JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 27, 983-994 (2011)

Data Hiding in Graphic Drawings by Structures of Object Groupings*

TSUNG-YUAN LIU[1] AND WEN-HSIANG TSAI[1,2]
[1]*Department of Computer Science*
National Chiao Tung University
Hsinchu, 300 Taiwan
[2]*Department of Information Communication*
Asia University
Taichung, 413 Taiwan

A new data hiding method is proposed for embedding a bit-string message into the structure of object groupings in a drawing. The objects in a drawing are grouped skillfully according to the message data to be embedded as well as the inter-object distance relationships between the objects. The groupings of objects in the resulting stego-drawing are visually imperceptible and robust against translation, scaling, rotation, and mirroring attacks. Possibilities of embedding variable-length message data into drawings are demonstrated for different data hiding applications, including data authentication, copyright protection, and covert communication. The method can be applied to a variety of graphic drawings, including flowcharts, network diagrams, circuit schematics, floor plans, *etc*. Good results obtained from experiments conducted on Microsoft Visio drawings are shown to confirm the feasibility of the proposed method.

Keywords: data hiding, vector graphics, object grouping, data authentication, copyright protection, covert communication, Microsoft Visio

1. INTRODUCTION

Data hiding is a technique that embeds information imperceptibly into given media for various purposes, including covert communication, copyright protection, data authentication, *etc*. Imperceptibility of hidden data is commonly achieved by exploiting the weaknesses of the human auditory and visual systems, using the techniques of, for example, changing the least significant bits of the pixels of a cover image to embed information [1], or shifting lines, words, or characters by a small amount in an image containing texts [2]. Most prior researches concentrated on images, audios, and videos as cover media [3-7].

Data hiding in vector drawings is comparatively less studied, due to the relatively low information content in such a kind of media that can be manipulated. Vector drawings usually contain lines, polygons, or objects of different shapes with uniform or gradient fills. Techniques developed for hiding data in bitmap images usually manipulate the luminance variations of images to embed information, and thus are unsuitable for vector drawings.

Data hiding in a vector drawing is most commonly achieved by altering the geometry of the shapes in the drawing to embed data. The manipulation of the shape geometry

Received September 16, 2009; revised December 22, 2009; accepted January 19, 2010.
Communicated by H. Y. Mark Liao.
* This work was supported partially by the National Science Council of Taiwan, project No. 97-2631-H-009-001.

can be done in the spatial domain or in one of the transform domains such as DFT, DWT, and DCT [8-14]. Kwon *et al*. [9] embedded invisible watermark signals into lines, arcs, and circles in a CAD drawing by modifying their lengths, angles, and radii, respectively. Detection of the watermark, however, requires the use of the original drawing. Solachidis and Pitas [10] achieved blind watermark detection by modifying the coordinates of the vertices in a polygonal line using Fourier descriptors. The embedded watermark is resilient to scaling, rotation, and translation attacks, but vulnerable to distortion attacks. The method was later enhanced by Doncel *et al*. [11]. Im *et al*. [12] proposed the use of wavelet descriptors for embedding watermarks that are robust against global and local geometrical distortions.

The above methods, however, are unsuitable for drawings such as flowcharts, network diagrams, circuit schematics, floor plans, *etc*. These diagrams are composed of objects that come from a *stencil*, and the contours of the individual objects in such diagrams cannot be altered. Fig. 1 shows an example of a floor plan drawing, where the shapes representing desks, chairs, servers, walls, doors, *etc*. all come from standard stencils, and cannot be individually altered.

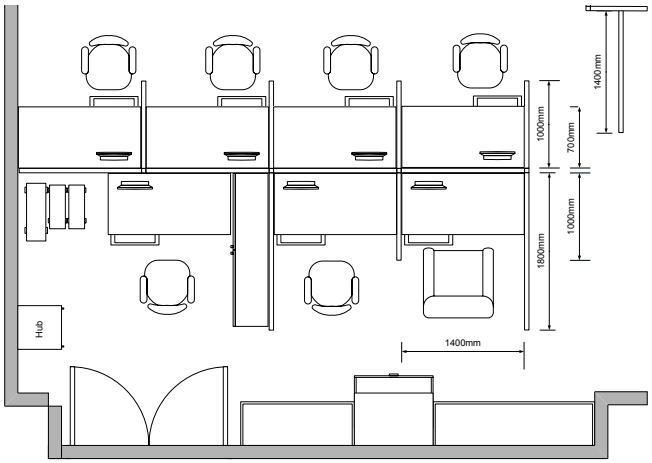

Fig. 1. A floor plan diagram of an office composed of different objects from stencils.

