

Line-based Cubism-like Image – A New Type of Art Image and Its Application to Lossless Data Hiding

Shan-Chun Liu and Wen-Hsiang Tsai, *Senior Member, IEEE*

Abstract—A new method of combining art image generation and data hiding to enhance the camouflage effect for various information hiding applications is proposed. First, a new type of computer art, called line-based Cubism-like image, which keeps a characteristic of the Cubism art — abstraction by prominent lines and regions from multiple viewpoints — is proposed. In the creation process with an input source image, prominent line segments in the image are detected and rearranged to form an abstract region-type art image of the Cubism flavor. Data hiding with the minimal distortion is carried out skillfully during the process of re-coloring the regions in the generated art image by shifting the pixels' colors for the minimum amount of ± 1 while keeping the average colors of the regions unchanged. Based on a rounding-off property in integer-valued color computation, the proposed data hiding technique is proved by theorems to be reversible, and thus useful for lossless recovery of the cover art image from the stego-image. Four security enhancement measures are also adopted to prevent hackers from extracting embedded data correctly. Experimental results show the feasibility of the proposed method.

Index Terms—computer art image, line-based Cubism-like image, cover image, stego-image, reversible data hiding.

I. INTRODUCTION

In recent years, the topic of automatic art image creation via the use of computers arouses interests of many people, and many methods have been proposed [1]-[9]. Hertzmann [1] surveys the ideas and algorithms of creating art images by *stroke-based rendering* which is an automatic approach to creating non-photorealistic imagery by the use of discrete elements like paint strokes and stipples. The common goal of creating these image styles is to make the generated art images look like some other types of images. For example, two images created by watercolor painting and oil painting in Hertzmann [2] and Hertzmann [3], respectively, are shown in Fig. 1. Some examples of other types of computer art images are shown in

Fig. 2, where Fig. 2(a) is one created by pen-and-ink drawing proposed by Salisbury [4], Fig. 2(b) is a stipple image via a stipple placement method proposed by Mould [5], and Fig. 2(c) shows a stained-glass image created by an image filter presented in Mould [6].

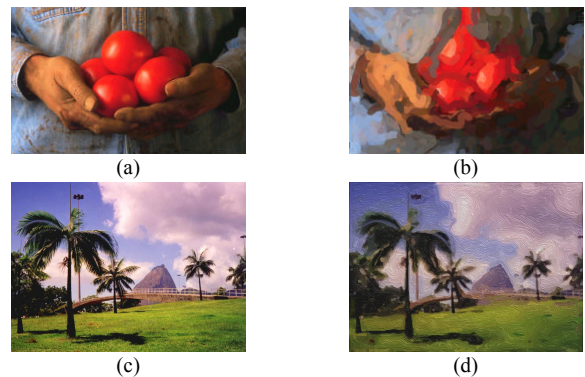


Fig. 1. Art images created by Hertzmann [2], [3]. (a) and (c) Original images (b) Created art image with watercolor painting effect [2]. (d) Created art image with oil painting effect [3].

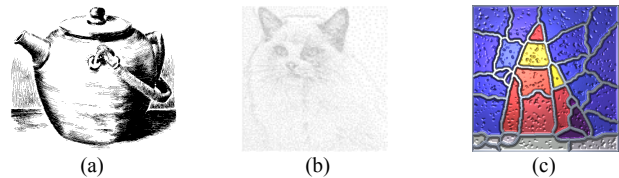


Fig. 2. Some types of computer-generated art images. (a) A pen-and-ink drawing from Salisbury [4]. (b) A stipple image from Mould [5]. (c) A stained glass image from Mould [6].

Mosaic image is also a type of computer art image about which many investigations have been conducted. Each mosaic image is composed of many small identical tiles, such as squares, circles, triangles, and so on. Different from conventional mosaic images which have tiles all arranged in a fixed orientation, Hausner [7] created a type of tile mosaic image by placing tiles to follow the edges in the input image to make the created art image look smoother. An example is shown in Fig. 3(a). Another important criterion for art image creation is to limit the number of strokes so that the resulting image looks like an abstract painting, such as the image shown in Fig. 3(b) which comes from Haeberli [8]. Besides, Song, et al. [9] produces an abstract synthetic art by fitting shapes like

Manuscript received November 29, 2011. This work was supported in part by the National Science Council, Taiwan under Grant 99-2631-H-009-001.

Shan-Chun Liu is with the Institute of Computer Science and Engineering at National Chiao Tung University, Hsinchu, Taiwan 30010 (e-mail: azurecoral@gmail.com).

Wen-Hsiang Tsai is with the Department of Computer Science at National Chiao Tung University, Hsinchu, Taiwan 30010. He is also with the Department of Information Communication at Asia University, Taichung, Taiwan 41354 (Tel.: 886-3-5731763, fax: 886-3-5721490, e-mail: whtsai@cis.nctu.edu.tw)

triangles or rectangles to the regions in segmented images, like the one shown in Fig. 3(c).

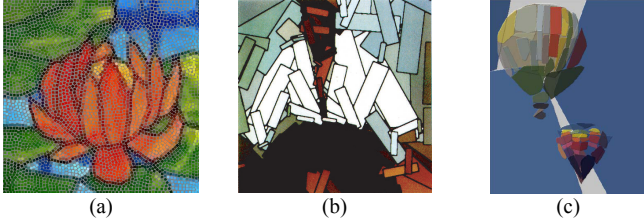


Fig. 3. Some more computer-generated art images (a) Image created by Hausner [7]. (b) Image created by Haerberli [8]. (c) Image created by Song, et al. [9].

Some paintings of the Cubism style are dominated by lines and regions, like those shown in Fig. 4, to show abstractly a characteristic of the Cubism art — multiple-viewpoint. In this study, we try to imitate such line-type Cubism paintings to create automatically an abstract type of art image, called *line-based Cubism-like image*, from a given source image. Line segments in the source image are detected by appropriate image processing techniques, and prominent ones are kept after removing noise line segments. Regions formed by the prominent lines then are created and the pixels in each region are re-colored identically by the average color of the region. Two art images so created in this study are shown in Fig. 5.

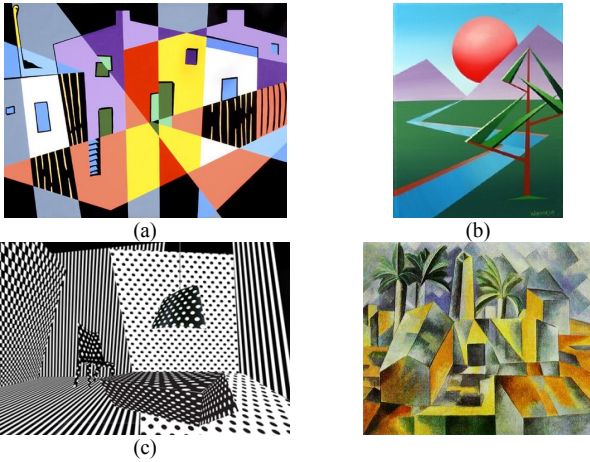


Fig. 4. Some images of line-type Cubism paintings. (a) “After Colonial Cubism” by Robert Rooney, 1993. (b) “Planet X Landscape Painting” by Mark Webster, 2010. (c) “New Cubism” by Koichiro Kimura, 2011. (d) “Factory, Horta de Ebbo” by Pablo Picasso, 1909.

On the other hand, Davis [20] proposed the concept of signal rich art, and pointed out watermarking as a key component in achieving such art. Data hiding is a technique for watermarking and other applications, which embeds data imperceptibly into a *cover image* (or more generally, into *cover media*), so that people cannot perceive the existence of the hidden data in the resulting *stego-image* (or *stego-media*). Otori and Kuriyama [21] hid data into texture images with the hidden data being robustly recoverable from images photographed from print media. Uccheddu et al. [22] proposed a wavelet-based blind technique for watermarking 3D models. It is desired in this study to hide message data into the generated art image for various applications. It is also hoped that the characteristic of

the art image creation process can be utilized effectively to carry out the data embedding work. This way of combining art image creation and data hiding, which may be called *aesthetic data hiding*, is a new idea of information hiding. Attracted by the art exhibited by the image, people hopefully will pay no attention to the hidden data in the art image; and via this camouflage effect, the embedded data can be kept securely or transmitted covertly.

