JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 37, 441-468 (2021)
DOI: 10.6688/JISE.202103_37(2).0011

Rectangle-Based Neo-Plasticism-Like Image – A New Type of Art Image and its Application to Covert Communication

SHAN-CHUN LIU[1], DA-CHUN WU[2,+] AND WEN-HSIANG TSAI[1,*]
[1]*Department of Computer Science*
National Chiao Tung University
Hsinchu, 300 Taiwan
[2]*Department of Computer and Communication Engineering*
National Kaohsiung University of Science and Technology
Kaohsiung, 824 Taiwan
E-mail: {azurecoral[1]; whtsai[*]}@gmail.com; dcwu@nkust.edu.tw[2]

A new type of art image, called rectangle-based Neo-Plasticism-like image, is proposed, via which messages can be hidden for covert communication. Also proposed is an automatic method for creating such art images, which applies recursive binary partitioning to a source image by finding the maximum mutual information of the spatial positions and image intensities of the divided sub-regions. The resulting image consists of rectangular regions separated by horizontal and vertical lines, which show the abstraction style of the Neo-Plasticism art. Attracted by the artistic image content, a hacker hopefully will pay no attention to the hidden secret. Two data hiding techniques based on the binary partition tree constructed in the art image creation process are proposed, which embed messages by replacing the LSBs of each rectangular region's colors or by generating additional partition lines in the region. A message extraction process is also proposed. Data security is considered seriously by randomizing the message bits before being embedded, changing randomly the priorities of the sub-regions used in message hiding, and embedding fake messages to interfere with the hacker. Good experimental results show the feasibility of the proposed techniques for covert communication via the proposed type of art image.

Keywords: computer art, data hiding, rectangle-based Neo-Plasticism-like image, binary partition tree, covert communication

1. INTRODUCTION

In recent years, automatic art image creation by computers becomes popular, and many methods have been proposed [1-9]. Hertzmann [2] gave a survey of methods for automatic creation of art images by stroke-based rendering using discrete elements like paint strokes and stipples. The common goal of these methods is to make the generated art image look like another type of image. For example, one type of art image created by watercolor painting by Hertzmann [3] is shown in Fig. 1, where Fig. 1 (a) is the original image and Fig. 1 (b) is generated by a computer program. Two examples of other types of computer art images are shown in Fig. 2, where Fig. 2 (a) was created by simulated pen-and-ink drawing proposed by Salisbury [5], and Fig. 2 (b) is a stained glass image created by an image filter presented in Mould [7].

In this study, we try to imitate another type of art, called Neo-Plasticism, to generate images with similar abstraction, called *rectangle-based Neo-Plasticism-like images*. As

Received September 4, 2019; revised March 8, 2020; accepted March 21, 2020.
Communicated by Jen-Hui Chuang.
[+] Corresponding author.
[*] Also with the Department of Information Communication at Asia University, Taichung, Taiwan.

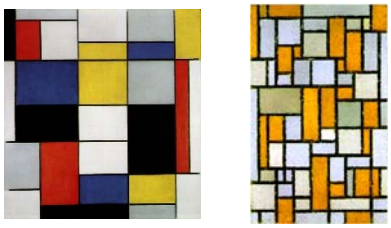
illustrations, Fig. 3 shows two examples of Neo-Plasticism art images, and Fig. 4 shows an example of the rectangle-based Neo-Plasticism-like image created by the art-image generation method proposed in this study.

(a) (b)

Fig. 1. Art images created by Hertzmann [3]; (a) An original image; (b) Created art image with a watercolor painting effect [3].

(a) (b)

Fig. 2. Two types of computer-generated art images; (a) A simulated pen-and-ink drawing generated by Salisbury [5]; (b) A simulated stain-glass image generated Mould [7].



(a) (b)

Fig. 3. Two examples of Neo-Plasticism art by Piet Mondrian; (a) Example 1 – "Composition A," 1920; (b) Example 2 – "Composition with Gray and Light Brown," 1918.

(a) (b)

Fig. 4. Rectangle-based Neo-Plasticism-like images generated in this study; (a) An original image; (b) A rectangle-based Neo-Plasticism-like images generated with (a) as the input.

Data hiding is a type of technique useful for covert communication, which embeds data imperceptibly into the *cover media* to prevent people from perceiving the existence of the data in the resulting *stego-media*. In this study, we deal with the art image generated by the proposed method, called rectangle-based Neo-Plasticism-like image, as the cover-media. Two criteria followed by existing data hiding methods are the *imperceptibility* of the distortion in the stego-image caused by data embedding as well as the *recoverability* of the hidden data from the stego-image. A weakness of the human visual system in differentiating small color differences in images is often utilized to achieve the goal of imperceptibility. In this aspect, the least significant bit (LSB) modification scheme by Chan and Cheng [10] and the contrast-keeping data embedding scheme by Wu and Tsai [11] are

two examples. To achieve recoverability which requires lossless data embedding, an approach is to compress part of the cover image and embed the compression result together with the payload into the cover image, like the methods proposed by Fridrich, *et al.* [12] and Awrangjeb and Kankanhalli [13]. Another approach is to manipulate a group of pixels as a unit to embed a bit of information, like Tian [14], Vleeschouwer, *et al.* [15], and Lee and Tsai [16]. A third approach is to apply the histogram shifting technique which can embed large volumes of data, *e.g.*, Ni, *et al.* [17] and Lee and Tsai [18].

In this study, image region partitioning is conducted to yield a binary partition tree, by which message data can be embedded in a key-controlled secure order into the Neo-Plasticism-like image created by the proposed art image generation method to achieve the goals of imperceptibility and recoverability. Each leaf of the binary partition tree corresponds to a created rectangular image region, into which message data are embedded by two techniques: replacements of the region colors' LSBs and coloring of additional partition lines applied to the region. Besides, by different numbers of partition iterations, Neo-Plasticism-like images with distinct abstraction levels can be created. Being attracted by the art exhibited by the created image, a hacker hopefully will pay no attention to the hidden message in the stego-image.

In the remainder of this paper, the proposed method for creating rectangle-based Neo-Plasticism-like images is presented in Section 2. The two proposed techniques for embedding a message into an input image and a method for extracting the message from a stego-image are described in Section 3, followed by experimental results in Section 4 which show feasibility of the proposed method and techniques. Some concluding remarks are included in Section 5.

2. CREATION OF RECTANGLE-BASED NEO-PLASTICISM-LIKE IMAGES

The Neo-Plasticism art aims to express an ideal idea of spiritual harmony and order. Artists of this school pursue pure abstraction by reducing a natural scene to the essentials of form and color, such as vertical and horizontal lines, three primary colors, and three non-primary colors. The idea of the art image creation method proposed in this study is inspired by this spirit of Neo-Plasticism. In the remainder of this section, the principle behind the proposed method is described in Section 2.1, and the details of the method are described as an algorithm in Section 2.2.

2.1 Principle Behind the Proposed Art Image Creation Method

Following the spirit of pure abstraction pursued by the Neo-Plasticism school, an input image is transformed automatically by the proposed method into an art image consisting of *rectangles of various sizes with uniform colors*, as illustrated by Fig. 5 (a). The created art image is therefore called rectangle-based Neo-Plasticism-like image.

2.1.1 Flowchart of the proposed art image creation process

A brief flowchart of the process of creating the proposed art image is shown in Fig. 5 (b). The input image is firstly partitioned recursively, each time using a horizontal or ver-

tical line, into multiple roughly-uniform rectangular regions. The position of each *partition line* is decided according to the *mutual information* [19] which is a measure about the intensities and spatial positions of the two sub-regions resulting from the *binary partitioning*. The information of the partition line and the two sub-regions are recorded as a node of a *binary partition tree*. Each rectangular region which is too small to be divided further is regarded as a leaf node of the tree. The binary partition tree is finally traversed to recolor every leaf-node region using the average color of all the pixels inside the region. In this way, a rectangle-based Neo-Plasticism-like image is created, which visually shows the spirit of harmony and order of the Neo-Plasticism art, like the example shown in Fig. 4. Furthermore, Neo-Plasticism-like images with different levels of abstraction can be created by performing different numbers of region-partitioning iterations during the art image creation process.

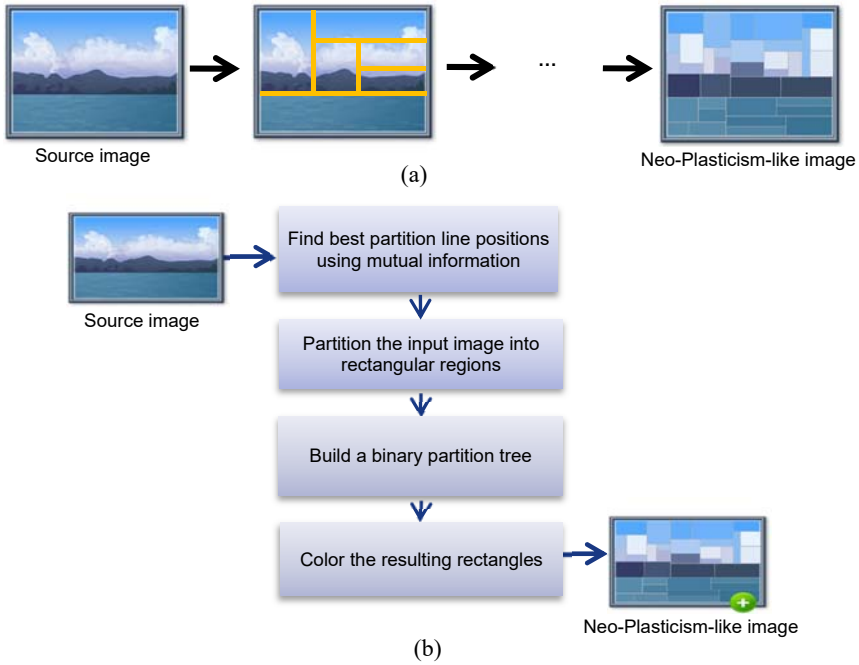


Fig. 5. Idea of proposed art image creation method; (a) Basic idea; (b) A flowchart of the proposed process of creating a rectangle-based Neo-Plasticism-like image.

