To be published in the proceedings of the *IEEE International Conference on Multimedia and Expo 2002 (ICME 2002),* Lausanne, Switzerland, Aug. 26-29, 2002

ROBUST WATERMARKING OF VECTOR DIGITAL MAPS

[1]Ryutarou Ohbuchi, [1]Hiroo Ueda, [2]Shuh Endoh

ohbuchi@acm.org, k7026@kki.yamanashi.ac.jp, endoshu@jp.ibm.com

[1]Department of Computing Science, Yamanashi University
4-3-11 Takeda, Kofu-shi, Yamanashi-ken, 400-8511, Japan.

[2] GIS Business Promotion, IBM Japan.

ABSTRACT

Digital maps are used, for example, in car navigation systems and web-based map services. As digital data, digital maps are easy to update, duplicate, and distribute. At the same time, illegal duplication and distribution or forgery of the maps is also easy. This paper proposes a digital watermarking algorithm for vector digital maps as a method to counter such abuses of the maps. A watermark bit is embedded by displacing an average of coordinates of a set of vertices that lies in a rectangular area created on a map by adaptively subdividing the map. The watermark is resistant against additive random noise, similarity transformation, and vertex insertion/removal, and, to some extent, cropping.

1. INTRODUCTION

Digital maps of various kinds have become quite popular in the last few years. They are used, for example, in car navigation systems, geographical information systems (GIS), and in web-based map services. As a digital data, digital maps are easy to update, duplicate, and distribute. They are also prone to forgery, illegal duplication and illegal distribution. *Digital watermarking* is a possible approach to counter such abuses. Digital watermark adds structure called watermark to the target data object imperceptibly and inseparably. The information encoded in the watermark can be used to identify the copyright owner or to detect tampering.

Most of the two-dimensional (2D) digital maps can be classified into either raster- or vector-digital maps. A raster digital map represents a map as raster image data, i.e., an image represented by a 2D array of pixels. As images, most of the watermarking algorithm developed for digital images can be applied to the raster digital maps. (For image watermarking techniques, see, for example, [2].) Vector digital maps employ geometrical primitives such as points, lines, polylines, and polygons to represent objects in the maps, such as building outlines, roads, rivers, and contour lines. Unlike the raster digital maps, the vector digital maps have an advantage of being able to be scaled and rotated without loss of quality.

In a vector digital map, connectivity among objects, e.g., vertices, is specified explicitly, and the distance between vertices are irregular. Consequently, standard techniques such as Fourier or wavelet transformation used in image watermarking algorithms can't be applied readily. Probably due to these difficulties, we know of only a few published works on watermarking vector digital maps [4, 5, 3]. Kurihara et al [4] encoded information into individual vertex coordinate, and their watermarks are not resilient against additive random noise. Endoh et al [5, 3] reported nearly a dozen algorithms to watermark vector digital maps. These methods targeted either vertex coordinate or vertex connectivity for watermarking.

This paper presents a *robust*, *informed-detection* watermarking algorithm for *vector* digital maps. The method embeds a bit by minutely displacing a group of vertices contained in a rectangle, and the bit is embedded repeatedly over the map. Averaging the displacement values among multiple vertices in the rectangle and among multiple embedding, combined with other techniques, makes the method resilient against (1) random noise added to the vertex coordinates, (2) insertion and deletion of vertices, (3) similarity transformation, (4) scrambling of order of geometric primitives in a data file, and, to some extent, (5) cropping.

The rest of the paper is structured as follows. In the next section, we present our watermarking algorithm. We then describe the results of evaluation experiments in Section 3, followed by a summary and conclusion in Section 4.