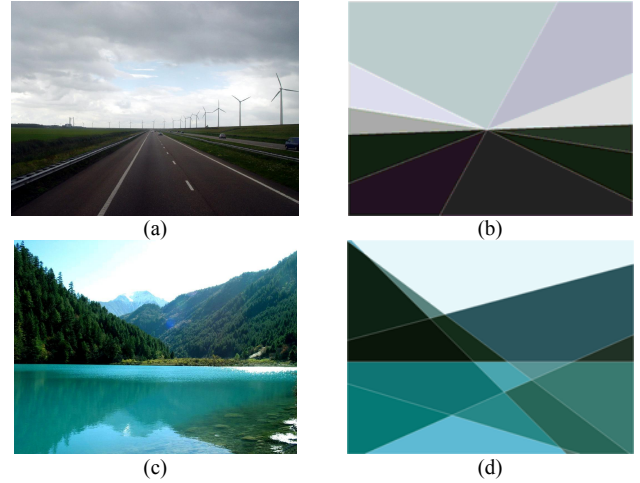


Fig. 5. Examples of line-based Cubism-like images created in this study. (a) and (c) Source images. (b) & (d) Created art images from the source images.

Two criteria for designing data hiding techniques are *imperceptibility* of distortion in the stego-image due to data embedding and *recoverability* of the original cover image content from the stego-image. To achieve imperceptibility, a weakness of the human visual system in differentiating small color or grayscale differences is often utilized, e.g., by the least significant bit (LSB) modification scheme proposed by Chan and Cheng [10] or by the contrast-keeping data embedding scheme proposed by Wu and Tsai [11]. In this study, a scheme of using the *minimum* color shiftings of ± 1 for data embedding is proposed. To achieve recoverability of the cover image which requires lossless data embedding, the most common approach is to compress a portion of the cover image and embed the result with the intended payload into the cover image, such as 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] and Vleeschouwer, et al. [15]. A third approach is to use the histogram shifting technique which can embed large volumes of data, e.g., Ni, et al. [16] and Lee and Tsai [17]. In this study, we use a new scheme of keeping average region colors unchanged to achieve the goal of lossless data hiding. Note that there exist very few lossless data hiding techniques so far.

More specifically, in this study we hide message data in the automatically-generated Cubism-like image during the image creation process by shifting the colors of the pixels in the image regions slightly for the *minimum* amounts of ± 1 while keeping the average colors of the regions *unchanged*. In this way, the original art style of the image with *uniform* regions may be kept. The color differences in the resulting image are difficult to be

found by a hacker because the human visual system is weak in discriminating small color changes. Also, by constraining the numbers of the embedded binary values of 0's and 1's to keep the average region colors unchanged, the data embedding process can be reversed so that lossless recovery of the cover image from the stego-image can be achieved, as proved by theorems in this study. In this way, the original art style in the cover image can be resumed. The security issue is also considered, and four enhancement measures are proposed. Good experimental results show the feasibility of the proposed method.

In the remainder of this paper, we describe the proposed technique for creating the line-based Cubism-like art image automatically in Section 2, the proposed data hiding technique in Section 3, followed by conclusions and some suggestions for future studies in Section 4. Experimental results are also shown in Sections 2 and 3 to demonstrate the feasibility of the proposed method.

II. LINE-BASED CUBISM-LIKE IMAGE CREATION PROCESS

A. Idea of Line-based Cubism-like Image Creation

Cubism artists transform a natural scene into geometric forms in paintings by breaking up, analyzing, and re-assembling objects in the scene from multiple viewpoints. In addition, with the scene objects rearranged to intersect at random angles, each Cubism painting seems to be composed of intersecting lines and fragmented regions in an abstract style. The idea of the proposed art image creation technique is inspired by these concepts of the Cubism art.

Specifically, there are two major stages in the proposed line-based Cubism-like image generation process — *prominent line extraction* and *region re-coloring*. In the first stage, at first we extract line segments from a given source image by edge detection and the Hough transform. Then, we conduct short line segment filtering and nearby line merging. In the second stage, at first we create regions in the image by extending the line segments to the image boundary to partition the image space. Then, we re-color the regions by the average region colors and whiten the boundaries of the regions.

B. Algorithm for Line-based Cubism-like Image Creation

The details of the above process are described as an algorithm in the following.

Algorithm 1: line-based Cubism-like image creation.

Input: a source image S , the minimum line segment length L_{\min} , and the minimum line distance D_{\min} .

Output: a line-based Cubism-like image S_C .

Steps.

Stage 1 — Prominent line extraction.

Step 1. (*Edge detection*) Apply Canny edge detection [18] to image S , resulting in a new image S' of edge points.

Step 2. (*Line segment detection*) Applying the Hough transform [19] to S' to find a list of line segments $L_1,$

L_2, \dots, L_m sorted according to their lengths, yielding a second new image S'' of the line type.

Step 3. (*Prominent line extraction*) Find prominent lines in S'' by the following steps.

3.1 Select those line segments in S'' with lengths larger than threshold L_{\min} and discard the others, resulting in a shorter list of line segments L_1', L_2', \dots, L_n' .

3.2 For all $i = 0$ through n and all $j = 0$ through n with $i \neq j$ and both L_i' and L_j' not deleted yet, compare L_i' and L_j' and if the distance between L_i' and L_j' is smaller than threshold D_{\min} , then delete the shorter one of L_i' and L_j' .

Stage 2 — Region re-coloring.

Step 4. (*Line extension*) Extend each remaining line segment in S'' to the image boundaries of S'' .

Step 5. (*Region partitioning*) Partition S'' into regions R_1, R_2, \dots, R_K by the extended lines.

Step 6. (*Region re-coloring*) Re-color each region R_i in S'' by the following steps with $i = 1, 2, \dots, K$.

6.1 Compute the area A_i (in unit of pixel) of R_i and the average color (C_{ir}, C_{ig}, C_{ib}) of all the pixels in R_i .

6.2 Re-color each pixel in R_i by (C_{ir}, C_{ig}, C_{ib}) .

Step 7. (*Line re-coloring*) Re-color all region boundaries in S'' by the white color.

Step 8. Take the final S'' as the desired line-based Cubism-like image S_C .

Two thresholds, the minimum line segment length L_{\min} and the minimum line distance D_{\min} , are used in Algorithm 1, which affect the flavor of the generated art image according to our observation of the experimental results. Considering the mutual influence between the image size and the line segment length, we take one tenth of the image width as the initial values of L_{\min} and D_{\min} for use in Algorithm 1, and repeated the executions of Algorithm 1 by varying the values of L_{\min} and D_{\min} in our experiment. Some results are shown in Fig. 6 from which we see that a smaller initial value of L_{\min} will cause extraction of more lines, which increases the complexity of the created image and gives an impression closer to the original image content. Contrarily, fewer lines will result from the use of a larger value of L_{\min} , making the resulting image simpler and more abstract. The effect of varying the value of D_{\min} is similar.