2.1.2 Finding the best partition line by mutual information

In information theory, the mutual information of two random variables is a quantity that measures the mutual dependence between the two random variables. The larger this quantity, the higher the dependence. The mutual information value between two discrete random variables X and Y is defined as

$$MI(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right) \tag{1}$$

where $p(x, y)$ is the joint probability distribution function of X and Y, and $p(x)$ and $p(y)$ are the probability distribution functions of X and Y, respectively. Furthermore, since $p(x, y) = p(x|y)p(y)$, Eq. (1) can be reduced to be

$$MI(X;Y) = \sum_{y \in Y} p(y) \sum_{x \in X} p(x \mid y) \log\left(\frac{p(x \mid y)}{p(x)}\right). \tag{2}$$

In this study, we use the measure of mutual information between image intensities and spatial positions proposed by Rigau *et al.* [19] for region partitioning. Given an image with N pixels and an intensity histogram with n_i pixels in intensity bin x_i, let X represent the bins x_i of the histogram with probability distribution $p(x_i) = n_i/N$; and let Y represent the spatial regions y_j with probability distribution $p(y_j) = N_j/N$, where N_j is the area of region j in the unit of pixel. As shown in Fig. 6, assume that we partition a region by a vertical line into two sub-regions y_1 and y_2 with respective areas N_1 and N_2, and that the number of bins is 128. Then, according to Eq. (2), we can compute the value of the mutual information between X and Y by

$$MI(X;Y) = \sum_{j=1}^{2} p(y_j) \sum_{i=0}^{127} p(x_i \mid y_j) \log\left(\frac{p(x_i \mid y_j)}{p(x_i)}\right) \tag{3}$$

where $p(x_i|y_j) = n_i/N_j$ is the conditional probability distribution of bins x_i in the sub-region y_j. Accordingly, we can apply Eq. (3) to calculate the mutual information for *each* possible partition line (vertical or horizontal) which divides a region into two sub-regions, and find the *maximum* mutual information, which means the maximum information quantity between the spatial positions and intensities of the two sub-regions, to obtain an appropriate partition-line position to divide the region.

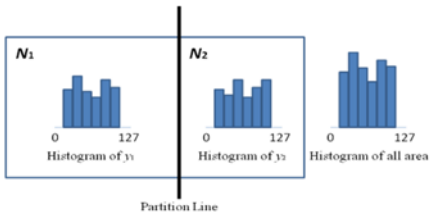


Fig. 6. An example of partitioning a region into two sub-regions with N_1 and N_2 pixels, respectively.

2.1.3 Region partitioning strategy

In more detail, in order to compute the measure of mutual information, we have to compute the probabilities of all the bins of the intensity histogram for each partitioned region; therefore, the computation load is large. To reduce the computation time, we transform the 3-D RGB color of each image pixel described by (r, g, b) into a 1-D color denoted by $h(r, g, b)$ according to the following equation:

$$h(r, g, b) = b + 8 \times r + 64 \times g, \tag{4}$$

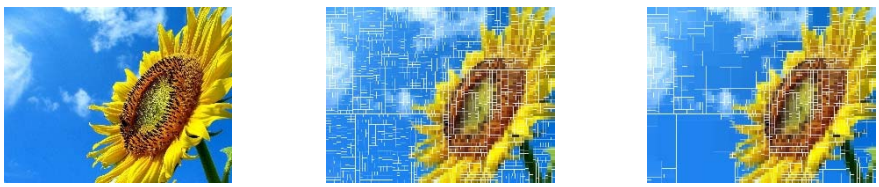
and quantize this 1-D color to limit the number of bins to be 128 as mentioned previously. We then use the resulting 1-D color as the intensity measure of the pixels in the image for computing the mutual information measure. Finally, we speed up the computation of the mutual information measure further by the following two ways: (i) trimming the first 10 pixels and the last 10 of each of the four boundaries of the region because the partition line will normally be found not to be near the two ends of any boundary; (ii) rescaling the image down to 1/9 before computing the mutual information.

Though we use the mutual information to divide an image, yet if a region in the image is almost smooth everywhere so that the colors of the pixels in the region are quite similar, then it is obviously unnecessary to partition the region. Therefore, we need a criterion to determine whether to partition a region or not. For this, based on the available information — the intensity histogram of the region, we use the following equation to compute a *similarity measure E* of the partitioning result with respect to the original region:

$$E = \sum_{i=0}^{127} (|H_1(i) - H_2(i)| / N) \times (B_n / B) \tag{5}$$

where N is the number of pixels in the original region, say, denoted as R, H_1 is the intensity histogram of a sub-region of R, H_2 is that of the other sub-region, B is the number of bins of the intensity histogram of R, and B_n is the number of bins with non-zero values. If the two sub-regions are quite different in their intensity histograms and the value of each bin is not zero, the value of this similarity measure E will approach 1. In this study, we use the small value of 0.05 as the lower bound of the similarity measure E for region partitioning. Accordingly, a region with smooth colors, whose value of E is not larger than 0.05, will not be partitioned further. An example illustrating the effect of using the similarity measure E is shown in Fig. 7.

In short, by partitioning a given image recursively based on the mutual information MI and the similarity measure E, a rectangle-based Neo-Plasticism image can be created, achieving the implementation of the concept of the Neo-Plasticism school which aims at transforming images into their abstract forms.



(a) (b) (c)

Fig. 7. Effects of the use of the similarity measure E; (a) A source image; (b) Result of using (a) as the input image through 30 partition iterations; (c) Result of using (a) as the input image through 30 partition iterations with 0.05 as the lower bound of the similarity measure E, yielding some larger uniform regions.

2.2 Proposed Art Image Creation Process

The details of the process described previously to create a rectangle-based Neo-Plasticism-like image from a given image is described as an algorithm in the following. A

threshold I for the number of partition iterations is used in the algorithm to control how far the source image S is to be partitioned: the use of a larger value of I will result in dividing S into smaller-sized rectangles.

Algorithm 1: Creation of a rectangle-based Neo-Plasticism-like image.

Input: A source image S, and a threshold I for the number of partition iterations.
Output: A rectangle-based Neo-Plasticism-like image O.
Steps.

Stage 1: Initialization.

Step 1. Transform the 3-D RGB color of each pixel in the input image S into a 1-D h-color according to Eq. (4) and quantize all the h-color values into the range of 0 to 127.

Step 2. Set up a stack A, initially empty.

Step 3. Regard image S as the root node of a binary partition tree T being empty initially, associate S with a *partition level* initially set equal to the input threshold I for the number of partition iterations, and push S into stack A as the initial region for partitioning.

Stage 2: Constructing a binary partition tree via the use of a stack.

Step 4. Pop up the top node R of stack A and its associated partition level I_R, and if the size of R is smaller than 20×20, then go to Step 6; else, create the probability density function $p(x_i)$ of the h-color histogram of R.

Step 5. Perform the following steps to partition region R into two sub-regions, where the region boundaries of R are described by its two diagonal corners at coordinates (X_{min}, Y_{min}) and (X_{max}, Y_{max}), respectively.

5.1 Find the best horizontal partition line in the following way in the horizontal direction from $Y_{min} + 10$ to $Y_{max} - 10$ according to the mutual information measure.

(a) For $i = 0, 3, 6, \ldots, n$ with $n = Y_{max} - Y_{min} - 20$, regard $Y_{min} + 10 + i$ as a horizontal partition line l_{hi} to divide region R into two sub-regions with areas y_1 and y_2, respectively, and compute the following data: (i) the marginal probabilities $p(y_1)$ and $p(y_2)$; (ii) the conditional probabilities $p(x_i|y_1)$ and $p(x_i|y_2)$; and (iii) the horizontal mutual information value MI_{hi} according to Eq. (3).

(b) Select the maximum mutual information value from $MI_{h1}, MI_{h2}, \ldots, MI_{hn}$, denote the result as MI_{hmax}, and denote the corresponding partition line as l_{hmax}.

(c) Apply partition line l_{hmax} to region R to obtain two sub-regions with h-color histograms H_1 and H_2, respectively, and compute accordingly the similarity value E_{hhmax} according to Eq. (5).

5.2 Perform operations similar to those of Step 5.1 to region R in the vertical direction from $X_{min} + 10$ to $X_{max} - 10$ to obtain the maximum mutual information denoted as MI_{vmax}, the corresponding vertical partition line denoted as l_{vmax}, and the similarity value denoted as E_{vmax} resulting from applying l_{vmax} to R.

5.3 Compare two maximum mutual information values MI_{hmax} and MI_{vmax} of the horizontal and vertical directions, respectively, choose the larger one denoted as MI_{max}, and apply the partition line l_{max} corresponding to MI_{max} to R, resulting

in the similarity value E_{max}.

5.4 If E_{max} is not larger than 0.05, then regard R as a leaf node of the binary partition tree T and conduct no further partitioning; else, regard R as an intermediate node of tree T and perform the following steps.