2. THE ALGORITHM

Our watermarking algorithm embeds a bit by displacing a group of vertices in a rectangular area. The rectangular area is computed adaptively to the density of vertices in the map, using a modified quadtree subdivision. A traversal of the subdivision hierarchy gives the order of embedded bits. The watermark gains resiliency to attacks by moving multiple vertices per rectangle and by repeatedly embedding a bit over the map. As an informed detection algorithm, watermarks are extracted by comparing the *reference map* (the map before watermarking, which may be escrowed) with the watermarked (and possibly attacked) map. The two maps are first registered by using an iterative optimization process to remove similarity transformation applied. The rectangular area subdivision used during the embedding is re-created on the reference and watermarked map. By comparing the coordinates of the vertices in the rectangles, and by re-creating the ordering, the embedded message bit string can be recovered.

2.1. Embedding

The embedding starts with the area subdivision step, which tries to produce rectangles (1) having more than d vertices per rectangle, and (2) the numbers of vertices in the rectangles are close to each other. The value of d is determined under a tradeoff. A large d increases resiliency of the bit embedded in the rectangle due to noise averaging. However, a smaller number of rectangles are available overall so that the information payload is decreased. The smaller number of rectangles also means less chip rate c leading to watermarks less resilient against resection, additive random noise, and other attacks.

We implemented and compared three area subdivision methods; (1) *Uniform* (UNIF), (2) *Quadtree* (QUAD), and (3) *Modified quadtree* (MQUAD). Each method subdivides a given map as described below.

(1) *Uniform* (UNIF): Subdivide the map into $k \times l$ uniform size rectangular sub-areas.
(2) *Quadtree* (QUAD): Subdivide the map adaptively according to the area-quadtree algorithm so that every rectangle contain more than d vertices.
(3) *Modified quadtree* (MQUAD): Subdivide the map adaptively as with the QUAD method. If a subdivision created a rectangle containing less than d vertices, the rectangle is merged with an adjacent rectangle at the same subdivision level. If there were more than one candidate for the pairing, the one with the smallest number of vertices is chosen.

For the UNIF subdivision, an ordering of rectangles is created by a simple raster scan of the rectangles. For the QUAD and MQUAD methods, the ordering is introduced by using a depth-first traversal of the tree-structured hierarchy of the subdivision.

After the subdivision, a watermark is embedded into the map by displacing the vertices in the rectangles using a simple modulation method similar to Hartung's [1]. The data to be embedded is an m-dimensional bit vector $\mathbf{a} = (a_1, a_2, ..., a_m)$ in which each bit takes values $\{0,1\}$. Each bit a_j is spread spatially over the map by duplicating each symbol by *chip rate* c, producing a watermark symbol vector $\mathbf{b} = (b_1, b_2, ...b_{mc})$, $b_i \in \{0,1\}$ of length $m \cdot c$. Repeatedly embedding the same bit c times increases resiliency of the watermark against additive random noise. Also, averaging the detected signal by d times upon watermark extraction reduces the effect of the additive random noise. If there are L rectangles that are usable for watermarking, *i.e.*, if they contain more than the predetermined d vertices, then $c = floor(L/n)$, where n is the number of bits of the message. Each element b_i of the symbol vector $\mathbf{b}$ is then repeated or *spread c* times;

$$b_i = a_j, \quad j \cdot c \le i < (j+1) \cdot c \tag{5}$$

We employed two spreading methods, the (1) *symbol repeating method* and (2) *message repeating method*. The symbol repeating repeats a bit next to each other in a series of ordered rectangles. The rectangle traversal method for the bit ordering clusters the modified rectangles spatially nearby. The message repeating repeats the entire message bit string, instead of each bit. This method spatially spreads the rectangles modified for each bit.