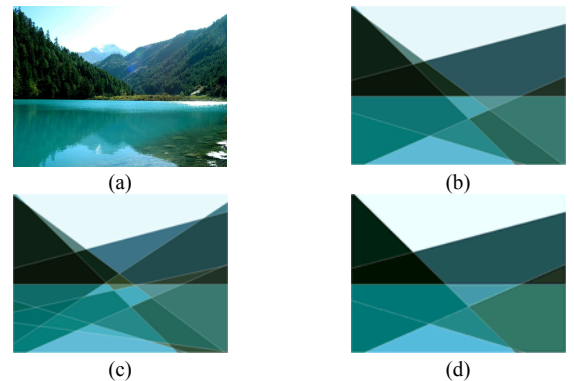


Fig. 6. Experimental results of varying threshold values of D_{\min} and L_{\min} . (a) Source image with size 1024×768 . (b) Art image created from (a) with initial values of $D_{\min} = 102, L_{\min} = 102$. (c) Art image created from (a) with $D_{\min} = 102, L_{\min} = 20$. (d) Art image created from (a) with $D_{\min} = 200, L_{\min} = 102$.

C. Experimental Results

From the above discussions, we see that different selections of the two threshold values L_{\min} and D_{\min} will result in totally different visual effects in the created art images. However, it is difficult to decide which result is better than the others because the decision is obviously dependent on people's individual feelings of art. Therefore, in this study we just offer a series of results yielded by the use of different sets of values of the two thresholds for the user to inspect and choose. Specifically, we use the three values of 0.5/10, 1/10, and 2/10 times the image width as the values for the thresholds L_{\min} and D_{\min} . As a result, each threshold has three choices, resulting in nine choices of the threshold pair. Then, we generate nine art images, each corresponding to one of the nine threshold combinations, for the user to choose as his/her favorite art image. Two examples using input images of a university church and the Eiffel Tower are shown in Figs. 7 and 8. In each of the two examples, the source image is shown in (a) the result of using the initial values of the two thresholds (1/10 times the image width) is shown in (b). A seemingly better choice is shown in (l).

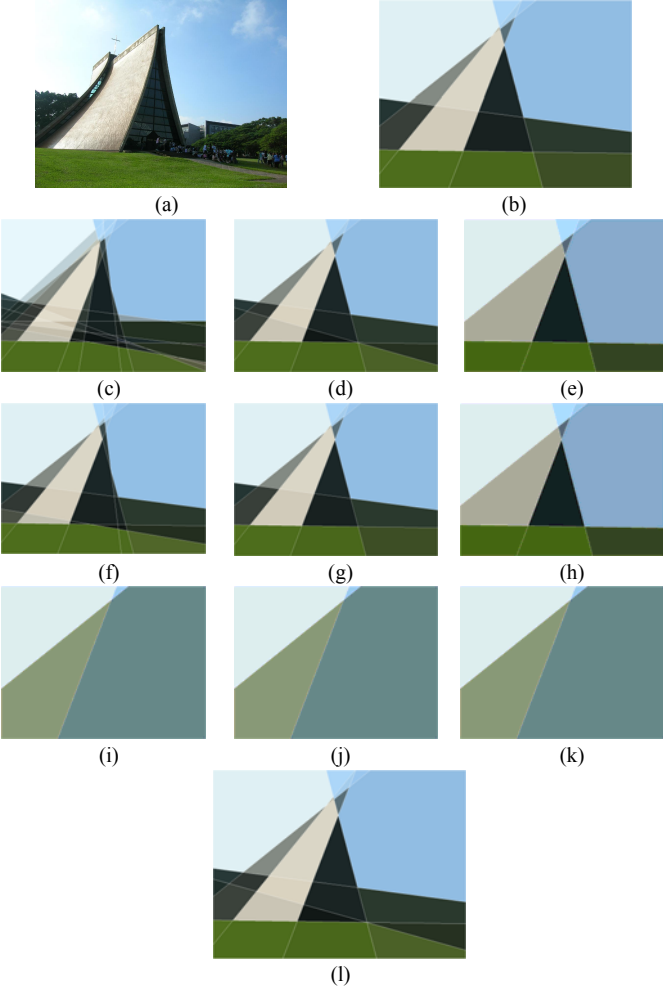


Fig. 7. Experimental results with a church image as input. (a) Source image with size 1024×768 . (b) Initial $D_{\min} = 102$ and initial $L_{\min} = 102$. (c) $(D_{\min}, L_{\min}) = (51, 51)$. (d) $(D_{\min}, L_{\min}) = (51, 102)$. (e) $(D_{\min}, L_{\min}) = (51, 204)$. (f) $(D_{\min}, L_{\min}) = (102, 51)$. (g) $(D_{\min}, L_{\min}) = (102, 102)$. (h) $(D_{\min}, L_{\min}) = (102, 204)$. (i) $(D_{\min}, L_{\min}) = (204, 51)$. (j) $(D_{\min}, L_{\min}) = (204, 102)$. (k) $(D_{\min}, L_{\min}) = (204, 204)$. (l) A seemingly better choice of the 9 images – (d).

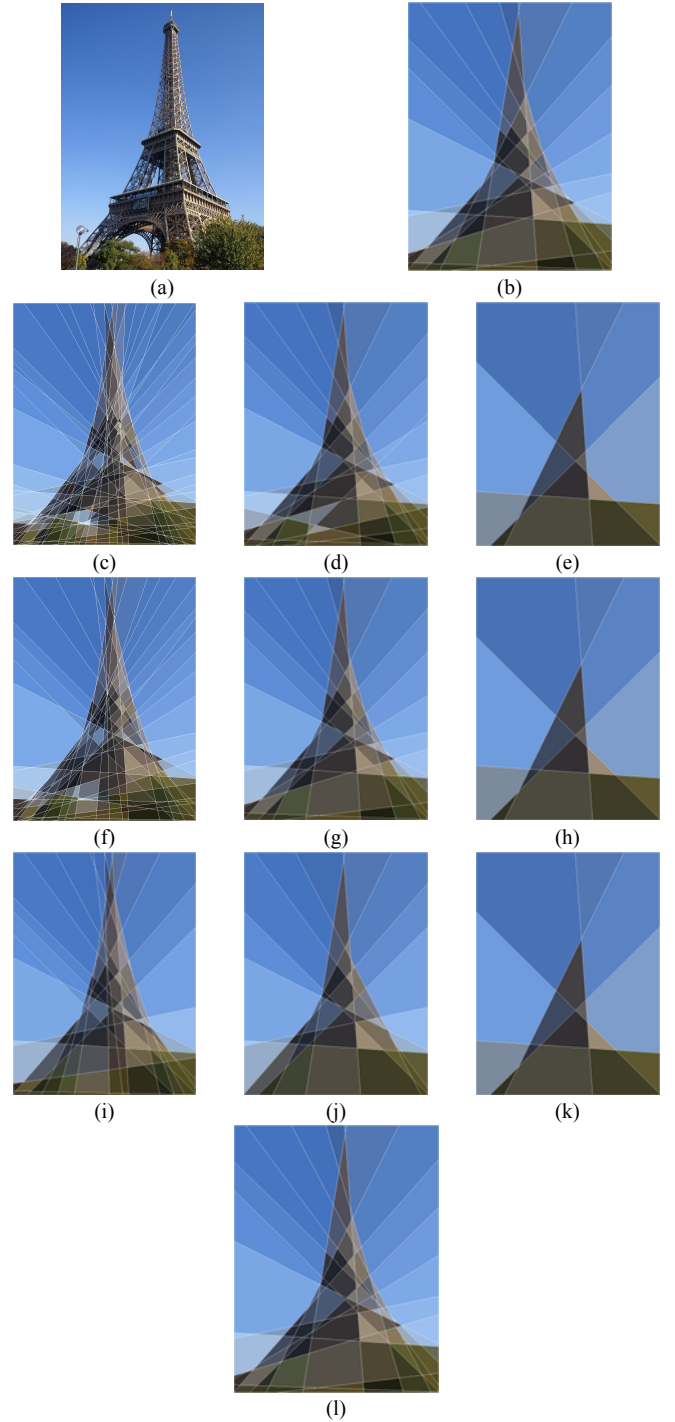


Fig. 8. Experimental results with an image of Eiffel Tower as input. (a) Source image with size 768×1024 . (b) Initial $D_{\min} = 102$ and initial $L_{\min} = 102$. (c) $(D_{\min}, L_{\min}) = (51, 51)$. (d) $(D_{\min}, L_{\min}) = (51, 102)$. (e) $(D_{\min}, L_{\min}) = (51, 204)$. (f) $(D_{\min}, L_{\min}) = (102, 51)$. (g) $(D_{\min}, L_{\min}) = (102, 102)$. (h) $(D_{\min}, L_{\min}) = (102, 204)$. (i) $(D_{\min}, L_{\min}) = (204, 51)$. (j) $(D_{\min}, L_{\min}) = (204, 102)$. (k) $(D_{\min}, L_{\min}) = (204, 204)$. (l) A seemingly better choice of the 9 images – (g)

III. DATA HIDING VIA LINE-BASED CUBISM-LIKE IMAGES

A. Idea of Proposed Data Hiding Technique

In the proposed Cubism-like image creation process described by Algorithm 1 above, one major step is to re-color the pixels in each image region with the average color of all the

pixels in the region, resulting in an image visually looking like the source image. We take advantage of this step as well as a weakness of the human visual system in differentiating small color changes to design the data hiding technique in this study.

