IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 2, Issue 2, Apr-May, 2014
ISSN: 2320 – 8791 (Impact Factor: 1.479)
www.ijreat.org

Reversible Data Hiding Using Line Based Cubism Image

Infant Jini N.S[1], Priya.S.V[2]

[1]**M.E. II Year, Department of Computer Science and Engineering, Sriram Engineering College, Perumalpattu – 602 024**

[2]**Associate Professor & HOD, Department of Computer Science and Engineering, Sriram Engineering College, Perumalpattu – 602 024**

Abstract

New types 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 siftings 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.

Keywords: *Computer art image, cover image, line-based Cubism-.*

1. 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 *strokebased 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]. Fig. 2(c)shows a stain-glass image created by an image filter presented in Mould [6].

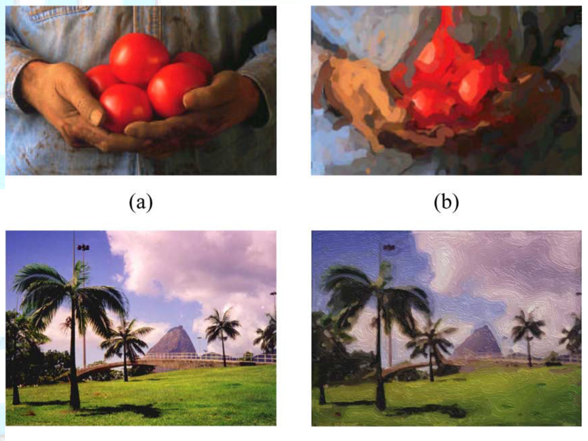


Fig. 1. Art images created by Hertzmann [2], [3]. (a) and (c) Original images .(b) Created art image with watercolor painting effect [2]. (d) Created art image with oil painting effect [3].

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 triangles or rectangles to the regions in segmented images, like the one shown in Fig. 3(c). 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 Regions formed by the prominent lines then are created and the pixels in each region are recolored identically by the average color of the region .Two art images so created in this study are shown in Fig. 5.

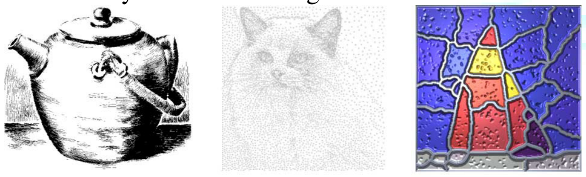

(a) (b) (c)

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

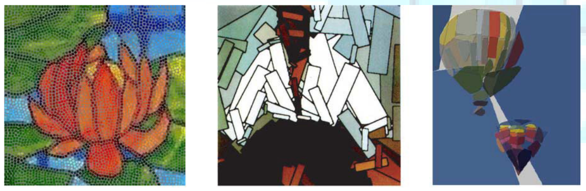

(a) (b) (c)

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

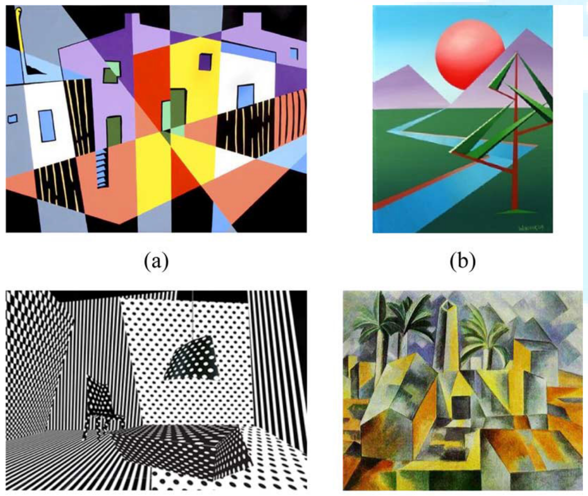

(a) (b)
(c) (d)