After the spreading, the bit vector $\mathbf{b}_i$ is converted to an embedding symbol vector $\mathbf{b}' = (b'_1, b'_2, ...b'_{mc})$ $b'_i \in \{-1,1\}$ by the following mapping to create a zero-mean signal;

$$b'_i = \begin{cases} -1, & \text{if } b_i = 0; \\ 1, & \text{if } b_i = 1. \end{cases} \tag{6}$$

Assume that there are L usable rectangles and that ith rectangle contains M vertices. Let $v_{i,m}$ be the coordinate of mth vertex $(1 < m < M)$ in the ith rectangle prior to the watermarking, $p_i \in \{-1,1\}$ be the *pseudo-random number sequence* (PRNS) generated from a known key k_w, and α $(\alpha > 0)$ be the modulation amplitude. The coordinate $\hat{v}_{i,m}$ of the vertex after watermarking is computed by the following formula;

$$\hat{v}_{i,m} = v_{i,m} + b'_i \cdot p_i \cdot \alpha \tag{7}$$

The extraction requires the key k_w used for the embedding. Key distribution can be achieved by using a public-key cryptography scheme, for example.

The modification amplitude α is chosen so that the displacement won't affect visual qualities of maps. The geographical map standard by the Geographical Survey Institute of Japan states that the maximal error arrowed in a 1/2500-scale map is 0.3mm, which corresponds to 75 cm in the real world. Random perturbation of vertices on the map by 1 or 2 integer coordinate points, that are, 10 or 20 cm in the real world, should be acceptable as long as there is no discontinuity artifacts introduced by the displacements.

2.2. Extraction

Prior to the extraction, a similarity transformation applied to the watermarked map is removed. This is done by minimizing the Euclid distance between vertices of landmarks in the watermarked and reference maps using an iterative optimization algorithm (Powell's method). After the alignment, vertices that are inserted to or deleted from the reference map are detected so that they are not used for the extraction. Then, the same rectangles used during the embedding are created on the reference map, which is then transferred to the watermarked map.

To extract a message bit, the algorithm compares the averaged vertex coordinates among a corresponding pair of rectangles. Let $\bar{v}_i$ be the number of vertices in the *i*th rectangle. The average of coordinates of vertices in the *i*th rectangle in the reference map $\bar{v}_i$, and the same in the watermarked map $\bar{\tilde{v}}_i$ are computed by averaging.

$$\bar{\tilde{v}}_i = \sum_{k=1}^{h_i} \tilde{v}_i \ , \ \ \bar{v}_i = \sum_{k=1}^{h_i} v_i \qquad (7)$$

and p_i be the same PRNS as is used for embedding.

$$q_j = \sum_{i=j\cdot c}^{(j+1)\cdot c-1} (\bar{\tilde{v}}_i - \bar{v}_i) \cdot p_i = \sum_{i=j\cdot c}^{(j+1)\cdot c-1} b'_i \cdot \alpha \cdot p_i^2 . \qquad (8)$$

When computing q_j, the value of $\bar{\tilde{v}}_i - \bar{v}_i$ is checked against a pair of thresholds to weed out unreliable signals. As the watermark is embedding by using one of the two displacement values, any values too far away from the two values are considered “unreliable”. If the PRNSs for the embedding and extraction are synchronized, and if disturbances applied to the vertex coordinates of $\hat{M}$ (e.g., additive random noise) are negligible,

$$q_j = c \cdot \alpha \cdot b'_i \qquad (9)$$

where q_j takes one of the two values $\{-\alpha c, \alpha c\}$. Since α and c are always positive, simply testing for the signs of q_j recovers the original message bit sequence a_j,

$$a_j = sign(q_j) \qquad (10)$$

The string a_j can easily be converted to the original message bit sequence b_i by applying an inverse of the mapping as the embedding.

3. EXPERIMENTS AND RESULTS

In all the experiments described below, we used the parameters *d*=10, $\alpha = 1$ (10cm in the real world), and c is set to the maximum automatically. (For example, if a map has the payload of 707 bits and the message is 32 bit, maximum chip rate becomes $c = 22$.)

3.1. Perceptibility

We first tested perceptibility of the watermark using 10 volunteers, all non-expert in GIS. We asked each one of them if she/he could tell a watermarked map if they are presented with a pair of maps printed full-size on a A4 paper, one watermarked and the other not. None could tell the watermarked maps from the non-watermarked map.