(a) Denote the two sub-regions resulting from applying the partition line l_{max} to R as R_1 and R_2.

(b) Associate the data of l_{max}, R_1, and R_2 with the tree node of R.

(c) Decrease the partition level I_R of R by 1, *i.e.*, set $I_R = I_R - 1$.

(d) If $I_R \neq 0$, then push R_1 and R_2 into stack A in order, each being associated with the new partition level I_R.

Step 6. Repeat Steps 4 and 5 until stack A is empty.

Stage 3: Recoloring the partitioned regions.

Step 7. Traverse the binary partition tree T in a depth-first manner, and perform the following two steps.

7.1 Color the partition line associated with each intermediate node of tree T by the white color.

7.2 Compute the average RGB color C_{avg} of the region R_{leaf} of each leaf node of T, and replace the color of every pixel in R_{leaf} by C_{avg}.

Step 8. Take the final image S as the desired art image O.

3. DATA EMBEDDING AND EXTRACTION VIA ART IMAGES

In this section, the ideas of the proposed data embedding techniques are described in Section 3.1. Then, the detailed algorithms of the proposed techniques are presented in Section 3.2. In Section 3.3, an algorithm describing the proposed data extraction process is given. Finally, in Section 3.4, some experimental results of applying the algorithms are shown, which indicate the effectiveness of the proposed techniques.

3.1 Ideas of Proposed Data Embedding Techniques

In this study, we propose to use the rectangle-based Neo-Plasticism-like image for secret message embedding. Using the art image for this purpose has the merit of attracting a hacker to concentrate on appreciating the abstract art content, reducing the potential chance of his/her attack on the image. Several issues involved in the design of a data embedding process to achieve this goal are discussed in the following.

3.1.1 Random partition directions versus alternative partition directions

In the proposed Neo-Plasticism-like image creation process described by Algorithm 1, a given image is partitioned recursively by finding the maximum mutual information to decide the position of the partition line for each partitioned region. After several iterations to get a result like that shown in Fig. 8 (a), a binary partition tree like the one shown in Fig. 8 (b) is built, which records the partitioning process with each leaf node of the tree corresponding to a final rectangular region. Without constraining *the order of the partition directions* (horizontal or vertical), consecutive partitioning operations in identical directions may occur, yielding a less balanced partition result, like Fig. 8 (a) where two consecutive

vertical partitions were applied at the beginning, yielding the leftmost *long* strip of region A which crosses the entire image.

To remedy this visual unbalance, we may constrain the partition directions at different tree node levels to be *in an alternative order* (*i.e.*, horizontal and then vertical repetitively, or reversely) in the message embedding process, resulting in a tree like that shown in Fig. 8 (d) with the corresponding partition result shown in Fig. 8 (c) which has no long vertical or horizontal strip crossing the entire image now. Another merit of adopting such a partition-direction constraining strategy is that an identical binary partition tree can be reconstructed at the beginning of the message extraction process with less image analysis efforts because a trial-and-error check of the partition direction (horizontal or vertical) at each tree node can be avoided. This partition-direction constraining strategy is adopted in one of the two message embedding techniques proposed in this study, which are to be described subsequently.

Fig. 8. Effects of two ways of building a binary partition tree; (a) A multiply partitioned image with the red line as the first partition line and a long strip A appearing at the left-hand side; (b) A binary partition tree built according to (a); (c) A multiply partitioned image with a fixed alternative order of horizontal and vertical partition directions, which is different from that of (a); (d) A binary partition tree built according to (c).

3.1.2 Two techniques for data embedding: LSB replacement and partition-line coloring

Two techniques are proposed in this study to hide secret messages into those image regions which are not partitioned further in the art image creation process, called *intact regions* henceforth. The first, called Technique 1, is proposed to embed message bits into the LSBs of the RGB color values of the intact regions which are generated in an alternative order of partition directions mentioned previously. The hidden message is invisible because the human vision cannot tell the color difference caused by the changes of the LSBs of the color values. This technique is useful when the number of partition iterations is large so that a large number of intact regions are generated.

However, it is possible that the input image has many large smooth sub-regions so that image partitioning can be applied only for a limited number of partition iterations. As a result, only a limited number of intact regions can be generated, and so only a few message bits can be embedded into the regions by Technique 1. In such a case, the second

technique, called Technique 2, is proposed for use, which embeds the message into the intact regions in another way consisting of two major steps: (i) partitioning each intact region further using additional horizontal and vertical partition lines, and (ii) coloring these lines with two nearly identical but differentiable colors to encode the binary message bits of 0 and 1, respectively. The colors of these partition lines in each intact region are chosen to be extremely close to the average color of the intact region's pixels so that the partition lines are invisible to an observer, just like the invisibility effect yielded by the LSB replacements of Technique 1.

An illustrative example of results yielded by Technique 2 is shown in Fig. 9, where the blue intact region labeled as D is used for message bit embedding. Specifically, a secret bit of 0 is embedded by drawing a horizontal partition line going through the *central* pixel of D or a sub-region of it, and a bit of 1 is embedded by drawing a vertical partition line in a similar way. The multiple partition lines in region D together show the result of embedding the message bits of 010010 where the blue background color is shown for *highlighting* the pink partition lines; in the real case the background color should be the average color of region D, which supposedly is also nearly pink (but different from the mutually differentiable pink colors of the horizontal and vertical partition lines chosen according to the average color of region D as mentioned previously). It is noted that no constraint on the partition direction order like the aforementioned alternativeness used in Technique 1 is enforced because the partition lines are invisible, and so no visual balance of the rectangular intact regions is necessary.

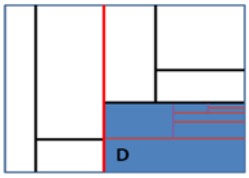


Fig. 9. An illustration of partition-line coloring used by the second proposed technique to embed message bits.

3.1.3 Marking the ending of secret-bit embedding

Both techniques encounter the difficulty of *marking the ending of secret-bit embedding* in the stego-image during the data embedding process. The use of a kind of *ending mark* is necessary, which must be easily detectable during the message extraction process. For Technique 1, a solution proposed in this study to solve this problem is to recolor the central pixel of the lastly-processed region to make its color different from those of the other pixels in the region. And for Technique 2, a solution proposed is to color an extra vertical partition line going through the central pixel of a region differently from the way the other vertical and horizontal partition lines in other regions are colored.

3.1.4 Security considerations

To increase the security of the embedded message data, in addition to generating an artistic stego-image to distract the hacker's notice on the embedded secret message, we use further three schemes all based on the use of *a secret key* to reduce the possibility for the

hacker to 'crack' the secret message effectively or to prevent him/her from guessing the hidden message correctly: (i) randomizing the secret message bits – using the secret key to randomize the order of the message bits before they are embedded into the cover image; (ii) randomizing the priority of the partition order – using the key to decide randomly, for the horizontal partition case, to hide a secret bit into the upper or the lower sub-region first; or for the vertical partition case, into the left or the right sub-region first; (iii) embedding of a fake message – using the secret key to generate a random-bit sequence as a fake message and imitating the regular process to embed the fake bit sequence into unprocessed intact regions (carried out by the second proposed technique for data embedding). With these multiple schemes for security enhancement, we can strongly protect the secret message from being stolen by a malicious user.

3.2 Algorithms of Proposed Data Embedding Techniques

In this section, the two proposed techniques for embedding messages in rectangle-based Neo-Plasticism-like images are described in detail as two algorithms, respectively. The first algorithm using Technique 1 for data embedding is based mainly on applying the LSB replacement scheme to the colors of the intact regions; and the second algorithm using Technique 2 is based on a scheme proposed in this study, which draws extra colored partition lines within the intact regions to encode message bits.

3.2.1 Data embedding by Technique 1

A flowchart of the proposed data embedding process using Technique 1 is shown in Fig. 10. Firstly, the input secret message is transformed into a random digit sequence using a secret key. Also, Algorithm 1 is applied to divide the input source image recursively into intact regions using partition lines of alternative directions (horizontal and then vertical, or reversely). A binary partition tree is built in the meantime. Then, the tree is traversed to construct a data-hiding sequence of the generated intact regions in a random order, which is followed finally to embed the transformed message now in the form of a random digit

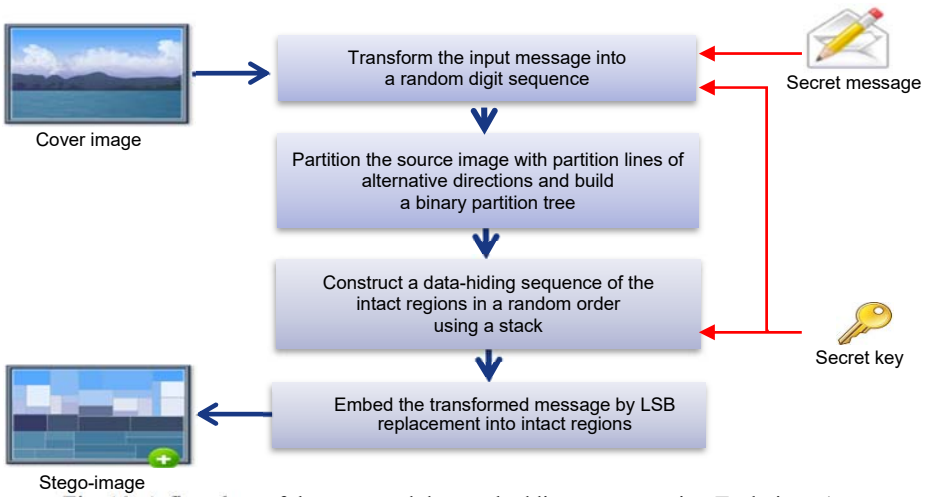


Fig. 10. A flowchart of the proposed data embedding process using Technique 1.

sequence into the intact regions by LSB (least significant bit) replacement. A stego-image is so generated which can be used for covert communication. A detailed algorithm of this data embedding process using Technique 1 is presented in the following.