A drawing of this kind can contain numerous objects, and for ease of manipulation, drawing editing applications allow objects to be grouped together such that each group can be manipulated as a unit. Each group can then be translated, scaled, rotated, mirrored, or colored as a whole. It is also possible for the groupings to be nested, that is, a group can contain several groups of objects.

In this paper a different approach to embedding data into drawings is proposed by manipulating the structure of object groupings. Such a structure can be represented by a tree, as illustrated in Fig. 2, where the internal nodes are the groups, and the leaves are the objects. In the figure, for example, Group 4 contains two simple objects while Group 1 includes two simple objects as well as two smaller groups. It is the use of such different combinations of the object groupings that fulfills information embedding in the proposed approach.

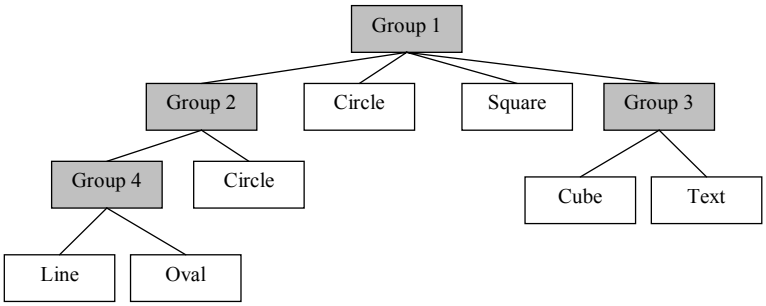

Fig. 2. Illustration of object groupings for data embedding in a drawing.

Compared to the previously mentioned data hiding techniques, the proposed method has several merits.

1. The method can be used to embed multiple data bits with a blind extraction capability, allowing for several different types of data hiding applications, whereas some techniques require the use of the original media (called *non-blind* methods) or can only embed watermarks with no message data (called *zero-bit watermarking* methods) [8, 11].
2. Manipulations of object groupings do not change the visual appearance of a drawing at all, whereas most other techniques degrade the quality of a drawing [8-13].
3. Any collection of shapes, lines, or text blocks that can be grouped together may be used by the proposed method for data embedding, whereas many other techniques can only be applied to specific drawing objects such as polylines, polygons, B-spline curves, *etc*. [8-10, 13, 14].
4. The method can be used to embed information in any graphical file format that supports nested grouping of objects, for example, AutoCAD drawings, Visio drawings, and PowerPoint presentation files.

In the remainder of this paper, the detail of the proposed data hiding method using structures of object groupings is described in section 2. Possible data hiding applications using the proposed method are then described in section 3. Experimental results that demonstrate the feasibility of the proposed method are presented in section 4, and conclusions with some suggestions for future works are included in section 5.

2. DATA HIDING USING STRUCTURES OF OBJECT GROUPINGS

In the proposed method, objects in a drawing are grouped in a certain way to embed data, and the grouping structure is examined for data extraction. As the grouping of objects can be nested, for simplicity we will refer to a simple object or a group of objects both as an *object*. That is, a drawing is composed of a collection of objects, where some objects are composed of smaller constituent objects. The details of the proposed data embedding and extraction processes are described in the sequel.

2.1 Data Embedding

The basic idea of the proposed data embedding process is to determine the inter-object distances for all objects in the drawing, sort the distances between pairs of objects, and then group the pairs of objects in turn or left them ungrouped, depending on the bits to be embedded. The detail is described as an algorithm below.