3.1. Payload

We compared information payloads among the three area subdivision methods. For the UNIF, we used two different numbers of subdivisions. Thus we compared 4 method-parameter pairs as a whole. Watermarking targets are 6 maps having various object densities as shown in Figure 1. As seen in Table 1, for all the maps tried, the MQUAD area subdivision method has the highest number of embeddable bits per map. We thus will use the MQUAD method for the attack resiliency experiments described in the next section.

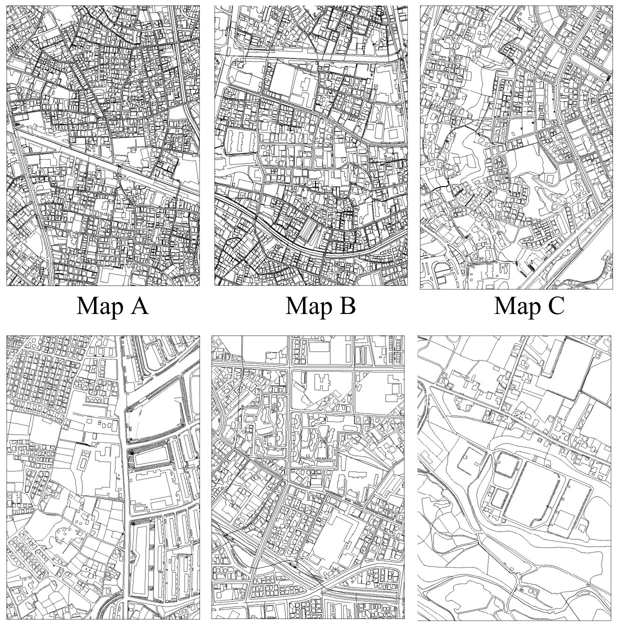

Figure 1. Maps used for the resiliency experiments.

Table 1. Numbers of embedded bits per map.

	UNIF		QUAD	MQUAD
	25×25	30×30		
Map A	553	708	621	936
Map B	513	621	543	828
Map C	485	524	361	707
Map D	376	428	257	591
Map E	351	348	231	456
Map F	185	159	142	304

3.2. Resiliency Against Attacks

As described above, we chose the MQUAD area subdivision algorithm for the following experiment, which compare the attack resiliency using the 8 attacks listed below. We compared the performance of the symbol and message repeating methods in this experiment.

- Translation: Translate all the vertices in the map by the same amount.
- Enlarge: Uniformly enlarge the map by an integer factor.
- Shrink: Uniformly shrinks the map. Coordinate values are rounded to the nearest integers.
- Object order scrambling: The order of appearance of objects (e.g., a polygon of a building) in the data file is scrambled.
- Vertex insertion: Vertices are added to target objects *i.e.*, polygons and polylines, while trying to preserve the appearance of the map.

- Vertex deletion: Vertices are deleted from target objects using the same criteria as the vertex insertion above.
- Additive random noise: Random noise having the amplitudes of either *A*=10 cm or *A*=50 cm is added to the vertex coordinate.
- Cropping: Each map is cropped according to the 8 cropping patterns shown in Figure 2. Areas of the cropped maps varied from 1/2 to 1/16 of the original.

Table 2 shows the results of the experiments. On almost all of the experiments, the message repeating method slightly outperformed the symbol repeating in this experiment. The advantage of message repeating method is significant especially for the cropping attacks with its spatially distributed modifications.

The watermark withstood similarity transformation, vertex insertion, vertex deletion, and additive random noise of amplitude 10 cm, *i.e.*, equal to the watermark modulation amplitude α . Adding the noise having amplitude of 50cm, five times the α , to the vertex coordinates partially destroyed the watermark. However, the noise with the amplitude 50cm clearly degraded the visual quality of the map.

Cropping the map down to 1/2~1/16 of the original size (by area) destroyed the watermarks. In this case, again, the message repeating method performed better than the other.