Algorithm 2: Embedding a message into a Neo-Plasticism-like image by Technique 1.

Input: A source image S, a secret message M, a secret key K, and a threshold I for the number of partition iterations.

Output: A stego-image O into which M is embedded.

Steps.

Stage 1: Transforming the secret message into a random sequence of 3-digit segments.

Step 1. Transform the input massage M of characters into a bit sequence with eight bits for each character, randomize the bit positions in the sequence by the secret key K, and denote the result by M'.

Step 2. Regard each bit of M' as a *digit*, append an *ending signal* consisting of at least one and no more than three identical digits other than 0's and 1's (such as 2, 22, or 222) to the end of M' to form a 3-bit *ending pattern* (such as 102, 022, or 222), and denote the resulting digit sequence with its length being a multiple of three by M'' (*e.g.*, with the resulting digit sequence being 010 101 022).

Step 3. Divide M'' into a series of 3-digit segments $m_1, m_2, \ldots, m_N$.

Stage 2: Partitioning the source image with color lines and building a binary partition tree.

Step 4. Perform Steps 4, 5 and 6 of Algorithm 1, using the input image S as the input region and constraining the partition directions (horizontal or vertical) to be in an alternative order, to obtain a binary partition tree T.

Step 5. While traversing the tree T in a depth-first manner, at each tree node N perform the following steps.

- 5.1 Use the secret key K to determine randomly a distinct *priority of the partition direction* (*i.e.*, to determine, for the horizontal partition case, to process the upper or the lower sub-region first; or to determine for the vertical partition case, to process the left or the right sub-region first).
- 5.2 Associate the determined priority with node N.
- 5.3 Take out the partition line l_N associated with node N, and recolor l_N by the RGB color (255, 255, 254) if l_N is a horizontal partition line; or by the color (255, 255, 255) if l_N is a vertical one.

Stage 3: Constructing a data-hiding sequence of the intact regions using a stack.

Step 6. Set up a stack A in the following way to construct a *data-hiding sequence* H which consists of the intact regions corresponding to the leaf nodes of the binary partition tree T and is to be used as an ordering set for hiding the message bits later in the next stage.

- 6.1 Push the root node of tree T into stack A.
- 6.2 Pop up the top node N in stack A, and if note N is a leaf node, then put N into H in order; else, push the two child nodes of node N into A in an order decided by the priority of the partition direction associated with node N (*i.e.*, push in later the child node whose corresponding sub-region has the higher priority).
- 6.3 Repeat the previous step until stack A is empty.

Stage 4: Embedding the transformed message by LSB replacement into the intact regions.

Step 7. Take in order a 3-digit segment m_t from the digit sequence M'', and let the three digits of m_t be denoted by (b_{1t}, b_{2t}, b_{3t}).

Step 8. Take in order a region R_i from the data-hiding sequence H, and take out the RGB values (C_{ri}, C_{gi}, C_{bi}) of the color of any pixel in R_i (which is the previously-computed average color of R_i).

Step 9. Embed m_t into R_i by performing one of the following two steps.

9.1 (*Conducting LSB replacement when necessary*) Adjust the LSBs of the RGB values (C_{ri}, C_{gi}, C_{bi}) of the color of every pixel in R_i according to the values of the three digits (b_{1t}, b_{2t}, b_{3t}) in the following way.

(a) Set the LSB of C_{ri} to be 0 if $b_{1t} = 0$; or to be 1 if $b_{1t} = 1$; or keep C_{ri} unchanged, otherwise.

(b) Set the LSB of C_{gi} to be 0 if $b_{2t} = 0$; or to be 1 if $b_{2t} = 1$; or keep C_{gi} unchanged, otherwise.

(c) Set the LSB of C_{bi} to be 0 if $b_{3t} = 0$; or to be 1 if $b_{3t} = 1$; or keep C_{bi} unchanged, otherwise.

9.2 (*Marking the ending of data embedding*) If the segment m_t *is* an ending pattern (such as 102, 022, or 222 as mentioned previously), then recolor by one of the following ways the *central pixel* P_m of region R_i to indicate the ending signal existing in m_t, assuming that the original RGB color values of P_m are denoted as (C_{rm}, C_{gm}, C_{bm}).

(a) If only the last digit b_{3t} of m_t is other than 0 or 1 (such as being 2), then flip the *second* LSB of the B color value C_{bm} of P_m.

(b) If both of the last two digits b_{2t} and b_{3t} of m_t are other than 0's and 1's (such as being 22), then flip the *second* LSBs of both the G and B color values C_{gm} and C_{bm} of P_m.

(c) If the three digits of m_t are all other than 0's and 1's (such as being 222), then flip the *second* LSBs of all the RGB values C_{rm}, C_{gm}, and C_{bm} of P_m.

Step 10. Repeat Steps 7, 8 and 9 until the digit sequence M'' is exhausted.

Step 11. Take the final S as the desired stego-image O.

3.2.2 Data embedding by Technique 2

A flowchart of the proposed data embedding process using Technique 2 is shown in Fig. 11. First of all, the input secret message is transformed into a random bit sequence using a secret key. Also, Algorithm 1 is applied to divide the input source image recursively into intact regions using partition lines of no alternation in directions (differently from Technique 1). A binary partition tree is built in the meantime. Then, the tree is traversed to construct a data-hiding sequence of the generated intact regions in a random order, which is finally followed to embed the transformed message now in the form of a random bit sequence into the intact regions by horizontal and vertical partition lines with two nearly identical but differentiable colors. The colors of the partition lines in an intact region are almost identical to the color of the region; therefore, the partition lines are invisible to an observer. A stego-image is so generated which can be used for covert communication. A detailed algorithm of this data embedding process using Technique 2 is presented next.

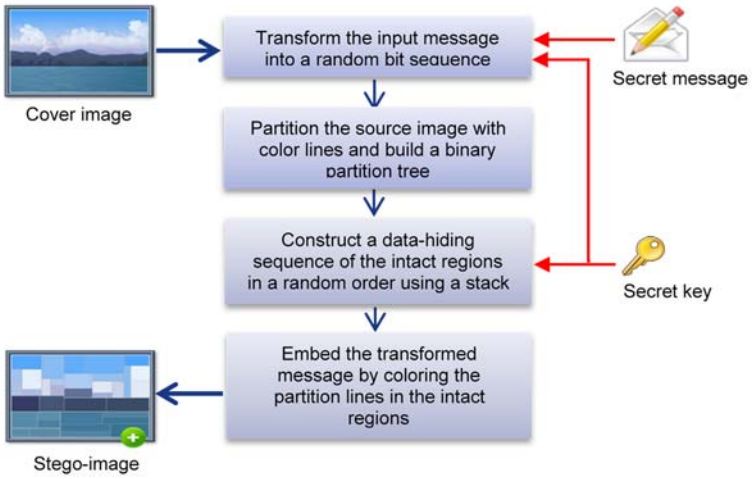


Fig. 11. A flowchart of the proposed data embedding process using Technique 2.

Algorithm 3: Embedding a message into a Neo-Plasticism-like image by Technique 2.

Input: A source image S, a secret message M, a secret key K, and a threshold I for the number of partition iterations.

Output: A stego-image O into which M is embedded.

Steps.

Stage 1: Transforming the secret message into a random bit sequence.

Step 1. Transform the input massage M of characters into a bit sequence with eight bits for each character, randomize the bit positions in the sequence by the secret key K, and denote the result by M'.

Stage 2: Partitioning the source image with color lines and building a binary partition tree.

Step 2. Perform Steps 4, 5, and 6 of Algorithm 1, using the input image S as the input region R, to obtain a binary partition tree T.

Step 3. While traversing the tree T in a depth-first manner, at each tree node N perform the following steps.

3.1 Use the secret key K to determine randomly a distinct *priority of the partition direction* (*i.e.*, to determine, for the horizontal partition case, to process the upper or the lower sub-region first; or to determine for the vertical partition case, to process the left or the right sub-region first).

3.2 Associate the determined priority with node N.

3.3 Take out the partition line l_N associated with node N, and recolor l_N by the RGB color of (255, 255, 255) if l_N is a horizontal partition line; or by the color (255, 255, 254) if l_N is a vertical one (note that the partition line colors used here are just the reverse of those used in Algorithm 2).

Stage 3: Constructing a data-hiding sequence of the intact regions using a stack.

Step 4. Set up a stack A in the following way to construct a *data-hiding sequence* H which consists of the intact regions corresponding to the leaf nodes of the binary partition tree T and is to be used as an ordering set for hiding the message bits later in the

next stage.

4.1 Push the root node of tree T into stack A.

4.2 Pop up the top node N in stack A, and if note N is a leaf node, then put it into H in order; else, push the two child nodes of node N into A in an order decided by the priority of the partition direction associated with node N (*i.e.*, push in later the child node whose corresponding sub-region has the higher priority).

4.3 Repeat the previous step until stack A is empty.

Stage 4: Embedding the transformed message by coloring the partition lines in the intact regions.