To be more specific, the proposed data hiding technique embeds message data into a cover cubism-like image by changing each pixel's color value in the cover image for the *minimum* amount of ± 1 in each color channel. As a result, people cannot tell the visual difference between the cover image and the stego-image. This effect, in addition to that of attracting people by the artistic content of the Cubism-like image, gives the proposed data hiding technique a camouflage effect which arouses no suspicion from hackers. Furthermore, a *reversible* region re-coloring scheme, which keeps the average color of each region unchanged, is designed as a substitute of the original re-coloring process in Algorithm 1. This reversibility guarantees that we can extract the data embedded in the stego-image to restore the original content of the cover image *losslessly*. It is also noted that changing pixels' colors slightly while keeping average region colors unchanged, as proposed in this study, creates integrally a *mosaic* effect in the regions, which makes the stego-image look nearly identical to the cover image and thus enhances the camouflage effect of the proposed technique.

The proposed data hiding technique as described above is designed according to theorems derived in this study from the rounding-off property in integer-valued color computation. The details are described next. We assume that each image used in this study is a color one of the RGB model.

B. Principle of Lossless Data Embedding

In the proposed region re-coloring process, when embedding a bit b into a pixel with color c , if b is 0, then we decrement c by an integer value a , and if b is 1, then we increment c by a . After hiding message bits into the pixels' colors in a region by color shifting in this way, the region's average color will also be changed. It is found in this study that the property of *rounding-off* in integer computation may be utilized to modify this region re-coloring process to keep the average region color unchanged, resulting in a reversible region re-coloring process, as proved in the following.

Lemma 1. The total numbers N_0 and N_1 of data bits of 0's and 1's, respectively, embeddable in an image region R with area A (in unit of pixel) by re-coloring R by color shiftings of the amount of $\pm a$ without changing the average color C_o of R in each of the three color channels is constrained by the following inequality:

$$-A/2 \leq (N_1 - N_0) \times a < A/2. \quad (1)$$

Proof.

For each of the three color channels R , G , and B , first let C_1, C_2, \dots, C_A be the color values of the A pixels in region R . Then, the average color value C_o of R is

$$C_o = (C_1 + C_2 + \dots + C_A)/A.$$

Next, while re-coloring R by color shiftings of the amount of

$\pm a$, if the average region color is to be kept unchanged, the computed average color C of R should lie in the range of $C_o - 0.5$ to $C_o + 0.5$ so that C can be *rounded* to be a value identical to the original average value C_o because the color values in each color channel are *integer numbers*. That is, the following inequality should hold:

$$-0.5 + C_o \leq C < C_o + 0.5.$$

Third, if N_0 0's and N_1 1's are embedded into R , then there arises a total extra amount of color shiftings of $-N_0 \times a + N_1 \times a$. Therefore, the new average region color C is

$$C = (C_1 + C_2 + \dots + C_A - N_0 \times a + N_1 \times a)/A.$$

Combining the three equalities derived above, we can get

$$C_o - 0.5 \leq C = [C_o A + (N_1 - N_0) \times a]/A < C_o + 0.5$$

which may be transformed into the following inequality:

$$-A/2 \leq (N_1 - N_0) \times a < A/2. \quad \square$$

Theorem 1. When the data bits are embedded into the cover image to cause the minimum distortion in the mean square error (MSE) sense, the constraint of (1) becomes

$$-A/2 \leq N_1 - N_0 < A/2. \quad (2)$$

Proof.

With each of N_0 pixels' values in each region R being decremented by a and each of N_1 pixels' values in R being incremented by a , it is easy to figure out that the MSE of R in the stego-image with respect to the cover image is just $N_0 \times (-a)^2 + N_1 \times (+a)^2 = (N_0 + N_1) \times a^2$. The minimum value of this MSE occurs obviously when $a = 1$. Consequently, if the data bits are embedded to cause the minimum distortion, a should be taken to be 1 and (1) becomes

$$-A/2 \leq N_1 - N_0 < A/2. \quad \square$$

The inequality (2) above constrains *integrally* the numbers N_0 and N_1 of embeddable message data bits of 0's and 1's, respectively, and is used in this study to design the proposed data hiding algorithm. The following corollary gives *respective* constraints to N_0 and N_1 for some extreme cases of data bits.

Corollary 1. When the stego-image is yielded with the minimum distortion by color shiftings of the amount of ± 1 in region re-coloring, the maximum numbers N_0 and N_1 of embedded 0's and 1's are constrained respectively by

$$A/4 \leq N_0 < 3A/4; \quad A/4 < N_1 \leq 3A/4. \quad (3)$$

And for the extreme case that the data bits are either all 0's or all 1's, the numbers N_0 or N_1 are constrained respectively by

$$N_0 \leq A/2; \quad N_1 < A/2. \quad (4)$$

Proof.

In the best case, the maximum number of embeddable bits in a region is just the number A of pixels in the region, that is,

$$N_0 + N_1 = A$$

which, when combined with (2) above, leads to the following

two inequalities:

$$-A/2 \leq 2N_0 - A < A/2; \quad -A/2 \leq A - 2N_1 < A/2,$$

or equivalently, to

$$A/4 \leq N_0 < 3A/4; \quad A/4 < N_1 \leq 3A/4.$$

respectively, as can be easily figured out.

On the other hand, for the extreme case where the message data are composed of all 0's, then N_1 equals zero, and (2) becomes $-A/2 \leq -N_0 < A/2$, or equivalently, $N_0 \leq A/2$. Similarly, if the message data are composed of all 1's, then N_0 equals zero, and (2) becomes $-A/2 \leq N_1 < A/2$ so that $N_1 < A/2$. \square

Theorem 2. Data embedding in a region by color shiftings of the amount of ± 1 to embed 0's and 1's, respectively, under the constraint of (2) is *lossless*, i.e., the embedded data may be retrieved to resume the original stego-image *perfectly*.

Proof.

If the numbers of 0's and 1's embedded in a region R are N_0 and N_1 , respectively, and if A and C_0 are the area and the original average color of R , then the new average color C of R is

$$C = [C_0A + N_0 \times (-1) + N_1 \times (+1)]/A = C_0 + (N_1 - N_0)/A$$

which, when constrained by (2), leads to

$$C_0 - 0.5 \leq C = C_0 + (N_1 - N_0)/A < C_0 + 0.5.$$

Accordingly, C becomes C_0 after being rounded to the nearest integer for use as a color value.