4. SUMMARY AND CONCLUSION

In this paper, we presented an algorithm to watermark vector digital maps. The algorithm embeds a bit into a map by displacing a set of vertices in a rectangle. The rectangles are generated by adaptively subdividing the map according to the density of vertices in the region. The watermark gains resiliency to attacks by displacing a group of vertices and by embedding the same message bit many times into the map.

Experiments showed that the watermarks produced by the method are resilient against additive random noise, addition and deletion of vertices, and similarity transformation. Its resiliency against cropping needs improvements, for example, by recruiting more geometrical primitives as the targets of modification.

REFERENCES

[1] F. Hartung, P. Eisert, and B. Girod, Digital Watermarking of MPEG-4 Facial Animation Parameters, *Computers and Graphics*, Vol. 22, No. 4, pp. 425-435, Elsevier, 1998.

[2] I. J. Cox, M. L. Miller, J. A. Bloom, *Digital Watermarking*, Morgan Kaufman Publishers, 2002.

[3] Shuh Endoh, Hiroshi Masuda, Ryutarou Ohbuchi, Satoshi Kanai, Development of Digital Watermarking Technology for Vector Digital Maps, *IPA Technology Expo* 2001 Reports, 2001 (Japanese).

[4] I. Kitamura, S. Kanai, and T. Kishinami, Watermarking Vector Digital Map using Wavelet Transformation, Proc. *Annual Conference of the Geographical Information Systems Association (GISA) 2000*, Vol. 9, pp.417-421, 2000 (Japanese).

[5] M. Kurihara, N. Komatsu, H. Arita, Watermarking Vector Digital Maps, Information Processing Society of Japan (IPSJ) Special Interest Group Report Vol. 2000, No. 36, (Computer Security No. 009-1, 2000) (Japanese).

ACKNOWLEDGEMENTS

We thank Zenrin Co. Ltd. for lending us with the map data for the experiments.

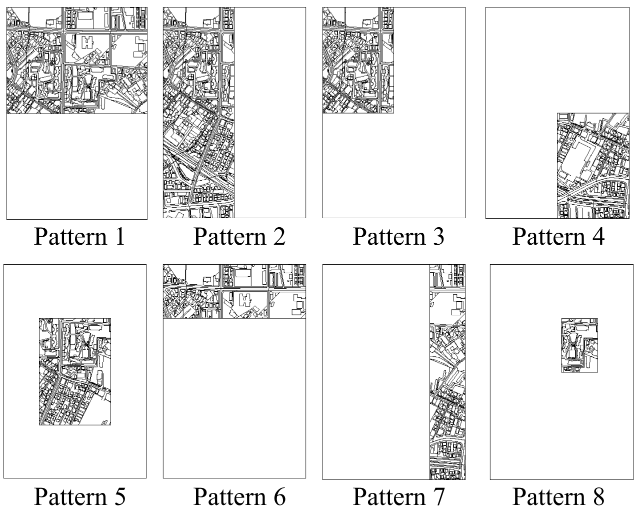

Figure 2. Eight cropping patterns used for the experiments.

Table 2. Bit error rate due to various attacks.

Attacks		Symbol repeating	Message repeating
Translation		0.0 %	0.0 %
Enlarge		0.0 %	0.0 %
Shrink		2.6 %	0.7 %
Object order scrambling		0.0 %	0.0 %
Vertex insertion		0.0 %	0.0 %
Vertex deletion		0.0 %	0.0 %
Additive random noise (A=10cm)		0.0 %	0.0 %
Additive random noise (A=50cm)		13.2 %	7.4 %
Cropping	Pattern 1	41.7 %	1.2 %
	Pattern 2	43.6 %	1.4 %
	Pattern 3	71.2 %	5.9 %
	Pattern 4	82.7 %	5.9 %
	Pattern 5	68.5 %	16.0 %
	Pattern 6	70.7 %	14.6 %
	Pattern 7	79.8 %	9.3 %
	Pattern 8	91.5 %	35.6 %