Step 5. Take in order a region R_i from the data-hiding sequence H, and take out the RGB values (C_{ri}, C_{gi}, C_{bi}) of the color of any pixel in R_i.

Step 6. Embed the remaining part of bit sequence M' into region R_i by the following steps.

6.1 Set up a stack A_i for R_i.

6.2 Partition region R_i in the following way using horizontal and vertical partition lines with two distinct colors representing message bits 0 and 1, respectively.

(a) Push R_i into stack A_i.

(b) Pop up the top region in stack A_i and denote it as R.

(c) If the area of region R is not larger than 2×2, then go to Step 6.2(h).

(d) Take a message bit b in order from the remaining part of M'.

(e) Perform one of the following two operations.

(i) If bit b is 0, then find the horizontal partition line going through the central pixel of R and color it by the RGB color (C_{ri} + 1, C_{gi}, C_{bi}).

(ii) If bit b is 1, then find the vertical partition line going through the central pixel of R and color it by the RGB color (C_{ri}, C_{gi}, C_{bi} + 1).

(f) Use the secret key K to determine randomly a priority of the partition direction for the two sub-regions of region R created in Step (e) above using the colored partition line (*i.e.*, to determine, for the horizontal partition case, to process the upper or the lower sub-region first; or to determine, for the vertical partition case, to process the left or the right sub-region first).

(g) Push each of the two sub-regions into stack A_i according to the determined priority (*i.e.*, push in later the sub-region which has the higher priority).

(h) Repeat Steps (b) through (g) until the bit sequence of M' is not exhausted or until stack A_i is empty.

6.3 Color the remaining pixels in region R_i (*i.e.*, color the pixels other than those falling on the partition lines generated in Step (e) above) by the average RGB color (C_{ri}, C_{gi}, C_{bi}) of R_i.

Step 7. Repeat Step 6 until the bit sequence of M' is exhausted.

Step 8. (*Marking the ending of data embedding*) Perform the following steps to mark the ending of data embedding.

8.1 Take in order from the remaining regions in H a region R whose size is larger than 2×2.

8.2 Color the vertical line going through the central pixel of R by the RGB color (C_{ri}, C_{gi} + 1, C_{bi}) as an ending signal, and call the line as an *ending-signal line*.

Step 9. (*Embedding fake message bits*) If there are any remaining intact regions in the data-embedding sequence H, then perform the following two steps.

9.1 Use the secret key K to generate a long random-bit sequence as a *fake message* M_f.
9.2 Perform steps similar to those in Step 6 to embed M_f into the remaining intact regions in order until the regions are exhausted.
Step 10. Take the final image S as the desired stego-image O.

Note that in Algorithm 2, we follow an alternative order of horizontal and vertical partition directions to generate an art image (see Step 2), while in Algorithm 3, we do not. Also note that the color (255, 255, 254) and (255, 255, 255) are used to color the horizontal and vertical lines, respectively, or reversely in the previous two algorithms; therefore, if the average color C of any pixel in an intact region is (255, 255, 254) or (255, 255, 255), then we replace C by the color (254, 254, 252) or (254, 254, 253), respectively, before C is used to be the color of every pixel in the region.

Also, in Algorithm 3, since each of the RGB values (C_{ri}, C_{gi}, C_{bi}) of the average color of the region R_i mentioned in Step 5 might be increased by 1 in Step 6.2 (e) and Step 8.2, to avoid an overflow that will occur when a color component value is 255, we replace it by 254 before it is used as a component of the color of every pixel in the region R_i.

3.3 Proposed Message Extraction Process

In the proposed secret message extraction process, which of the two techniques, Technique 1 or Technique 2, is used in the secret message embedding process is decided firstly. This is carried out by checking the color of the partition line, either horizontal or vertical, which crosses the entire image. Then, by reconstructing the binary partition tree and applying the reverse randomization using the secret key, the information of the data-hiding sequence mentioned in Algorithms 2 and 3 is acquired. Accordingly, the secret message embedded in the stego-image is extracted by two ways corresponding to the reverse versions of the two techniques applied in the data embedding process. An algorithm describing the details of this message extraction process is presented next.

Algorithm 4: Extracting the hidden secret message from a stego-image.
Input: A stego-image S, and a secret key K identical to that used in Algorithms 2 and 3.
Output: The secret message M embedded in S.
Steps:
Stage 1: Determining the used data hiding technique and retrieving the binary partition tree.
Step 1. Scan in the stego-image S any partition line that crosses the entire image in either the horizontal or the vertical direction (*i.e.*, that goes either from the left image boundary to the right, or from the top image boundary to the bottom), and decide the technique used in creating the stego-image S according to one of the following two cases:
(a) if a horizontal partition line of the mentioned property is detected with the color (255, 255, 254), then decide the message embedding process to be based on Technique 1 and go to Step 2; else, to be on Technique 2 and go to Step 4;
(b) if a vertical partition line of the mentioned property is detected with the color (255, 255, 255), then decide the message embedding process to be based on

Technique 1 and go to Step 2; else, to be on Technique 2 and go to Step 4.

Stage 2: Reconstructing the data-hiding sequence while retrieving the binary partition tree.

Step 2. (*For the case of adopting Technique 1 in data embedding*) Reconstruct in the following way the data-hiding sequence H, empty initially, while retrieving the binary partition tree T from the stego-image S.

2.1 Set up a stack A, let the stego-image S as the first input region, and push it onto the top of stack A as the root of the tree T being empty initially.

2.2 Pop up the top region R in stack A, regard it as a node of tree T, and perform the following steps.

(a) In an alternative order, find in region R a partition line l_R, either horizontal of the color (255, 255, 254) or vertical of the color (255, 255, 255), which crosses the entire range of R.

(b) If a partition line l_R of the mentioned property is found to exist, then perform the following steps:

(i) take out the sub-regions R_1 and R_2 at the two sides of l_R in region R as two child nodes of node R, respectively;

(ii) use the secret key K to decide the priority of the partition order of R_1 and R_2; and

(iii) push R_1 and R_2 into stack A according to the decided priority (the child node with the higher priority is pushed in later);

else, perform the following steps:

(i) regard region R as a leaf node of tree T; and

(ii) put R together with the partition line l_R into the data-hiding sequence H in order.

2.3 Repeat the previous step, Step 2.2, until no more region in stack A can be popped out.

Stage 3a: Extracting the embedded message for the case of using Technique 1 in data embedding.

Step 3. Extract the hidden secret message M from the stego-image S according to the data-hiding sequence H in the following way.

3.1 Set up a bit sequence Q, empty initially.

3.2 Take out in order an unprocessed region R_i from sequence H.

3.3 Extract three bits Q_{ri}, Q_{gi}, and Q_{bi} respectively from the LSBs of the R, G, and B values of the color of any pixel in region R_i except the central pixel P_m of R_i.

3.4 If the B value of the color of the central pixel P_m in R_i is *not* different from Q_{bi} which is the average B color value of R_i, then perform the following two steps to extract three message bits:

(a) take Q_{ri}, Q_{gi}, and Q_{bi} as three desired message bits and put them into Q in order; and

(b) repeat Steps 3.2 through 3.4;

else, perform the following four steps to extract the last one or two message bits:

(a) extract the *second* LSBs Q_{ri}', Q_{gi}', and Q_{bi}' of the R, G, and B values of the color of any pixel P_i of region R_i, respectively, except the central

pixel P_m of R_i;
(b) extract the *second* LSBs Q_{rm}', Q_{gm}', and Q_{bm}' of the R, G, and B values of the color of the central pixel P_m of R_i, respectively;
(c) perform one of the following three cases:
(i) if only Q_{bi}' is different from Q_{bm}', then extract two more message bits Q_{ri} and Q_{gi} and put them into Q in order;
(ii) if both Q_{gi}' and Q_{bi}' are different from Q_{gm}' and Q_{bm}', respectively, then extract one more message bit Q_{ri} and put it into Q in order;
(iii) if Q_{ri}', Q_{gi}', and Q_{bi}' are all different from Q_{rm}', Q_{gm}', and Q_{bm}', respectively, then no more message bit is extracted; and
(d) go to Step 6.

Stage 3b: Extracting the embedded message for the case of using Technique 2 in data embedding.

Step 4. Reconstruct the data-hiding sequence H, empty initially, while retrieving the binary partition tree T from the stego-image S in a way similar to Step 2 above except that Step 2.2(a) is modified to be as follows (with the requirements of finding in the alternative order removed and the colors of the searched partition line changed):
Step 2.2 (a) Find in region R a partition line l_R, either horizontal of the color (255, 255, 255) or vertical of the color (255, 255, 254), which crosses the entire range of R.