With the new average color C of region R being unchanged and equal to C_0 , we may compute the difference between C_0 and the color value C_p of each pixel P in R to check if there is a color shifting of ± 1 at P ; if so, then extract the data bit of 0 or 1, respectively, and replace the shifted color C_p of P by the original average region color C_0 . In this way, the original cover art image can be recovered perfectly. This means that the original data embedding scheme is lossless. \square

C. Algorithm of Proposed Data Hiding Technique

The proposed data hiding process, which is based on the creation process of the line-based Cubism-like image and the lossless data embedding principle described previously, is composed of two stages — *data string randomization and embedding capacity computation*; and *data embedding*. In the first stage, at first we transform the data string to be hidden into a digit sequence of 0's and 1's and append an *ending pattern* (with at least one digit other than 0 and 1) to the end of the sequence to keep its length a multiple of three. By the ending pattern, we can determine where the embedded data string ends in a sequence of extracted digits in the later data extraction process. Next, we try to obtain the information of two parameters of each region, namely, its area and average color, by performing Algorithm 1. Then, we use a secret key to randomize the order of the regions in the input image, and take the resulting sequence as the order for data hiding. For each region, in order to keep the average color of the region unchanged, we limit the embedded amount of message data bits in each region by the constraint of (2) in Theorem 1. In the

second stage, we embed the input data sequence by shifting the pixels' colors for the amounts of ± 1 according to the above-mentioned data hiding order. After the data sequence is exhausted, there might exist regions into which no data is embedded. We deal further with these intact regions to keep the coloring style of all regions consistent. For this, we create a random binary string of 0's and 1's with the bit numbers roughly constrained by (2), and use the same data embedding process to embed it into the intact regions. At the end, a stego-image is generated with the input data string embedded imperceptibly. A detailed algorithm to implement these steps is given as follows.

Algorithm 2. Embedding a data string into a Cubism-like image created from a given image.

Input: an image S , a secret key K_s , four random-number generator functions f_1, f_2, f_3 , and f_4 , and a message data string M in character form.

Output: a stego-image S' into which M is embedded.

Steps.

Step 1 — *Data string randomization and embedding capacity computation.*

Step 1. (*Randomizing and segmenting the data string*) Randomize and segment the data string M by the following steps.

- 1.1 Transform M in character form into a binary string, and randomize the order of its bits to generate a new string M' using function f_1 with secret key K_s as the seed.
- 1.2 Regard each bit of M' as a digit and append an ending pattern with at least one and no more than three identical digits d other than 0's and 1 to the end of M' to form a new digit sequence M'' with its length being a multiple of three.
- 1.3 Divide M'' into a sequence of 3-digit segments, m_1, m_2, \dots, m_N .

Step 2. (*Generating an art image and computing related parameters*) Generate an art image and compute the areas and average colors of the regions in the image by the following steps.

- 2.1 Perform Algorithm 1 with S as the input source image to obtain an output art image S' which has regions R_1, R_2, \dots, R_K with areas A_1, A_2, \dots, A_K and average region colors $(C_{1r}, C_{1g}, C_{1b}), (C_{2r}, C_{2g}, C_{2b}), \dots, (C_{Kr}, C_{Kg}, C_{Kb})$, respectively.
- 2.2 Randomize the order of regions R_1 through R_K using function f_2 with secret key K_s as the seed to generate an ordered region sequence $CS = \{R'_1, R'_2, \dots, R'_K\}$, and change in accordance the orders of the areas and average region colors of the regions, resulting in the new ordered sequences of areas and average region colors, $\{A'_1, A'_2, \dots, A'_K\}$ and $\{(C'_{1r}, C'_{1g}, C'_{1b}), (C'_{2r}, C'_{2g}, C'_{2b}), \dots, (C'_{Kr}, C'_{Kg}, C'_{Kb})\}$, respectively.

Step 3. (*Calculating the maximum data embedding capacity of each region*) Take sequentially an *unprocessed* region R'_i in sequence CS , and compute the maximum data

embedding capacity Q_i of R_i' by the following steps with the initial value of Q_i set to be zero.

- 3.1 Let (N_{r0}, N_{r1}) , (N_{g0}, N_{g1}) , and (N_{b0}, N_{b1}) denote the numbers of 0's and 1's embeddable in the R , G , and B color channels, respectively, in R_i' with their initial values all set to be zero.
- 3.2 Take an *unprocessed* 3-digit segment m_i with digits $d_r d_g d_b$ of data string M'' and compute (N_{r0}, N_{r1}) , (N_{g0}, N_{g1}) , and (N_{b0}, N_{b1}) for d_r , d_g , and d_b , respectively, in the following way:
 - (a) if $d_r = 0$, increment N_{r0} by 1; else, increment N_{r1} by 1;
 - (b) if $d_g = 0$, increment N_{g0} by 1; else, increment N_{g1} by 1;
 - (c) if $d_b = 0$, increment N_{b0} by 1; else, increment N_{b1} by 1.
- 3.3 If all of the following three inequalities hold:

$$\begin{aligned} -A_i'/2 &\leq N_{r1} - N_{r0} < A_i'/2; \\ -A_i'/2 &\leq N_{g1} - N_{g0} < A_i'/2; \\ -A_i'/2 &\leq N_{b1} - N_{b0} < A_i'/2, \end{aligned}$$

then increase Q_i by 3; else, regard Q_i to have reached the maximum data embedding capacity for R_i' according to Theorem 1 and go to Step 4.

- 3.4 If the data sequence M'' is not exhausted, then go to Step 3.2 to repeat the above two steps.

Stage 2 — Data embedding.

Step 4. (*Embedding the data*) Perform the following steps to embed data into region R_i' of S' .

- 4.1 Randomize the order of the pixels in R_i' to generate an ordered pixel sequence $HS = \{p_1', p_2', \dots, p_i'\}$ using function f_3 with secret key K_s and the index i of R_i' together as the seed.
- 4.2 Embed an *unembedded* 3-digit segment m_i with digits $d_r d_g d_b$ of sequence M'' into an *unprocessed* pixel p_j' with color values $(C_{jr}', C_{jg}', C_{jb}')$ taken sequentially from pixel sequence HS by the following steps.
 - (a) Obtain new color values $(C_{jr}'', C_{jg}'', C_{jb}'')$ for p_j' by modifying the original ones $(C_{jr}', C_{jg}', C_{jb}')$ in the following way for $h = r, g$, and b :
 - i. increment C_{jh}' by 1 if $d_h = 1$;
 - ii. decrement C_{jh}' by 1 if $d_h = 0$;
 - iii. do nothing to C_{jh}' if $d_h = d$ (an ending pattern digit).
 - (b) Re-color pixel p_j' by the new color values $(C_{jr}'', C_{jg}'', C_{jb}'')$.
 - (c) Decrement the maximum data embedding capacity Q_i by 3.
 - (d) If Q_i is not equal to zero, then go to Step 4.2 to repeat the steps of (a) through (c) above.

Step 5. (*Ending of looping*) Repeat Steps 3 and 4 if sequence M'' is not exhausted.

Step 6. (*Re-coloring the intact regions*) Re-color each region R_k' with area A_k' in S' , which has not been used for data embedding so far, by the following steps.

- 6.1 Create a random digit string B with size $\lfloor A_k'/2 \rfloor - 1$ of 0's and 1's for R_k' using function f_4 with secret key K_s as the seed, and call B a *camouflage string*.
- 6.2 Perform Steps 3 through 5 to re-color the pixels of R_k' to embed camouflage string B .

Step 7. Take the final image S' as the desired stego-image.

In the above algorithm, *wrap-around* problems in color values might occur in Steps of 4.2(a)-i and 4.2(a)-ii when the average region color value in any of the three color channels is 255 or 0. In such cases, we will obtain a stego-image with undesirable noise. To avoid such cases, we adjust the extreme average color values of 255 and 0 in any color channel to be 253 and 1, respectively, before data hiding. Such slight color alternations in the generated stego-image will cause nearly no visual difference to humans but will solve the problem.