Algorithm 1: Data embedding by structure of object groupings.
Input: A drawing with objects $D = \{o_1, o_2, \ldots, o_L\}$ and a bit string $S = b_1b_2\ldots b_N$ to be embedded.
Output: A stego-drawing D' with an appropriate structure of object groupings representing S.
Steps:
1. Set D' to be equal to D initially.
2. Create an auxiliary *helper set* P and set it to be empty initially.
3. Take in order a bit b_i from the input bit string S, where $1 \le i \le N$, and perform the following steps.
   (a) Calculate the distances between every pair of objects in D' that are *not* in P.
   (b) Find the two objects o_j and o_k in D' such that their distance is the smallest among those of all object pairs in D' that are *not* in P.
   (c) If $b_i = 1$, then group o_j and o_k in D' together; otherwise, add the pair (o_j, o_k) to P as an ungrouped one.
4. Take the final D' with the resulting structure of object groupings as the output stego-drawing.

The purpose of creating the set P in the above algorithm is to record pairs of objects that are not grouped together for embedding 0's, so that these object pairs are not considered further. As an example, supposed that we want to embed the bits 1010010011 into the drawing shown in Fig. 3 with the inter-object distances listed in Table 1. We find that the objects "File server" and "Application server" are the closest two objects (with distance 0.2791), and is so the first pair considered. Since the first bit to be embedded is 1, the two objects are grouped together to form a new object, called Group 1, according to Step 3c of the algorithm. The closest object pair in the resulting drawing is then Group 1 and the object "Switch 1" (with distance 0.4134). These two objects are *not* grouped in order to embed a "0," and the pair is recorded in the set P so that they are not considered further for grouping. The objects "Switch 2" and "Workstations" (with distance 0.4977) are then considered, and grouped to form Group 2 to embed a "1," and so on. The resulting structure of object groupings after embedding 1010010011 is shown in Fig. 4.

Table 1. Distances between all pairs of objects in Fig. 3.

	Workstations	Router	Switch 2	Switch 1	File Server
Application Server	0.9139	1.7557	0.9903	0.4134	0.2791
File Server	1.9525	2.1690	1.7464	0.8552	
Switch 1	1.5705	1.0851	0.5998		
Switch 2	0.4977	0.7566			
Router	1.9052				

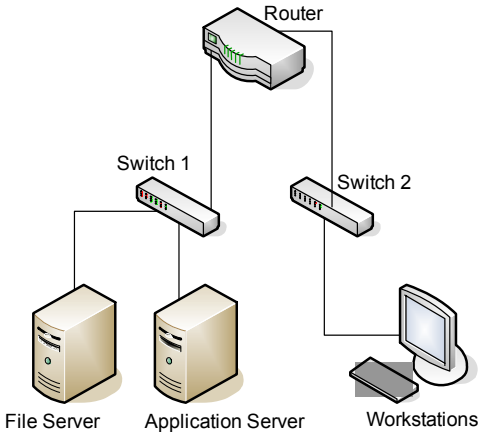

Fig. 3. A simple drawing used as an example for embedding by object grouping.

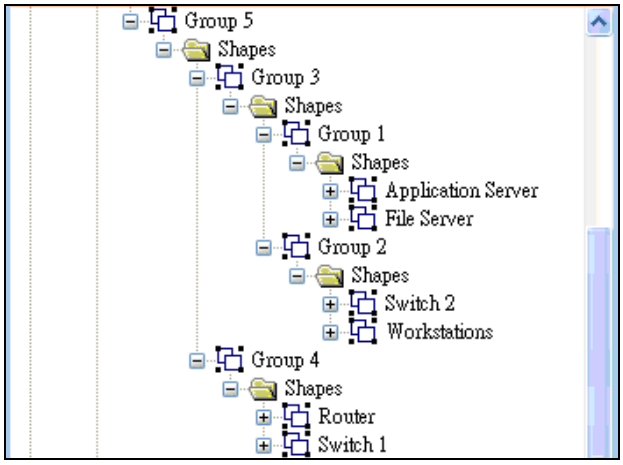

Fig. 4. Resulting structure of object groupings of Fig. 3 after embedding 1010010011.

2.2 Embedding Capacity

In Algorithm 1, when two objects are grouped into one for every bit of 1 embedded (step 3c), the number of objects in the stego-drawing D' decreases by one. The maximum number of 1's that can be embedded using the proposed method is thus $L - 1$ where L is the number of objects in the input drawing D.