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

Data hiding is a technique for watermarking and other applications, which embeds data imperceptibly in to 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

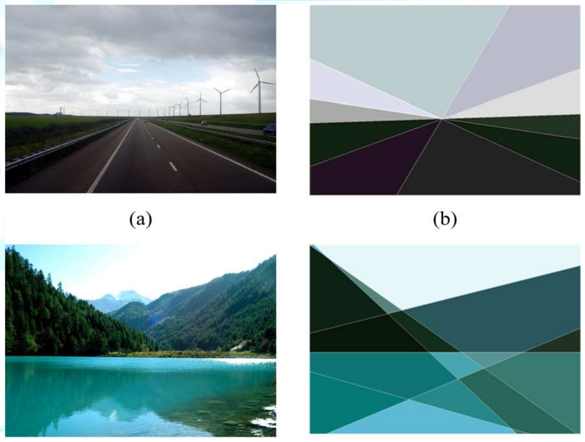

(a) (b)
(c) (d)

Fig. 5. Two examples of line-based Cubism-like images created automatically in this study. (a) and (c) Source images. (b) and (d) Created art images from the source images, respectively.

camouflage effect, the embedded data can be kept securely or transmitted covertly. 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 for data embedding is proposed.

2. Proposed Line-Based Cubism-Like Image Creation Process

2.1 Idea of Line-Based Cubism-Like Image Creation

Cubism artists transform a natural scene into geometric forms in paintings by breaking up, analyzing, and reassembling 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. Specifically, there are two major stages in the proposed line based Cubism-like image generation process—*prominent line extraction* and *region recoloring*. 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.

Algorithm for Line-Based Cubism-Like Image Creation
.

Algorithm 1: ***line-based cubism-like image creation***

Input: a source image , and two thresholds—the minimum line segment length *Lmin*, and the minimum line distance *Dmin* .

Output: a line-based Cubism-like image *Sc* .

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 *L1,L2,……,Lm* 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 *Lmin* and discard the others, resulting in a shorter list of line segments *L1',L2',….,Ln'* .
3.2 For all *i*=0 through *n* and all *j*=0 through *n* with *i*≠*j* and both *Li'* and *Lj'* not deleted yet, compare *Li' Lj'* and if the distance between and is smaller than threshold *Dmin*, then delete the shorter one of *Li'* and *Lj'* .

Stage 2— Region recoloring.

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

Step 5. (*Region Partitioning*) Partition *S″* in to regions *R1,R2,…,Rk* by the extended lines
.
Step 6. (*Region recoloring*) Recolor each region *Ri* in *S″* by the following steps with *i=1,2,…,k.*
6.1 Compute the area *Ai* (in unit of pixel) of *Ri* and the average color (*Cir,Cig,Cib*)of all the pixels in *Ri*.
6.2 Recolor each pixel in *Ri* by (*Cir,Cig,Cib*) .

Step7.(*Line recoloring*) Recolor all region boundaries in *S″* by the white color.

Step 8. Take the final *S″* as the desired line-based Cubism-like image *Sc*.

2.2 Experimental Results

According to the above discussions, we see that different selections of the two threshold values *Lmin* and *Dmin* will result in totally different visual effects in the created art images. 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 *Lmin* and *Dmin*. As a result, each threshold has three choices, resulting in nine choices of the threshold pair. Then, we generate nine art images,

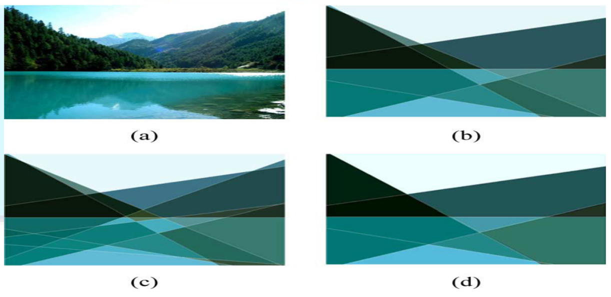


Fig. 6. Experimental results of varying the threshold values of *Dmin* and *Lmin* .(a) Source image with size 1024× 768. (b) Cubism-like image created from (a) with initial values of *Dmin=102*and *Lmin=102* . (c) Cubism-like image created from (a) with *Dmin=102* and *Lmin=20.* (d) Cubism-like image created from (a) with *Dmin=200* and *Lmin=102.*

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), and a seemingly better choice is shown in (l).