D. Data Extraction Process

The proposed data extraction process, which is based on Theorem 2, is basically a reverse version of the proposed data hiding process and consists of two stages — *embedded data extraction and data de-randomization*. In the first stage, we recover the region re-coloring sequence in the stego-image and obtain the area and the average color of each region in the stego-image. Based on the average color of each region, we retrieve the message data embedded in the stego-image by comparing it with the pixels' colors in the region. In the second stage, the retrieved data are de-randomized to get the original message data using the secret key. The details are described as an algorithm in the following.

Algorithm 3: extracting the hidden data string.

Input: a stego-image S ; and the secret key K_s and three random number generators f_1, f_2 , and f_3 used in Algorithm 2.

Output: the data string M embedded in S .

Steps.

Stage 1 — Embedded data extraction.

- Step 1. (*Extracting the regions and related parameters*) Conduct region growing in a raster-scan order to find the regions R_1, R_2, \dots, R_K in S , and compute their respective areas A_1, A_2, \dots, A_K and average colors $(C_{1r}, C_{1g}, C_{1b}), (C_{2r}, C_{2g}, C_{2b}), \dots, (C_{Kr}, C_{Kg}, C_{Kb})$.
- Step 2. (*Retrieving the region re-coloring sequence*) Randomize the initial raster-scan order of the regions R_1 through R_K to retrieve the region re-coloring sequence $CS = \{R_1', R_2', \dots, R_K'\}$ using f_2 with secret key K_s as the seed, and change in accordance the orders of the areas and the average colors of the regions, resulting in the new ordered sequences of areas and average colors, $\{A_1', A_2', \dots, A_K'\}$ and $\{(C_{1r}', C_{1g}', C_{1b}'), (C_{2r}', C_{2g}', C_{2b}'), \dots, (C_{Kr}', C_{Kg}', C_{Kb}')\}$, respectively.
- Step 3. Create a message data sequence M' with an empty initial content.
- Step 4. (*Extracting the embedded data*) Take an *unprocessed* region R_i' in CS and perform the following steps to extract the embedded data in R_i' to compose M' .

- 4.1 Randomize the initial raster-scan order of the pixels in R_i' to retrieve the pixel re-coloring sequence $HS = \{p_1', p_2', \dots, p_T'\}$ using function f_3 with secret key K_s and the index of R_i' together as the seed.
- 4.2 Take sequentially an *unprocessed* pixel p_j' with color $(C_{jr}'', C_{jg}'', C_{jb}'')$ in sequence HS .
- 4.3 Extract three embedded data digits $d_r, d_g,$ and d_b from the difference values between the color $(C_{jr}'', C_{jg}'', C_{jb}'')$ of pixel p_j' and the average color $(C_{ir}', C_{ig}', C_{ib}')$ of region $R_i,$ respectively, by the following way, where $h = r, g,$ and b :
 - (a) if $C_{jh}'' < C_{ih}'$, then set d_h to be 0;
 - (b) if $C_{jh}'' > C_{ih}'$, then set d_h to be 1;
 - (c) if $C_{jh}'' = C_{ih}'$, then set d_h to be d (an ending pattern digit).
- 4.4 If none of $d_r, d_g,$ and d_b is d (the ending pattern digit), then append $d_r d_g d_b$ to M' and go to Step 4 to repeat Steps 4.1 through 4.3; else, ignore digits of d and append the remaining one(s) to M' .

Stage 2 — Data de-randomization.

Step 5. Re-order the digits in M' using f_1 with secret key K_s as the seed, regard the result as a binary string composed of 0's and 1's, and transform it into character form as the desired data string M .

Note that in Step 4.4 above, with the help of the ending pattern digit d , the end of embedded data string can be detected so that extraction of the *camouflage strings*, which were embedded into some regions as conducted in Step 6 of Algorithm 2, can be skipped.

E. Security Consideration

As can be seen, under the usual assumption that the algorithms are known to the public, a hacker could extract the embedded data from a stego-image by the proposed data extraction process described by Algorithm 3. Against this, we adopt four measures in Algorithm 2 to enhance the security of the proposed technique using a secret key: (1) randomization of the data string to be embedded; (2) randomization of the processing order of the regions; (3) randomization of the processing order of the pixels in each region; and (4) embedding camouflage strings in intact regions to mislead a hacker to guess data in them erroneously. With these measures, the risk for the embedded data to be stolen is greatly reduced.

F. Experimental Results

Figs. 9 through 14 show some experimental results of applying the proposed data hiding method to Cubism-like images. Fig. 9(a) and Fig. 12(a) are two source images. Figs. 9(b) and 12(b) are the generated Cubism-like images using Algorithm 1 with no message data embedded. Fig. 9(c) is a stego-image into which a message data string “Meet me at 21:30. See you.” has been embedded with the secret key “test” and Fig. 12(c) is a stego-image into which a message data string “Hi, I am Helen. Nice to meet you!” has been embedded with the secret key “door.” As can be seen, the stego-images are

almost identical to the cover art images of Figs. 9(b) and 12(b). On the other hand, the message data can be retrieved only when the right key is used in the data extraction process, like the results shown in Figs. 10 and 13. If a hacker uses a wrong key in the data extraction process, the extraction work will fail, as shown by the examples of Figs. 11 and 14.

In addition, the stego-images qualities are still good after the average region colors are changed for data embedding, as indicated by the very small MSE and high PSNR values listed in Table 1 of all the stego-images shown previously in this paper. The MSE and PSNR values were computed with respect to the generated Cubism-like images as the cover images.

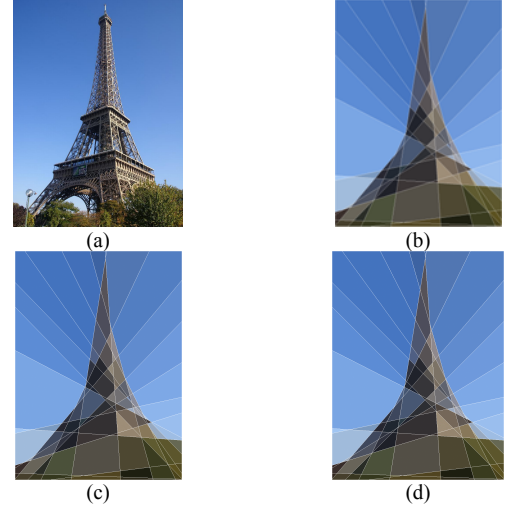


Fig. 9. An experimental result. (a) Source image. (b) Generated Cubism-like image with no message data embedded. (c) Stego-image with color shiftings of ± 1 . (d) Stego-image with color shiftings of ± 1 through ± 8 .

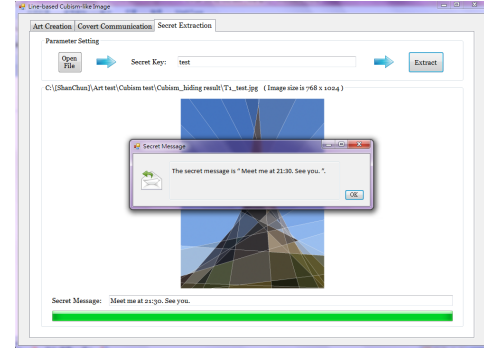


Fig. 10. Result of extracting embedded message data with a right key.

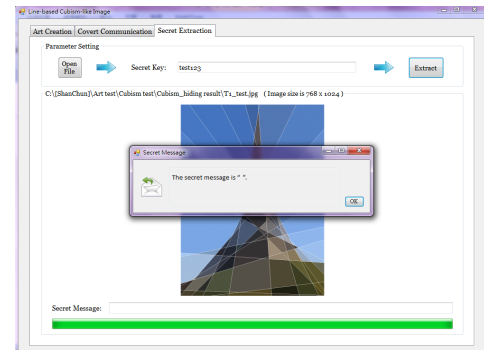


Fig. 11. Result of extracting erroneous message data with a wrong key.

