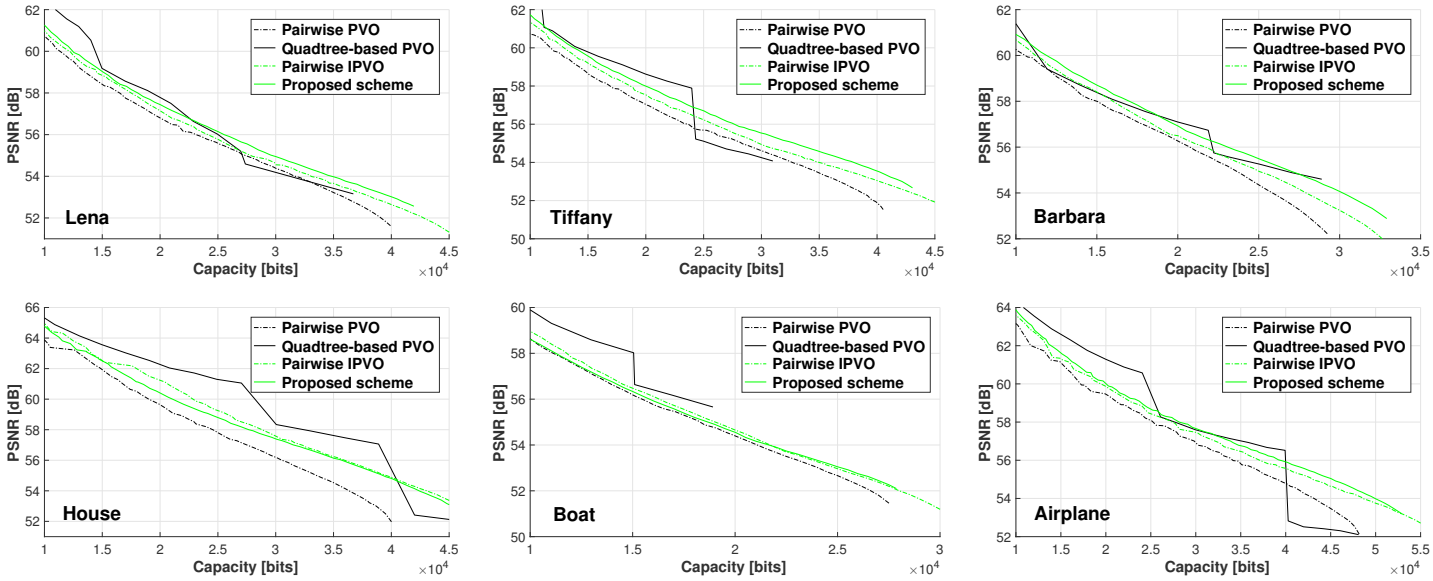

Fig. 3. PSNR/Embedded bits results for the pairwise PVO scheme of [13], the pairwise IPVO scheme of [14], the quadtree-based PVO of [15] and the proposed PEO approach.

Table 1. PSNR comparison ([dB]) between pairwise IPVO [14] and the proposed approach on the Kodak set.

Test image	10,000 bits		20,000 bits		35,000 bits		Test image	10,000 bits		20,000 bits		35,000 bits	
	Pairwise IPVO	Proposed scheme	Pairwise IPVO	Proposed scheme	Pairwise IPVO	Proposed scheme		Pairwise IPVO	Proposed scheme	Pairwise IPVO	Proposed scheme	Pairwise IPVO	Proposed scheme
Kodim01	64.07	63.44	59.42	58.13	53.72	53	Kodim13	58.19	59.87	52.23	53.84	-	-
Kodim02	64.3	64.31	60.76	60.79	57.73	57.76	Kodim14	62.49	62.48	58.26	58.3	54.29	54.39
Kodim03	65.38	65.72	62.43	62.49	59.66	59.56	Kodim15	65.02	66.0	61.86	62.62	58.9	59.39
Kodim04	63.91	64.29	60.42	60.87	57.12	57.61	Kodim16	65.08	65.36	61.92	61.78	58.65	58.53
Kodim05	63.03	63.24	58.77	59.49	54.49	55.4	Kodim17	64.26	64.45	60.8	60.83	57.27	57.64
Kodim06	66.22	66.1	62.64	61.81	58.84	58.24	Kodim18	61.8	61.85	57.8	58.02	53.86	54.36
Kodim07	64.64	65.45	62.13	62.04	59.31	59.16	Kodim19	63.38	64.33	60.13	60.75	56.94	57.18
Kodim08	60.54	60.71	55.89	56.05	-	-	Kodim20	62.46	62.46	59.0	58.99	55.59	55.59
Kodim09	63.44	63.88	60.45	60.49	57.69	57.31	Kodim21	63.75	64.19	60.81	60.58	57.52	57.14
Kodim10	63.34	63.85	60.3	60.24	57.22	57.17	Kodim22	63.4	63.64	59.6	59.89	56.04	56.39
Kodim11	65.73	65.15	61.82	61.41	58.52	57.27	Kodim23	64.44	65.22	61.45	61.94	58.82	59.01
Kodim12	64.64	64.99	61.55	61.47	58.35	58.42	Kodim24	62.57	67.46	58.8	63.22	54.7	58.96
							Average	63.71	64.1	60.11	60.25	57.05	57.67

4. EXPERIMENTAL RESULTS

In this section, we evaluate the effectiveness of the proposed prediction-error-ordering reversible data hiding scheme. The results are presented for six classic test images and for the graylevel version of the Kodak set. These test images are shown in Fig. 2.