On the other hand, the method can embed 0's more efficiently. Specifically, it can be seen from the following analysis of performing the steps of Algorithm 1 that for a drawing with L objects, the maximum number of bits that can be embedded is $(L - 1)^2$ when the input data string is of the form:

$$\underbrace{00\ldots0}_{\frac{L(L-1)}{2}-1}1\underbrace{00\ldots0}_{(L-2)-1}1\underbrace{00\ldots0}_{(L-3)-1}1\ldots1\underbrace{00}_{3-1}1\ \underbrace{0}_{2-1}\ 11. \tag{1}$$

Performance analysis of Algorithm 1 for embedding the maximum number of bits.

1. All object pairs except "the pair G_1 with the farthest *in-pair distance*" are ungrouped and added to the set P to embed the first $[L(L-1)/2-1]$ 0's, where the term "in-pair distance" means the distance between the two objects in an object pair.
2. G_1 is taken as Group 1 to embed the first "1" in the data string, which results in $L-2$ new distance relationships between Group 1 and the remaining $L-2$ objects.
3. All object pairs, each with Group 1 and an object of the remaining $L-2$ ones, except the pair G_2 with the farthest in-pair distance are ungrouped and added to the set P to embed the next $[(L-2)-1]$ 0's.
4. G_2 is taken as Group 2 to embed the second "1" in the data string, resulting in $L-3$ new distance relationships.
5. Steps 3 and 4 above are repeated in a similar way for the remaining objects and groups until no more objects can be considered for grouping.

Accordingly, the maximum number of bits that can be embedded is

$$\begin{aligned}
&\{[L(L-1)/2-1]+1\}+\{[(L-2)-1]+1\}+\{[(L-3)-1]+1\}+\ldots+\{[2-1]+1\}+1\\
&=L(L-1)/2+(L-2)+(L-3)+\ldots+1\\
&=L(L-1)/2+(L-1)(L-2)/2\\
&=(L-1)^2,
\end{aligned}$$

as mentioned previously.

On the other hand, the expected number of random bits (equal occurrence probabilities of 0's and 1's) that can be embedded by the proposed method using a drawing with L objects is smaller than $2(L-1)$, as discussed now.

First, as mentioned previously, at most $(L-1)$ 1's can be embedded for a drawing with L objects. Also, from the above performance analysis of the proposed algorithm for embedding the maximum number of bits, we see that at least one "0" can be embedded for each "1" embedded except the last one. If the input string includes 0's and 1's alternatively as in the extremely random case, then exactly $2(L-1)-1$ bits can be embedded. In real cases, 1's may appear *consecutively*, such that objects in a drawing will be exhausted *faster* than appropriate numbers of 0's are embedded, resulting in less 0's being embedded when compared with the extremely random case. In short, for the average case, the number of random bits that can be embedded with a drawing containing L objects is roughly $2(L-1)-1 \approx 2L$ if L is large enough.

2.3 Data Extraction

The process for extracting the data embedded in a stego-drawing using Algorithm 1 is described below. Basically, the algorithm first removes the structure of object groupings to recover the original cover drawing. It then uses the same procedure as that of data embedding to gradually reconstruct the same structure of object groupings, and checks in the meantime the object grouping structure in the stego-drawing to determine the previously embedded bits one by one.

Algorithm 2: Data extraction from structure of object groupings.
Input: A stego-drawing D' with a certain structure of object groupings generated by Algorithm 1.
Output: A bit string $S = b_1 b_2 \ldots b_N$ extracted from the structure of object groupings in D'.
Steps:
1. Ungroup all existing object groups in D' to recover the original drawing $D = \{o_1, o_2, \ldots, o_L\}$.
2. Create an auxiliary *helper set* P and set it to be empty initially.
3. Initialize an empty bit string S.
4. Extract a bit b_i and append it to the end of S, where $1 \le i \le N$, by performing the following steps.
   (a) Calculate the distances between every pair of objects in D that are *not* in P.
   (b) Find the two objects o_j and o_k in D such that their distance is the smallest among those of all object pairs in D that are *not* in P.
   (c) Check if the object pair (o_j, o_k) is a group in the grouping structure of the original stego-drawing D': if so, then set b_i to be 1 and take (o_j, o_k) as a group in D; otherwise, set b_i to be 0 and add the pair (o_j, o_k) to P as an ungrouped one.
5. Take the final S as the desired output bit string.

In the above algorithm, it is assumed that the number of bits N in the embedded bit string is either previously known or determinable during data extraction in a certain way, like prefixing the data bit string S with a fixed-length bit segment that contains the value N.