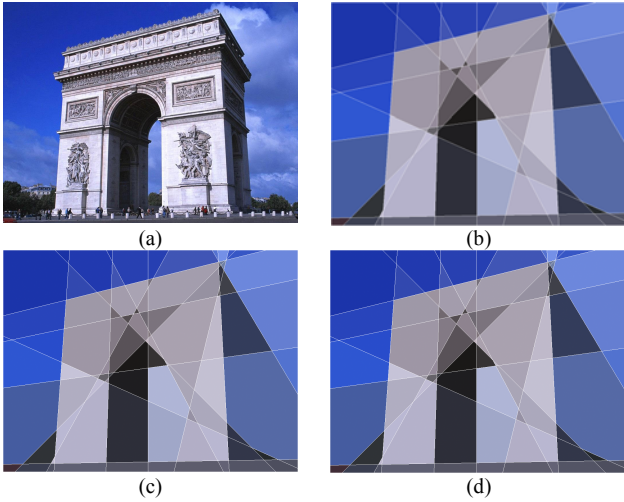


Fig. 12. Another experimental result. (a) Source image (cover image). (b) Generated Cubism-like image with no message data embedded. (c) Stego-image with color shiftings of ± 1 . (d) Stego-image with color shiftings of ± 1 through ± 8 .

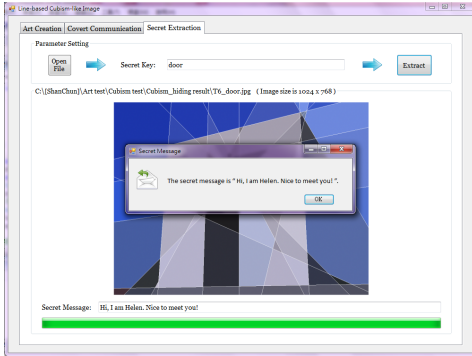


Fig. 13. Result of extracting embedded message data with a right key.

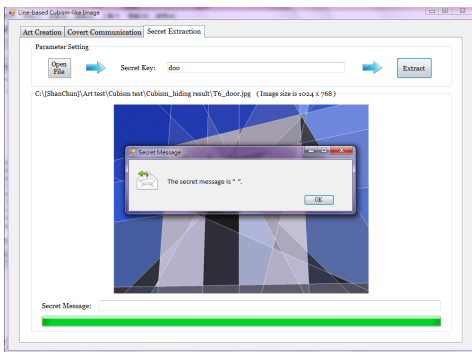


Fig. 14. Result of extracting erroneous message data with a wrong key.

G. Extension of Proposed Method

The proposed method and experimental results presented so far are all based on the color shiftings of ± 1 for data embedding. Actually, an extension of the proposed method based on color shiftings of ± 1 through $\pm a$ with $a > 1$ may also be tried. For example, with $a = 2$, we may use the color shiftings of $-2, -1, +1, +2$ to represent the embeddings of 2-bit message data 00, 01, 10, 11, respectively. In this way, it is not difficult to figure out that the data embedding capacity may be doubled at the sacrifice of stego-image quality degradation. Similarly, with $a = 4$, the capacity may be quadrupled, and so on. To see the

quality degradation trend, we conducted an additional series of experiments for $a = 1, 2, 4$, and 8 using an extended version of the proposed method. The input message is made to be long enough to fill up all the regions in each cover image. Two resulting images for $a = 8$ are shown in Figs. 9(d) and 12(d), which, when compared with Figs. 9(c) and 12(c), respectively, exhibit almost no visible difference. Also, we list all the MSE and PSNR values for $a = 1, 2, 4$, and 8 in Table 2, from which we can see that there is no obvious degradation in the stego-image quality as a increases from 1 to 8.

Table 1. MSE and PSNR of stego-images generated with color shiftings of ± 1 .

Source image	Fig. 5	Fig. 6	Fig. 7	Fig. 8	Fig. 12
Stego-image					
MSE	0.4516	0.4835	0.4827	0.4861	0.4872
PSNR	51.583	51.287	51.294	51.263	51.254

Table 2. MSE and PSNR of stego-images with color shiftings ± 1 through $\pm a$.

Stego-image	Fig. 5	Fig. 6	Fig. 7	Fig. 8	Fig. 12
MSE ($a = 1$)	0.4691	0.5010	0.5001	0.4878	0.4886
PSNR ($a = 1$)	51.419	51.133	51.140	51.248	51.242
MSE ($a = 2$)	0.4739	0.5057	0.5050	0.4931	0.4910
PSNR ($a = 2$)	51.373	51.091	51.098	51.202	51.220
MSE ($a = 4$)	0.4963	0.5281	0.5273	0.4956	0.4935
PSNR ($a = 4$)	51.173	50.904	50.910	51.179	51.198
MSE ($a = 8$)	0.5637	0.5955	0.5948	0.5973	0.5012
PSNR ($a = 8$)	50.620	50.382	50.387	50.369	51.131

H. Comparisons with Other Reversible Data Hiding Methods

To demonstrate more of the feasibility of the proposed method, we have compared it with several other typical reversible data hiding methods [16, 23, 24] based on the histogram modification, difference expansion, and integer-to-integer wavelet transform techniques, respectively.

First, while the other methods hide data *directly* into cover images to yield stego-images, the proposed method instead generates an art image first into which data then are embedded, as shown in Fig. 15. The stego-image so generated is of an art flavor which hopefully will attract from hackers more attention on the artistic content and less suspicion about the hidden data. In contrast, the stego-image yielded by the other methods looks similar to the cover image in most cases.

Next, distortion reduction in the stego-image is a critical issue of other methods, but it is not so for the proposed method because the stego-image yielded by the proposed method is just an art image with an abstract style and so does not have to look precisely like the original cover image. Even if image distortion is cared about, the proposed method yields good-quality stego-images as well with very high PSNR values, all larger than 50, as can be seen from Tables 1 and 2. As to the data hiding capacity, it can be figured out from Section III.B that when the data to be embedded are random in binary form, the

proposed method yields a data hiding rate r whose value is roughly in proportion to the value a of color shifting, i.e., $r = 1$ when $a = \pm 1$; $r = 2$ when $a = \pm 2$; and so on, which are quite high. If the comparison considers the resulting stego-image quality and the data hiding rate simultaneously, then the aforementioned PSNR and hiding rates suggest that the proposed method is superior to most existing methods.

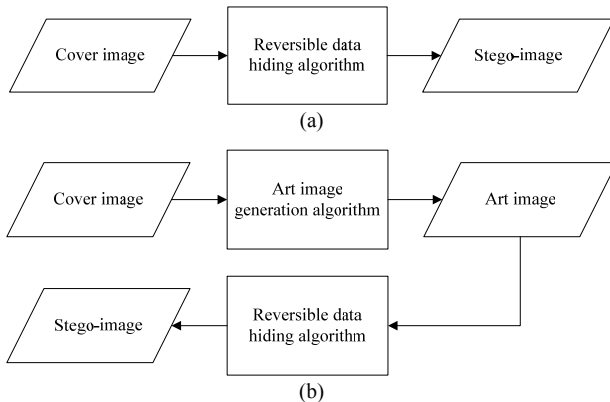


Fig. 15. Comparison of frameworks of conventional and proposed data hiding methods. (a) Framework of conventional data hiding methods. (b) Framework of the proposed method.

Finally, about the computational complexity, the computing load of the proposed method during data hiding mainly comes from Steps 3 and 4 of Algorithm 2, whose complexity may be analyzed to be of the order $O(n)$ where n is the size of the image, because the main tasks in the two steps are just scanning of the pixels in the regions of the art image. This linear complexity says that the computational load of the proposed method is light.