3. Data Hiding Via Line-Based Cubism-Like Images

3.1 Idea of Proposed Data Hiding Technique

In the proposed Cubism-like image creation process described by Algorithm 1 above, one major step is to recolor 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.

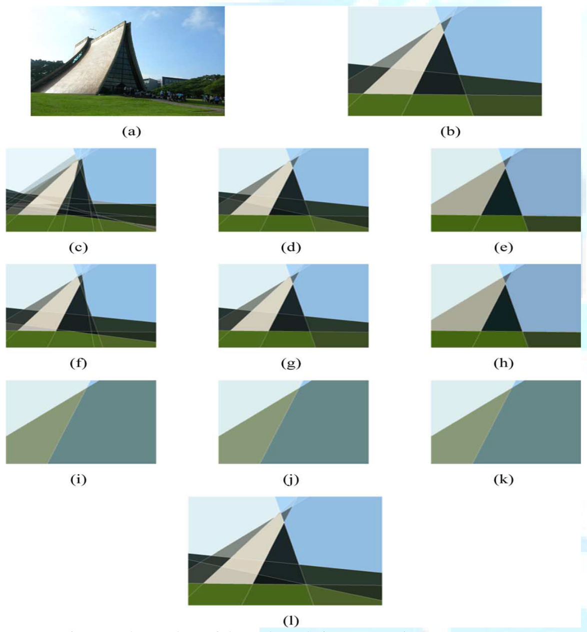


Fig. 7. Experimental results with a church image as input. (a) Source image with size 1024×768. (b) Initial *Dmin=102* and initial *Lmin=102*. (c).*(Dmin,Lmin)=(51,51)* . (d). *(Dmin.Lmin)=(51,102).* (e).*(Dmin.Lmin)=(51,204).* (f).*(Dmin,Lmin)=(102,51).* (g). *(Dmin,Lmin)=(102,102).* (h). *(Dmin.Lmin)=(102,204)* . (i). *(Dmin.Lmin)=(204,51)* . (j). *(Dmin.Lmin)=(204,102)* (k). *(Dmin,Lmin)=(204,204)* . (l) A seemingly better choice of the 9 images—(d).

As a result, people cannot tell the visual difference between the cover image and the stegoimage. 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 recoloring scheme, which keeps the average color of each region unchanged, is designed as a substitute of the original recoloring 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*.

3.2 Principle of Lossless Data Embedding

In the proposed region recoloring 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-of f* in integer computation may be utilized to modify this region recoloring process to keep the average region color unchanged, resulting in a reversible region recoloring process, as proved in the following.