3. DATA HIDING APPLICATIONS

The proposed method uses relative inter-object distances as the basis for forming the structure of object groupings. It can be figured out that the technique is robust to attacks such as translation, scaling, rotation, and mirroring of the drawing as a whole since these transformations do not alter the inter-object distances. Based on this property, we describe below some possible data hiding applications using the proposed method.

3.1 Data Authentication

The proposed method can be used for authenticating a drawing by embedding a random-key controlled bit string with $(L - 1)$ 1's as an authentication signal such that all objects in the drawing are grouped together, forming a structure of mutually-validating object groups. If an attacker attempts to modify the stego-drawing by changing some of the object groupings, moving some of the objects around, or adding or removing one or more objects, then some of the inter-object distances in the drawing will be changed. The data extracted from such a drawing will thus be different from the expected embedded authentication signal and tampering with the drawing be detected.

It is noted that the above method by itself cannot detect attacks where an object is replaced by another with the same dimension, or where an object's internal properties (such as color and caption) are modified. For such cases, the proposed method can be combined with other authentication mechanisms which cover the objects' properties. For

example, one simple technique is to create an encrypted hash value [15] from all the objects' properties and then embed this hash value as an authentication signal using the proposed algorithms, resulting in the combined method of object-grouping and object-property encryption techniques. Tampering with an object's property in a stego-drawing then can be detected since the authentication signal extracted from such a tampered drawing using the proposed algorithms will be different from that re-calculated directly from the changed objects' properties.

3.2 Copyright Protection

The proposed method can be used to embed an invisible watermark into a drawing for the copyright protection purpose, where the watermark is taken to be in the form of a bit string. Such a watermarking scheme, as mentioned previously, is robust to translation, scaling, rotation, and mirroring operations of the drawing, which are possible attacks during a copyright infringement activity. In addition, as mentioned in section 3.1, a watermark so created can survive alterations to object property modifications such as changing the colors or captions of objects, which are also possible attacks in copyright infringement.

Although the watermark can be removed by changing the structure of object groupings in the drawing, for ease of manipulations, copyright offenders typically do not perform grouping/ungrouping operations unless necessary. On the contrary, to avoid unintentional operations of ungrouping all objects to destroy the watermark completely, it is recommended to create a watermark signal with a short length (with less than $(L - 1)$ 1's) such that the watermark signal covers object groupings within short distances only. In this way, the watermark signal will still be intact if a copyright offender removes peripheral objects such as drawing captions and comments.

3.3 Covert Communication

Since communications via drawings such as flowcharts are common and the proposed data hiding procedure does not affect the visual appearance of the drawing, the method is suitable for the covert communication purpose as well. Specifically, a secret message in the form of a bit string can be embedded into a cover drawing imperceptibly using the proposed algorithms. Also, grouping of objects for object manipulations in a drawing is a subjective choice by the author and usually does not follow a predictable rule, making automatic steganalysis difficult.

To face the common assumption made in covert communication that the data hiding algorithms used are known to the public, it is suggested to randomize the secret message in advance by some symmetric encryption algorithm such as AES [16] before it is taken as input to the proposed data embedding process.

4. EXPERIMENTAL RESULTS

A series of experiments were conducted on some drawings created with Microsoft Visio 2003, a popular package for drawing flowcharts, network diagrams, floor plans, *etc*.

Two examples of them are shown in Figs. 1 and 5. The proposed method can be conveniently implemented in Visio, which supports nested grouping of objects as well as functionality that allows distances between any two objects to be computed [17]. Three different types of drawings as listed in Table 2 were tested to demonstrate the generic applicability of the proposed method. The table lists the number of objects available for grouping in each drawing, and the average number of random bits embeddable over ten independent trials. The average number of bits embeddable is approximately twice the number of objects in the drawing, which matches the theoretical prediction mentioned in section 2.2.