IV. CONCLUSIONS

In this paper, a new method of combining art image generation and data hiding to enhance the camouflage effect for various information hiding applications is proposed. At first, a new type of computer art, called line-based Cubism-like image, and a technique to create it automatically from a source image have been proposed. The method finds line segments in the source image by the Canny edge detection technique and the Hough transform, combines nearby line segments, extends the remaining lines to the image boundaries, and re-color the created regions by their average colors, to create an abstract type of the original source image as the desired art image. Then, by utilizing the characteristics of the Cubism-like image creation process, a data hiding technique has been proposed. Based on the minimum color shiftings of the values of ± 1 , the technique embeds message data into the pixels of the regions of the generated art image while keeping the average region colors unchanged. The data embedding process is proved to be lossless by theorems so that the cover image can be recovered perfectly after the embedded message data are extracted.

The proposed method has several merits. First, it generates Cubism-like images as stego-images to distract the hacker's attention to the message data embedded in them. Also, by using

the minimum color shiftings of ± 1 to embed data bits, the resulting pixels' color differences between the generated Cubism-like image and the stego-image are so small that a hacker will take no notice of the existence of the hidden data. Consequently, the proposed data hiding technique is very suitable for use in covert communication or secret keeping. Furthermore, four measures of randomization of the input message data and the processing order of them with a secret key and several random-number generating functions have been adopted in the proposed method. This enhances greatly the security of the proposed method.

For future studies, about the Cubism-like image creation process, it is interesting to investigate automatic selection of appropriate art images for people. About the use of the proposed data hiding technique, besides covert communication and secret keeping, it may also be tried to conduct image authentication by embedding authentication signals into a generated art image for verification of possible tampering with the image. To increase the data embedding capacity, the histogram modification technique [16] may be adopted for data hiding if the constraint of keeping region colors unchanged can be removed; or simple LSB replacement may be applied as well if the requirement of cover-image reversibility is not needed.

REFERENCES

- [1] A. Hertzmann, "A survey of stroke-based rendering," *IEEE Computer Graphics and Applications*, vol. 23, no. 4, pp. 70-81, July-Aug. 2003.
- [2] A. Hertzmann, "Painterly rendering with curved brush strokes of multiple sizes," *Proc. 1998 Int. Conf. on Computer Graphics & Interactive Techniques (SIGGRAPH 1998)*, Orlando, Florida, USA, pp. 453-460, July 1998.
- [3] A. Hertzmann, "Fast paint texture," *Proc. 2002 Int. Conf. on Computer Graphics & Interactive Techniques (SIGGRAPH 2002)*, Annecy, France, June 3-5, pp. 91-96, 2002.
- [4] M. P. Salisburly, M. T. Wong, J. F. Hughes, and D. H. Salesin, "Orientable textures for image-based pen-and-ink illustration," *Proc. 1997 Int. Conf. on Computer Graphics & Interactive Techniques (SIGGRAPH 1997)*, Los Angeles, California, USA, pp. 401-406, 1997.
- [5] D. Mould, "Stipple placement using distance in a weighted graph," *Proc. Int. Symp. on Computational Aesthetics in Graphics, Visualization & Imaging*, Banff, Alberta, Canada, pp. 45-52, 2007.
- [6] D. Mould, "A stained glass image filter," *Proc. of 14th Eurographics Workshop on Rendering*, Leuven, Belgium, pp. 20-25, 2003.
- [7] A. Hausner, "Simulating decorative mosaics," *Proc. 2001 Int. Conf. on Computer Graphics & Interactive Techniques (SIGGRAPH 01)*, Los Angeles, California, USA, pp. 573-580, August 2001.
- [8] P. Haerberli, "Paint by numbers: abstract image representations," *Proc. 1990 Int. Conf. on Computer Graphics & Interactive Techniques (SIGGRAPH 1990)*, Dallas, Texas, USA, pp. 207-214, 1990.
- [9] Y. Z. Song, P. L. Rosin, P. M. Hall, and J. Collomosse, "Arty shapes," *Proc. Computational Aesthetics in Graphics, Visualization & Imaging*, Lisbon, Portugal, pp. 65-72, 2008.
- [10] C. K. Chan and L. M. Cheng, "Hiding data in images by simple LSB substitution," *Pattern Recog.*, vol. 37, pp. 469-474, March 2004.
- [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 Recog. Letters*, vol. 20, pp. 1511-1517, August 1999.
- [12] J. Fridrich, M. Goljan and R. Du, "Lossless data Embedding—new paradigm in digital watermarking," *EURASIP Journal on Applied Signal Processing*, vol. 2, pp. 185-196, 2002.
- [13] M. Awrangjeb and M. S. Kankanhalli, "Reversible watermarking using a perceptual model," *J. Electron. Imag.*, vol. 14, no. 013014, Mar. 2005.
- [14] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Trans. on Circuits Syst. & Video Technol.*, vol. 13, no. 8, pp. 890-896, Aug. 2003.

- [15] C. de Vleeschouwer, J. F. Delaigle and B. Macq, "Circular interpretation of bijective transformations in lossless watermarking for media asset management," *IEEE Trans. on Multimedia*, vol. 5, no. 1, pp. 97-105, Mar. 2003.
- [16] Z. Ni, Y. Q. Shi, N. Ansari and W. Su, "Reversible Data Hiding," *IEEE Trans. on Circuits Syst. & Video Technol.*, vol. 16, no. 3, pp. 354-362, March 2006.
- [17] 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," *J. of Inform. Sci. & Eng.*, vol. 27, no. 4, pp. 1265-1282, 2011.
- [18] J. Canny, "A computational approach to edge detection," *IEEE Trans. on Pattern Analysis & Machine Intelligence*, vol. 8, no. 6, pp. 679-698, 1986.
- [19] R. C. Gonzalez and R. E. Woods, *Digital image processing*. 2nd ed., Prentice Hall, Upper Saddle River, New Jersey, USA, 2002.
- [20] B. Davis, "Signal rich art: enabling the vision of ubiquitous computing," *Proc. SPIE 7880*, 788002 (2011); doi:10.1117/12.881742.
- [21] H. Otori and S. Kuriyama, "Robust Data Hiding on Texture Images," *Proceedings of Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP '09)*, Kyoto, Japan, pp. 563-566, Sept. 12-14, 2009.
- [22] F. Uccheddu, M. Corsini and M. Barni, "Wavelet-based blind watermarking of 3D models," *Proceedings of 2004 Workshop on Multimedia and Security (MM & Sec '04)*, Magdeburg, Germany, pp. 143-154, Sept. 20-21, 2004.
- [23] A. M. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform," *IEEE Trans. on Image Processing*, vol. 13, no. 8, pp. 1147-1156, 2004.
- [24] S. Lee, C. D. Yoo and T. Kalker, "Reversible image watermarking based on integer-to-integer wavelet transform," *IEEE Trans. Inf. Forensics & Security*, vol. 2, no. 3, pp. 321-330, 2007.

surveillance, and autonomous vehicle applications. He is a Life Member of the Chinese Pattern Recognition and Image Processing Society, Taiwan and a Senior Member of the IEEE.



Shan-Chun Liu received the B.S. degree in computer science from National Chiao Tung University, Taiwan, in 2009 and the M.S. degree in computer science from National Chiao Tung University, Taiwan in 2011. She has been a research assistant at the Computer Vision Laboratory in the Department of Computer Science at National Chiao Tung University from August 2009 to July 2011. Her current research interests include information hiding, image processing, and computer art.



Wen-Hsiang Tsai 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 Chair Professor of Computer Science. At NCTU, he has served as the Head of the Dept. 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 Chinese Image Processing and Pattern

Recognition Society of Taiwan, and from 2004 to 2008, the Chair of the Computer Society of the IEEE Taipei Section 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*, the *International Journal of Pattern Recognition and Artificial Intelligence*, and the *Journal of Information Science and Engineering*. He has published 146 journal papers and 233 conference papers and received many awards, including the Annual Paper Award from the Pattern Recognition Society of the USA; the Academic Award of the Ministry of Education, Taiwan; the Outstanding Research Award of the National Science Council, Taiwan; the ISI Citation Classic Award from Thomson Scientific, and more than 40 other academic paper awards from various academic societies. His current research interests include computer vision, information security, video