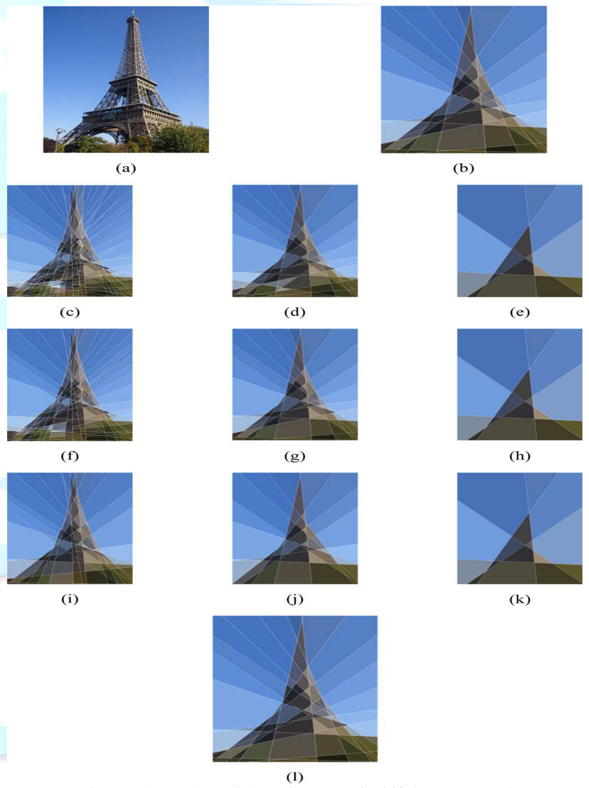


Fig. 8. Experimental results with an image of Eiffel Tower as input. (a) Source image with size 768× 1024. (b) Initial *Dmin*=102 and initial *Lmin*=102. . (c).*(Dmin,Lmin)=(51,51)* . (d) *(Dmin.Lmin)=(51,102).* (e).*(Dmin.Lmin)=(51,204).* (f).*(Dmin,Lmin)=(102,51).* (g).*(Dmin,Lmin)=(102,102).* (h). *(Dmin.Lmin)=(102,204)* . (i). *(Dmin.Lmin)=(204,51)* . (j). *(Dmin.Lmin)=(204,102)* (k). *(Dmin,Lmin)=(204,204)* . (l) A seemingly better choice of the 9 images—(g).

Lemma 1: The total numbers and of data bits of 0's and 1's, respectively, embeddable in an image region with area (in unit of pixel) by recoloring by color shiftings of the amount of without changing the average color of in each of the three color channels is constrained by the following inequality:

$$\frac{-A}{2} \le (N1 - N0) \times a < \frac{A}{2}.$$

Proof: For each of the three color channels R,G , and B,first let *C1,C2,….,CA* be the color values of the A pixels in region R . Then, the average color value *Co* of R is ,

$$C0 = \frac{(C1 + C2 + \cdots . + CA)}{A}$$

Next, while recoloring R by color shifting of the amount of ±a, if the average region color is to be kept unchanged, the computed average color C of R should lie in the range of C0-0.5 to C0+0.5 so that C can be *rounded* to be a value identical to the original average value C0 because the color values in each color channel are *integer numbers*. That is, the following inequality should hold:

$$-0.5 + C0 \le C < C0 + 0.5.$$

Third, if N0 0's and N1 1's are embedded intoR , then there arises a total extra amount of color shiftings .Therefore, the new average region color C is

$$C = \frac{(C1 + C2 \ldots + CA - N0 \times a + N1 \times a)}{A}$$

Combining the three equalities derived above, we can get

$$C0 - 0.5 \le C = \frac{[C0A + (N1 - N0) \times a]}{A} < C0 + 0.5$$

which may be transformed into the following inequality:

$$\frac{-A}{2} \le N1 - N0 < \frac{A}{2}$$

3.3 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 derandomization*. In the first stage, we recover the region recoloring 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 derandomized to get the original message data using the secret
key. The details are described as an algorithm in the following.

Algorithm 2: ***extracting the hidden data string***

Input: a stego-image *S*; and the secret key *Ks* and three random number generators *f1, f2*, and *f3* .
Output: the data string *M* embedded in *S* .
Steps.
Stage 1—Embedded data extraction.
Step 1.Extracting the regions and related parameters
Step 2. Retrieving the region recoloring sequence
Step 3. Create a message data sequence with an empty initial content.
Step 4. Extracting the embedded data unprocessed

Stage 2—Data derandomization.

Step 5. Reorder the digits in using with secret key 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 . Note that in Step 4.4 above,with the help of the ending pattern
digit , 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.

3.4 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 by a hacker is greatly reduced.