Step 5. Extract the hidden secret message M from the stego-image S according to the data-hiding sequence H in the following way.
5.1 Set up a bit sequence Q, initially empty.
5.2 Take out in order an unprocessed region R_i with the associated partition line l_i from sequence H.
5.3 Find the first partition line l_0, horizontal or vertical, which goes through the central pixel P_m of region R_i, crosses the entire range of R_i, and has pixels all of a single color C_0.
5.4 Obtain the colors C_v and C_h of the vertical and horizontal partition lines, respectively, as well as the color C_e of the possibly-existing ending-signal line in region R_i by one of the following two ways:
(a) if the firstly-found partition line l_0 in region R_i is horizontal with color $C_0 = (C_{r0}, C_{g0}, C_{b0})$, then take the color C_h of all the horizontal partition lines in R_i to be $C_h = C_0 = (C_{r0}, C_{g0}, C_{b0})$, the color C_v of all the vertical partition lines to be $C_v = (C_{r0} - 1, C_{g0}, C_{b0} + 1)$, and the color C_e of the ending-signal line to be $C_e = (C_{r0} - 1, C_{g0} + 1, C_{b0})$;
(b) if the firstly-found partition line l_0 in region R_i is vertical with color $C_0 = (C_{r0}, C_{g0}, C_{b0})$, then take the color C_v of all the vertical partition lines in R_i to be $C_v = C_0 = (C_{r0}, C_{g0}, C_{b0})$, the color C_h of all the horizontal partition lines to be $C_h = (C_{r0} + 1, C_{g0}, C_{b0} - 1)$, and the color C_e of the ending-signal line to be $C_e = (C_{r0}, C_{g0} + 1, C_{b0} - 1)$.
5.5 Set up a stack A for use to extract the message bits embedded in R_i in the following way.
(a) Push R_i associated with the first partition line l_0 into stack A.
(b) Pop up the top region R and the associated partition line l_R in stack A.

(c) Check the color C_R of partition line l_R: if $C_R = C_e$, then go to Step 6.
(d) Extract a message bit of 0 if $C_R = C_h$; or a bit of 1 if $C_R = C_v$; and put the extracted bit in order into bit sequence Q.
(e) For each sub-region R_s of region R at a side of partition line l_R, if the area of R_s is larger than 2×2, then use the colors C_h, C_v and C_e to recognize the type of the partition line l_s going through the central pixel of R_s according to one of the following four cases:
  (i) if the color of l_s is C_h, then it is a horizontal partition line;
  (ii) if the color of l_s is C_v, then it is a vertical partition line;
  (iii) if the color of l_s is C_e, then it is an ending-signal line;
  (iv) if the color of l_s is not any of C_h, C_v and C_e, then decide that no partition line or ending-signal line exists in R_s.
(f) Use the secret key K to determine the priority of the partition order of the two sub-regions of region R.
(g) For each sub-region R_s of R, if its area is larger than 2×2 and a partition or ending-signal line of the four cases mentioned in (e) was found in R_s, then push R_s together with the found partition line into stack A according to the determined priority of the two sub-regions (the one with the higher priority is pushed in later).
(h) Repeat Steps (b) through (g).

Stage 4: Post-processing the extracted message bits to be the desired secret message.

Step 6. Use the secret key K to reorder the extracted bit sequence Q.
Step 7. Transform every eight bits of Q into a decimal number, resulting in a new sequence Q'; and then transform Q' into a sequence of characters as the desired secret message M.

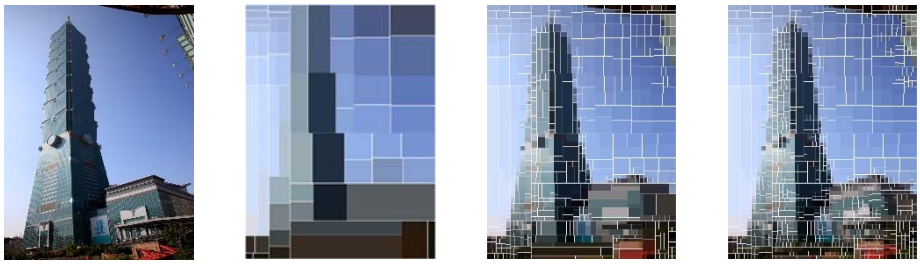
4. EXPERIMENTAL RESULTS AND DISCUSSIONS

In this section, after presenting some experimental results using the proposed algorithms, some discussions about the use of the algorithms are given, followed by a comparison of the proposed method with several existing studies of data hiding in art images is described.

4.1 Experimental Results

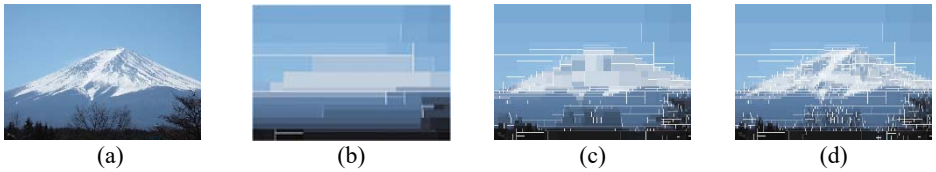
Firstly, some experiments of applying Algorithm 1 to create Neo-Plasticism-like images have been conducted in this study. It is known that different selections of the allowed number of partition iterations (*i.e.*, the value of the threshold I) will result in different effects. Therefore, a user performing the algorithm is allowed to select his/her own value of this partition iteration number because everybody's feeling of art is different. Some Neo-Plasticism-like images yielded by Algorithm 1 using different choices of this iteration number are shown in Figs. 12, 13, and 14, from which we can see that a more abstract style of Neo-Plasticism can indeed be created using a smaller partition iteration number. On the other hand, by using a large partition iteration number, the resulting image will look similar to the source image but with a mosaic effect. Obviously, which choice is the best depends on one's feeling of abstract art.

Next, four examples of experimental results of applying the message embedding and extraction techniques to generated Plasticism-like images using Algorithms 1-4 for covert communication are shown in Figs. 15-18. In the first example shown in Fig. 15. Fig. 15 (a) shows the source image, and Fig. 15 (b) shows a Neo-Plasticism-like image generated by Algorithm 1 with no secret message embedded. By using the source image of Fig. 15 (a) and a secret key "life" as the inputs, a stego-image generated by Algorithm 2 (using Technique 1 for data embedding) with the message "Enjoy your own life without comparing it with that of another" embedded is shown in Fig. 15 (c).



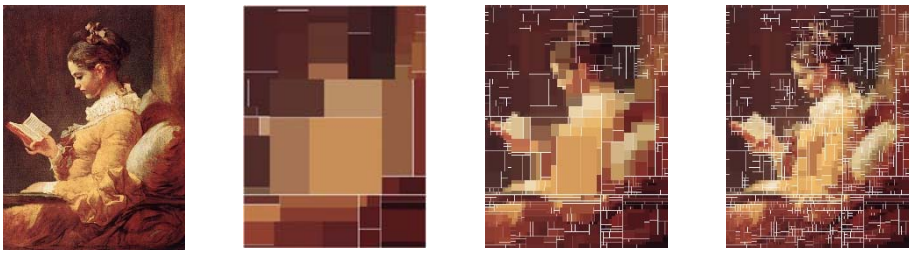
(a) (b) (c) (d)

Fig. 12. Experimental results of art image creation – Example 1; (a) A source image; (b) A Neo-Plasticism-like image created from (a) through 5 partition iterations; (c) A Neo-Plasticism-like image created from (a) through 10 partition iterations; (d) A Neo-Plasticism-like image created from (a) through 15 partition iterations.



(a) (b) (c) (d)

Fig. 13. Experimental results of art image creation – Example 2; (a) A source image; (b) A Neo-Plasticism-like image created from (a) with through 5 partition iterations; (c) A Neo-Plasticism-like image created from (a) through 10 partition iterations; (d) A Neo-Plasticism-like image created from (a) through 15 partition iterations.



(a) (b) (c) (d)

Fig. 14. Experimental results of art image creation – Example 3; (a) A source image; (b) A Neo-Plasticism-like image created from (a) with through 5 partition iterations; (c) A Neo-Plasticism-like image created from (a) through 10 partition iterations; (d) A Neo-Plasticism-like image created from (a) through 15 partition iterations.

In addition, Fig. 15 (d) shows the correct result of extracting the embedded message from Fig. 15 (c) by Algorithm 4 using the right key "life," while Fig. 15 (e) shows a wrong extraction result using an incorrect key "live." These two message extraction results show the integral correctness of the proposed data embedding and extraction processes described by Algorithms 2 and 4, respectively. One more example of results yielded by Algorithm 2 (using Technique 1 for message embedding) and Algorithm 4 is shown in Fig. 16. Furth-

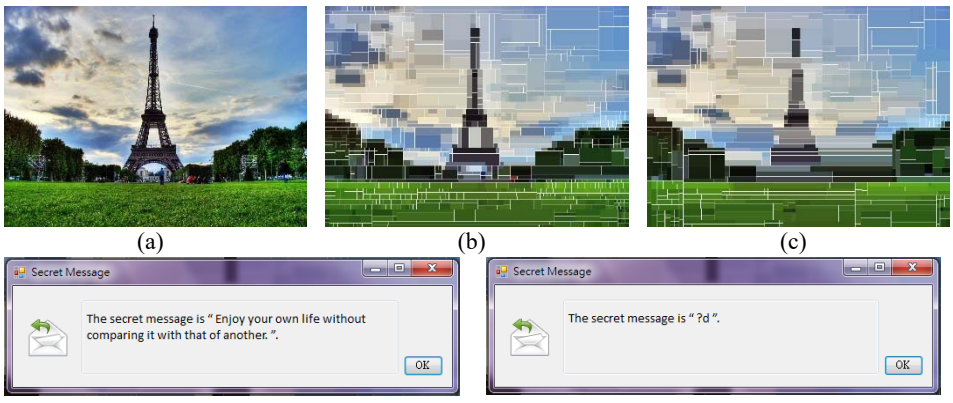


Fig. 15. An experimental result of using the first data hiding technique, Technique 1 (Algorithm 2) and the corresponding data extraction process (Algorithm 4); (a) The source (cover) image; (b) A Neo-Plasticism-like image generated from (a) with no message embedded through 10 partition iterations; (c) A stego-image created from (a) by embedding the message "Enjoy your own life without comparing it with that of another." with the secret key "life."; (d) Correct result of extraction of the message using the right secret key "life."; (e) Incorrect result of extraction using the wrong key "live."