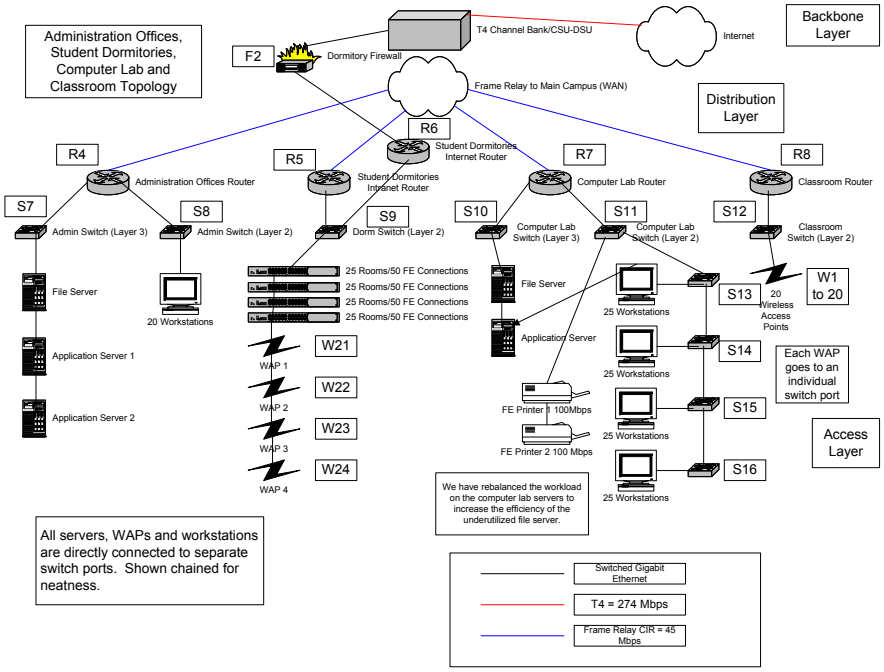

Fig. 5. A network layout diagram used in the experiments (source: UCF).

Table 2. Experimental results of embedding capacity for different drawings.

	Type of drawing	Number of objects	Bits embeddable
A	Network topology	113	235.0
B	Office layout	78	156.4
C	Flowchart	44	82.0

The stego-drawings were then attacked by translation, scaling, and rotation operations in the experiments. Specifically, we translated, scaled, and/or rotated all the objects in the drawing simultaneously in the attacks and applied the algorithms subsequently. The results show that the bits embedded in the object grouping structures survive these attacks.

To test the relation between the number of grouped objects and detectability of ob-

ject removal attacks discussed in sections 3.1 and 3.2, we conducted some additional experiments where bit strings of varying lengths were embedded into the test drawings and ratios of detected attacks were computed. Specifically, after embedding a bit string into a drawing by Algorithm 1, *one* of the objects was removed from the drawing, and the bit string extracted by Algorithm 2 was checked against the original one to see whether the object removal was detected. The results are summarized in Table 3, where (1) the *consumed data-hiding capacity* is defined as the length of the embedded bit string over the approximate average data-hiding capacity of the drawing (the value $2L$ mentioned at the end of section 2.2); and (2) the *object-removal detection rate* is defined as the number of successful detections of object removals over the total number of objects in the drawing.

Table 3. Experimental results of tampering detection.

	Consumed data-hiding capacity	Object-removal detection rate
A	32%	71/113
B	47%	51/78
C	84%	44/44

It can be seen that the detection rate is higher when the capacity is highly utilized, as expected in the discussion in section 3. With high capacity consumptions, removal of any object will alter the data bits embedded in the drawing. On the other hand, with low capacity consumptions, some peripheral objects such as drawing captions can be removed without affecting the embedded data. This property is useful for embedding signals of different strengths for different data hiding applications, matching the concepts revealed by the discussions made in section 3.

5. CONCLUSIONS AND FUTURE WORKS

In this paper, a new data hiding method has been proposed, which embeds message data imperceptibly into the structure of object groupings in a drawing, in contrast with prior works that alter objects themselves for data hiding applications. The proposed method is generic and can be applied to a variety of drawings, including flowcharts, network diagrams, circuit schematics, floor plans, *etc*. The method creates a structure of object groupings based on the data to be embedded as well as inter-object distances in the drawing, yielding a stego-drawing that is robust against attacks such as translation, scaling, rotation, and mirroring operations. By varying the amount of data embedded into a drawing, the proposed method has been shown to be applicable to several different data hiding applications, including data authentication, copyright protection, and covert communication.