4. Conclusion and Future Work

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

TABLE I
RSNR AND MSE VALUES OF STEGO-IMAGES WITH RESPECT TO GENERATED ART IMAGES

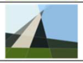
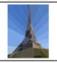
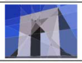
Source image					
Stego-image					
	Fig. 5	Fig. 6	Fig. 7	Fig. 8	Fig. 12
MSE	0.452	0.464	0.483	0.485	0.487
PSNR	51.583	51.287	51.294	51.263	51.254

TABLE II
MSE AND PSNR OF STEGO-IMAGES WITH COLOR SHIFTINGS ±1 THROUGH ±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

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 recolor 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, the technique embeds message data in to pixels of the regions of the generated art image while keeping the average 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 to embed data bits,

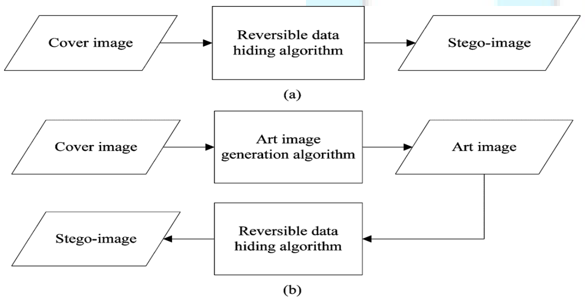

Cover image
Reversible data hiding algorithm
Stego-image
(a)
Cover image
Art image generation algorithm
Art image
Stego-image
Reversible data hiding algorithm
(b)

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.

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 Comput. Graphics Applicat.*, vol. 23, no. 4, pp. 70–81, Jul./Aug. 2003.

[2] A. Hertzmann, "Painterly rendering with curved brush strokes of multiple sizes," in *Proc. 1998 Int. Conf. on Computer Graphics & Interactive Techniques (SIGGRAPH 1998)*, Orlando, FL, Jul. 1998, pp. 453–460.

[3] A. Hertzmann, "Fast paint texture," in *Proc. 2002 Int. Conf. Computer Graphics & Interactive Techniques (SIGGRAPH 2002)*, Annecy, France, Jun. 3–5, 2002, pp. 91–96.

[4] M. P. Salisbury, M. T. Wong, J. F. Hughes, and D. H. Salesin, "Orientable textures for image-based pen-and-ink illustration," in *Proc. 1997 Int. Conf. Computer Graphics & Interactive Techniques (SIGGRAPH 1997)*, Los Angeles, CA, 1997, pp. 401–406.

[5] D. Mould, "Stipple placement using distance in a weighted graph," in *Proc. Int. Symp. Computational Aesthetics in Graphics, Visualization & Imaging*, Banff, Alberta, Canada, 2007, pp. 45–52.

[6] D. Mould, “A stained glass image filter,” in *Proc. 14th Eurographics Workshop on Rendering*, Leuven, Belgium, 2003, pp. 20–25.

[7] A. Hausner, “Simulating decorative mosaics,” in *Proc. 2001 Int. Conf. Computer Graphics & Interactive Techniques (SIGGRAPH 01)*, LosAngeles, CA, Aug. 2001, pp. 573–580.

[8] P. Haeberli, “Paint by numbers: Abstract image representations,” in *Proc. 1990 Int. Conf. Computer Graphics & Interactive Techniques (SIGGRAPH 1990)*, Dallas, TX, 1990, pp. 207–214.

[9] Y. Z. Song, P. L. Rosin, P. M. Hall, and J. Collomosse, “Arty shapes,” in *Proc. Computational Aesthetics in Graphics, Visualization & Imaging*, Lisbon, Portugal, 2008, pp. 65–72.

[10] C. K. Chan and L. M. Cheng, “Hiding data in images by simple LSB substitution,” *Pattern Recog.*, vol. 37, pp. 469–474, Mar. 2004