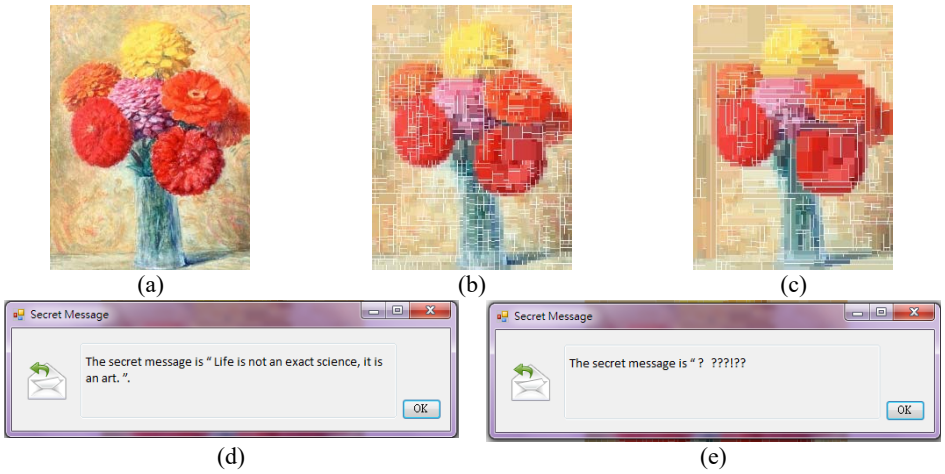


Fig. 16. A second experimental result of using the first data hiding technique, Technique 1 (Algorithm 2) and the corresponding data extraction process (Algorithm 4); (a) The source (cover) image; (b) A Neo-Plasticism-like image generated from (a) with no message embedded through 10 partition iterations; (c) A stego-image created from (a) by embedding the message "Life is not an exact science, it is an art." with the secret key "flower."; (d) Correct result of extraction of the message using the right secret key "flower."; (e) Incorrect result of extraction using the wrong key "flowes."

ermore, two examples of experimental results yielded by Algorithm 3 (using Technique 2 for data embedding) and Algorithm 4 are shown in Figs. 17 and 18, which show the integral correctness of Algorithms 3 and 4.

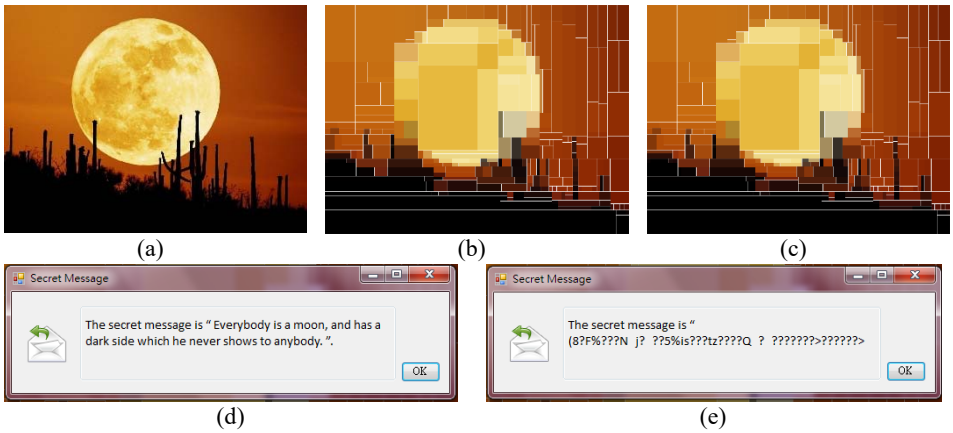


Fig. 17. An experimental result of using the first data hiding technique, Technique 2 (Algorithm 3) and the corresponding data extraction process (Algorithm 4); (a) The source (cover) image; (b) A Neo-Plasticism-like image generated from (a) with no message embedded through 10 partition iterations; (c) A stego-image created from (a) by embedding the message "Everybody is a moon, and has a dark side which he never shows to anybody." with the secret key "moon."; (d) Correct result of extraction of the message using the right secret key "moon."; (e) Incorrect result of extraction using the wrong key "moo."

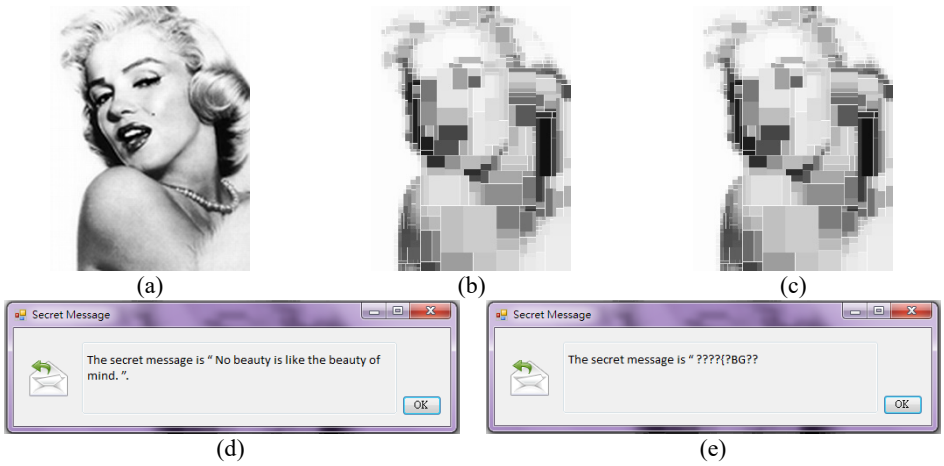


Fig. 18. An experimental result of using the first data hiding technique, Technique 2 (Algorithm 3) and the corresponding data extraction process (Algorithm 4); (a) The source (cover) image; (b) A Neo-Plasticism-like image generated from (a) with no message embedded through 10 partition iterations; (c) A stego-image created from (a) by embedding the message "No beauty is like the beauty of mind." with the secret key "woman."; (d) Correct result of extraction of the message using the right secret key "woman."; (e) Incorrect result of extraction using the wrong key "womal."

It can be observed that the resulting patterns of the partition lines found in Figs. 15 (b) and (c) look different from each other. This is owing to the fact that the horizontal and vertical partition lines used in data embedding by Technique 1 (Algorithm 2) are applied in an alternative order to generate the art image of Fig. 15 (c) while this alternation is not carried out while using Algorithm 2 to generate the art image of Fig. 15 (b) which includes no embedded message. The same observation can be found in Figs. 16 (b) and (c). However, this situation of differences in partition-line patterns is not found in Figs. 17 (b) and (c), and neither in Figs. 18 (b) and (c); instead, Figs. 17 (b) and (c) look identical to each other, and so do Figs. 18 (b) and (c). The reason is that the operation of alternatively applying horizontal and vertical partition lines is not adopted in the message embedding process using Technique 2 (Algorithm 3).

4.2 Discussions

Though Algorithm 2 based on the use of Technique 1 has the merit of yielding a visually more balanced art image with the alternatively-appearing partition lines, yet it has the disadvantage of generating a less optimal art image in the sense of maximizing the total measure of the mutual information of the entire image. The case of using Algorithm 3 has the reverse effect, so there is a tradeoff between visual balance and content harmony in choosing Algorithm 2 or 3 for data embedding.

Furthermore, it may be asked in real applications that with an input image S selected already, which algorithm, Algorithm 2 using Technique 1 or Algorithm 3 using Technique 2, should be chosen for embedding a given message M. An answer to this question is to apply Algorithm 2 initially with a *copy* S' of the input image S and the message M as the input data. If the entire message M can be embedded into S' to yield a stego-image O', then take O' as the desired stego-image O. Otherwise, discard O' and repeat the message embedding process using instead Algorithm 3 with the *original* input image S and the message M as the input data to get a stego-image O as the output.

Of course, there might still arise the case that the message M still cannot be fully embedded into the input image S by Algorithm 3. In this case, a solution is to choose a more complicated image S' with less uniform regions so that more rectangles can be generated as intact regions during the message embedding process (using either Algorithm 2 or 3) in order to hide more message bits.

4.3 Comparisons with Existing Methods

There are very few studies on data hiding *via art images* as done in this study. Some of them are Hung *et al.* [20], Hung *et al.* [21], and Liu and Tsai [22]. In this section, we compare the proposed method with these three methods, with the result shown in Table 1. Hung *et al*. [20] generate a type of so-called stained-glass image, and embed a binary message into the cover art image for information protection by modifying the number of nodes in the tree structure obtained from scanning the art image pixels. Hung *et al.* [21] create a type of so-called tile mosaic image and embed a watermark image into the cover art image for the purpose of watermarking for copyright protection by modifying the orientations of the tiles in the art image. Lai and Tsai [22] construct a type of so-called line-based Cubism-like image and embed a binary message into the cover art image for the purpose of lossless

data hiding by modifying the region colors in the art image while keeping the average colors unchanged.

It is noted by the way that the art-image generation and data embedding schemes used by the four methods are all different because the line and region structures in these types of art image for use in the schemes are different. As a consequence, the qualities of the art images generated by these methods and the bit embedding capabilities yielded of them are also all different and variable.

Table 1. A comparison of the proposed method with three art generation methods with data hiding capabilities.