It has been shown that the amount of data embeddable by the proposed method is on average twice the number of objects in the drawing. One future work may be directed to investigating alternative encodings that use more bits of 0's so that larger amounts of data can be embedded in cover drawings. Another is to investigate other variations of structures of object groupings which can provide larger data embedding capacities. Other applications of the proposed method can also be studied.

REFERENCES

1. D. C. Wu and W. H. Tsai, "A steganographic method for images by pixel-value diferencing," *Pattern Recognition Letters*, Vol. 24, 2003, pp. 1613-1626.
2. J. T. Brassil and N. F. Maxemchuk, "Copyright protection for the electronic distribution of text documents," *Proceedings of the IEEE*, Vol. 87, 1999, pp. 1181-1196.
3. W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *I.B.M. Systems Journal*, Vol. 35, 1996, pp. 313-336.
4. F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding – A survey," *Proceedings of the IEEE*, Vol. 87, 1999, pp. 1062-1078.
5. M. Wu, H. Yu, and A. Gelman, "Multi-level data hiding for digital image and video," *Proceedings of SPIE*, Vol. 3845, 1999, pp. 10-21.
6. P. C. Su and C. C. Kuo, "Steganography in JPEG2000 compressed images," *IEEE Transactions on Consumer Electronics*, Vol. 49, 2003, pp. 824-832.
7. R. Chandramouli, M. Kharrazi, and N. Memon, "Image steganography and steganalysis concepts and practice," *Digital Watermarking*, LNCS 2939, 2004, pp. 35-49.
8. X. Niu, C. Shao, and X. Wang, "A survey of digital vector map watermarking," *International Journal of Innovative Computing*, *Information and Control*, Vol. 2, 2006, pp. 1301-1316.
9. K. R. Kwon, B. J. Jang, E. J. Lee, and Y. Huh, "Copyright protection of architectural CAD drawing using the multiple watermarking scheme," in *Proceedings of IEEE International Conference on Multimedia and Expo.*, 2004, pp. 871-874.
10. V. Solachidis and I. Pitas, "Watermarking polygonal lines using fourier descriptors," *IEEE Computer Graphics and Applications*, Vol. 24, 2004, pp. 44-51.
11. V. R. Doncel, N. Nikolaidis, and I. Pitas, "An optimal detector structure for the fourier descriptors domain watermarking of 2D vector graphics," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 13, 2007, pp. 851-863.
12. D. H. Im, H. Y. Lee, S. J. Ryu, and H. K. Lee, "Vector watermarking robust to both global and local geometrical distortions," *IEEE Signal Processing Letters*, Vol. 15, 2008, pp. 789-792.
13. A. Giannoula, N. Nikolaidis, and I. Pitas, "Watermarking of sets of polygonal lines using fusion techniques," in *Proceedings of IEEE International Conference on Multimedia and Expo*, 2002, pp. 26-29.
14. R. Ohbuchi, "A shape-preserving data embedding algorithm for NURBS curves and surfaces," in *Proceedings of the Computer Graphics International*, 1999, pp. 177-180.
15. H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-hashing for message authentication," Internet RFC 2104, 1997.
16. Advanced Encryption Standard (AES), FIPS Pub 197, November 2001, http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf.
17. Microsoft Corporation, Visio 2003 SDK documentation, MSDN Library, Office Developer Center, Visio 2003, http://msdn.microsoft.com/en-us/library/bb421578(office.11).aspx.

Tsung-Yuan Liu (劉宗原) received his B.S. degree in Electrical Engineering from the University of the Witwatersrand, Johannesburg, South Africa, the M.B.A. degree from National Taiwan University, Taipei, Taiwan, R.O.C., and is currently pursuing the Ph.D. degree at the College of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, R.O.C.

He is a Software Engineer of Google, Taipei, Taiwan. His research interests include information hiding, image processing, web search, data mining, and artificial intelligence.

Wen-Hsiang Tsai (蔡文祥) received the B.S. degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan, R.O.C., the M.S. degree from Brown University, Providence, RI, and the Ph.D. degree from Purdue University, West Lafayette, IN.

Currently, he is a Chair Professor with the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, R.O.C., and was the President of Asia University, Taichung, Taiwan. So far he has published 135 journal papers and 220 conference papers. His research interests include image processing, computer vision, information security, and autonomous vehicle applications.