The performance of the proposed PEO approach is first compared with the results reported in [13], [14] and [15]. The corresponding plots are shown in Fig. 3. Based on these results, PEO outperforms [13] and [14] in terms of PSNR. While the quadtree-based PVO of [15] can generate better PSNRs at lower capacities, its results are erratic. Overall, PEO offers stable results that are superior for higher embedding capacities.

The PEO approach is also evaluated on the Kodak set and the results are shown in Table 1. The gain in PSNR of PEO over the pairwise IPVO scheme of [14] increases with the embedding capacity, reaching an average improvement in PSNR of 0.62 dB for an embedding capacity of 35,000 bits.

5. CONCLUSIONS

An original prediction-error-ordering based reversible data hiding scheme was proposed, derived from pixel-value-ordering. Besides the block based embedding, the proposed scheme also splits the image pixels into two sets. Each set is embedded separately, using the other set for prediction and local complexity evaluation. The proposed approach appears to outperform other state-of-the-art PVO based schemes.

6. REFERENCES

[1] Y. Q. Shi, X. Li, X. Zhang, H. T. Wu and K. Ma, "Reversible data hiding: advances in the past two decades", *IEEE Access*, 4, pp. 3210-3237, 2017.

[2] Z. Ni, Y. Q.Shi, N. Ansari and W. Su, "Reversible data hiding", *IEEE Transactions on circuits and systems for video technology*, 16 (3), pp. 354-362, 2006.

[3] W. Hong, T.-S. Chen, C.-W. Shiu, Reversible data hiding for high quality images using modification of prediction errors, *Journal of Systems and Software*, 82 (11), 1833-1842, 2009.

[4] V. Sachnev, H. J. Kim, J. Nam, S. Suresh, Y. Q. Shi, "Reversible Watermarking Algorithm Using Sorting and Prediction", *IEEE Trans. on Circuits and Systems for Video Technology*, 19 (7), pp. 989-999, 2009.

[5] I.-C Dragoi and D. Coltuc, "Local-Prediction-Based Difference Expansion Reversible Watermarking", *IEEE Trans. on Image Processing*, vol. 23, no. 4, pp. 1779–1790, 2014.

[6] X. Li, W. Zhang, X. Gui and B. Yang, "Efficient reversible data hiding based on multiple histograms modification", *IEEE Trans. on Information Forensics and Security*, vol. 10, no. 9, pp. 2016–2027, 2015.

[7] B. Ou, X. Li, Y. Zhao, R. Ni and Y.-Q. Shi, "Pairwise Prediction-Error Expansion for Efficient Reversible Data Hiding", *IEEE Trans. on Image Processing*, 22 (12), pp. 5010–5021, 2013.

[8] I.-C. Dragoi and D. Coltuc, "Adaptive pairing reversible watermarking," *IEEE Trans. Image Process.*, vol. 25, no. 5, pp. 2420–2422, 2016.

[9] X. Li, J. Li, B. Li and B. Yang, "High-fidelity reversible data hiding scheme based on pixel-value-ordering and prediction-error expansion", *Signal Process.*, 93 (1), pp. 198–205, 2013.

[10] F. Peng, X. Li and B. Yang, "Improved PVO-based reversible data hiding", *Digit. Signal Process.*, 25, pp. 255–265, 2014.

[11] B. Ou, X. Li and J. Wang, "Improved PVO-based reversible data hiding: A new implementation based on multiple histograms modification", *Journal of Visual Communication and Image Representation*, 38, pp. 328–339, 2016.

[12] P. Rahmani and G. Dastghaibyfard, "A Reversible Data Hiding Scheme Based on Prediction-Error Expansion Using Pixel-based Pixel Value Ordering Predictor", *Proc. of the 2017 Artificial Intelligence and Signal Processing (AISP2017)*, pp. 219-223, 2017.

[13] B. Ou, X. Li and J. Wang, "High-fidelity reversible data hiding based on pixel-value-ordering and pairwise prediction-error expansion", *Journal of Visual Communication and Image Representation*, 39, pp. 12–23, 2016.

[14] I. C. Dragoi, I. Caciula and D. Coltuc, Improved Pairwise Pixel-Value-Ordering for High-Fidelity Reversible Data Hiding, *25th IEEE International Conference on Image Processing (ICIP)*, pp. 1668-1672, 2018.

[15] F. Di, M. Zhang, X. Liao and J. Liu, High-fidelity reversible data hiding by Quadtree-based pixel value ordering, *Multimedia Tools and Applications*, pp. 1-17, 2018.