Method / Item	Hung *et al.* [20]	Hung *et al.* [21]	Liu and Tsai [22]	Proposed Method
Type of art image	Stained-glass image	Tile mosaic image	Line-based Cubism-like image	Rectangle-based Neo-Plasticism-like image
Hidden object type (an example)	A secret message (‘Damou, one thing I did not dare to mention to you till today is: I love you.’) (in Chinese)	A watermark image ()	A secret message (‘Meet me at 21:30. See you.’)	A secret message (‘Everybody is a moon, and has a dark side which he never shows to anybody.’)
cover image (of the above example)				
stego-image (of the above example)				
Data hiding technique	Modifying the number of tree nodes in the stained image	Modifying the orientations of the tiles in the tile image	Modifying the region colors in the Cubism-like image while keeping the average colors unchanged.	Replacing the LSBs of each region’s colors or generating additional partition lines in the region
Application purpose	Information protection	Watermarking for copyright protection	Lossless data hiding	Covert communication

5. CONCLUDING REMARKS

A type of computer art, called rectangle-based Neo-Plasticism-like image, has been proposed in this study, which has the spirit of abstraction of the Plasticism art school in terms of simple colored shape and line components. To create such a type of image by computers, an automatic generation method has also been proposed, which applies recursive binary-space partitioning to a source image by finding the maximum mutual information of the spatial positions and the image intensities of the resulting sub-regions. After partitioning the image into rectangular regions using vertical and horizontal partition lines through a number of partition iterations, each intact region is recolored with the average color of the pixels in the region, yielding an art image which appears to have the style of abstraction of the Neo-Plasticism art.

Furthermore, two data hiding techniques have been proposed for covert communication via the proposed Neo-Plasticism-like image which, by its artistic appearance, hopefully will reduce the hacker's suspicion of the existence of the hidden secret message in the image. Both techniques utilize the characteristics of the Neo-Plasticism-like image creation process to build binary partition trees which are then followed to embed secret message bits into the LSBs of the color values of the rectangular regions or into the colors of the nearly-invisible horizontal and vertical partition lines in the regions. A corresponding data extraction process which can recover the message embedded by either of the two data hiding techniques has also been proposed.

In addition, data security has also been considered seriously and three different protection measures have been implemented, including randomizations of the bit sequence of the input message and the priorities of the partition directions of the sub-regions, as well as embedding of fake message data to pretend hidden secret messages. These measures hopefully will stop or interfere with the hacker from stealing the secret message. Effectiveness of the proposed techniques were proved by good experimental results.

ACKNOWLEDGEMENTS

This work was supported partially by the Ministry of Science and Technology, Taiwan under Projects No. NSC 99-2631-H-009-001 and No. NSC 100-2631-H-009-001 of which the third author Wen-Hsiang Tsai is the PI and the second author Da-Chun Wu is the Co-PI of both projects. A short paper with the basic idea of this study was published in [23].

REFERENCES

1. L. Avila and M. Bailey, "Art in the digital age," *IEEE Computer Graphics and Applications*, Vol. 36, 2016, pp. 6-7.
2. A. Hertzmann, "A survey of stroke-based rendering," *IEEE Computer Graphics and Applications*, Vol. 23, 2003, pp. 70-81.
3. A. Hertzmann, "Painterly rendering with curved brush strokes of multiple sizes," in *Proceedings of International Conference on Computer Graphics and Interactive Technology*, 1998, pp. 453-460.

4. A. Hertzmann, "Fast paint texture," in *Proceedings of International Conference on Computer Graphics and Interactive Technology*, 2002, pp. 91-96.
5. M. P. Salisbury, M. T. Wong, J. F. Hughes, and D. H. Salesin, "Orientable textures for image-based pen-and-ink illustration," in *Proceedings of International Conference on Computer Graphics and Interactive Techniques*, 1997, pp. 401-406.
6. N. Matsumura, H. Tokura, Y. Kuroda, Y. Ito, and K. Nakano, "Tile art image generation using conditional generative adversarial networks," in *Proceedings of the 6th International Symposium on Computing and Networking Workshops*, 2018, pp. 209-215.
7. D. Mould, "A stained glass image filter," in *Proceedings of the 14th Eurographics Workshop on Rendering*, 2003, pp. 20-25.
8. A. Hausner, "Simulating decorative mosaics," in *Proceedings of International Conference on Computer Graphics and Interactive Technology*, 2001, pp. 573-580.
9. M. A. N. I. Fahim and S. Hossain, "A simple way to create pointillistic art from natural images," in *Proceedings of the 3rd IEEE International Conference on Cybernetics*, 2017, pp. 1-5.
10. C. K. Chan and L. M. Cheng, "Hiding data in images by simple LSB substitution," *Pattern Recognition*, Vol. 37, 2004, pp. 469-474.
11. D. C. Wu and W. H. Tsai, "Embedding of any type of data in images based on a human visual model and multiple-based number conversion," *Pattern Recognition Letters*, Vol. 20, 1999, pp. 1511-1517.
12. J. Fridrich, M. Goljan, and R. Du, "Lossless data Embedding – new paradigm in digital watermarking," *EURASIP Journal on Applied Signal Processing*, Vol. 2, 2002, pp. 185-196.
13. M. Awrangjeb and M. S. Kankanhalli, "Reversible watermarking using a perceptual model," *Journal of Electronic Imaging*, Vol. 14, 2005, No. 013014.
14. J. Tian, "Reversible data embedding using a difference expansion," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, 2003, pp. 890-896.
15. C. de Vleeschouwer, J. F. Delaigle, and B. Macq, "Circular interpretation of bijective transformations in lossless watermarking for media asset management," *IEEE Transactions on Multimedia*, Vol. 5, 2003, pp. 97-105.
16. Y. L. Lee and W. H. Tsai, "A new data transfer method via signal-rich-art code images captured by mobile devices," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 25, 2015, pp. 688-700.
17. Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 16, 2006, pp. 354-362.
18. C. W. Lee and W. H. Tsai, "A lossless large-volume data hiding method based on histogram shifting using an optimal hierarchical block division scheme," *Journal of Information Science and Engineering*, Vol. 27, 2011, pp. 1265-1282.
19. J. Rigau, M. Feixas, and M. Sbert, "An information theoretic framework for image segmentation," in *Proceedings of IEEE International Conference on Image Processing*, Vol. 2, 2004, pp. 1193-1196.
20. S. C. Hung, D. C. Wu, and W. H. Tsai, "Data hiding in computer-generated stained glass images and its applications to information protection," *IEICE Transactions on Information and Systems*, Vol. E103-D, 2020, No. 04.

21. S. C. Hung, T. Y. Liu, and W. H. Tsai, "A new approach to automatic generation of tile mosaic images for data hiding applications," in *Proceedings of Conference on Digital Contents Management and Applications*, 2005, pp. 11-20.
22. S. C. Liu and W. H. Tsai, "Line-based cubism-like image – A new type of art image and its application to lossless data hiding," *IEEE Transactions on Information Forensics and Security*, Vol. 7, 2012, pp. 1448-1458.
23. S. C. Liu, D. C. Wu, and W. H. Tsai, "Rectangle-based neo-plasticism-like image – A new type of image and its application to data hiding by binary space partitioning," in *Proceedings of Conference on Computer Vision, Graphics and Image Processing*, 2019.

Shan-Chun Liu (劉珊君) received the B.S. degree in 2009 and the M.S. degree in 2011, both in Computer Science from National Chiao Tung University, Hsinchu, Taiwan. She was a Research Assistant in the Computer Vision Laboratory in the Department of Computer Science at National Chiao Tung University. She is currently with Taiwan Semiconductor Manufacturing Company, Limited, Hsinchu, Taiwan. Her current research interests include information hiding, image processing, and computer art.

Da-Chun Wu (吳大鈞) was born in Taiwan in 1959. He received the B.S. degree in Computer Science and the M.S. degree in Information Engineering from Tamkang University, Taipei, Taiwan, in 1983 and 1985, respectively, and the Ph.D. degree in Computer and Information Science from National Chiao Tung University, Hsinchu, Taiwan, in 1999. He joined the faculty of the Department of Information Management at Ming Chuan University, Taipei, in 1987. From 2002 to 2018, he was with National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan. From 2010 to 2014, he was the Director of Library and Information Center of the university, and from 2015 to 2018, he was the Head of the Department of Computer and Communication Engineering.

Dr. Wu is currently a Professor in the Department of Computer and Communication Engineering at National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan. His recent interests include multimedia security, image processing, machine learning, and artificial intelligence.

Wen-Hsiang Tsai (蔡文祥) received the B. S. degree in EE from National Taiwan University, Taiwan, in 1973, the M. S. degree in EE from Brown University, USA in 1977, and the Ph. D. degree in EE from Purdue University, USA in 1979. Since 1979, he has been with National Chiao Tung University (NCTU), Taiwan, where he is now a Life-time Chair Professor of Computer Science. At NCTU, he has served as the Head of the Department of Computer Science, the Dean of General Affairs, the Dean of Academic Affairs, and a Vice President. From 1999 to 2000, he was the Chair of the Image Processing and Pattern Recognition Society of Taiwan, and from 2004 to 2008, the Chair of the IEEE Computer Society in Taiwan. From 2004 to 2007, he was the President of Asia University, Taiwan.

Dr. Tsai has been an Editor or the Editor-in-Chief of several international journals, including Pattern Recognition, IEEE Transactions on Information Forensics and Security, and the Journal of Information Science and Engineering. He has published 162 journal papers and 259 conference papers, and received 56 paper awards. His research interests include computer vision, information security, and autonomous vehicle applications.