SB-RAWVec – A Semi-Blind Watermarking Method for Vector Maps

Karina Magalhães
Instituto de Computação - UNICAMP
Campinas, SP, Brazil - 13084-971
Email: karina.magalhaes@students.ic.unicamp.br

Ricardo Dahab
Instituto de Computação - UNICAMP
Campinas, SP, Brazil - 13084-971
Email: rdahab@ic.unicamp.br

***Abstract*—RAWVec [1] is a private watermarking method for vector maps that uses a raster image as watermark. Visually recognizable watermarks can add extra information like integrity and authentication, while blind watermarks do not need the original data to be published for the detection. This work presents a semi-blind version of RAWVec, i.e. a method that uses the original vector map during the detection without revealing it, but maintains the watermark as a raster image.**

I. Introduction

A watermark is some information added inconspicuously to digital data, usually to assure copyright detection on multimedia data, like images, videos and audio. Vector maps are images used in Geographic Information Systems (GIS), especially because they are easily scalable, have small file size and can be updated and maintained easily.

The *original data* must be embedded with the watermark, thus resulting in the new, copyrighted *marked data*. To test if some *target data* is an unauthorized copy of the marked data, a watermark detection procedure should be performed on the target data. This procedure usually requires some extra information to which we refer as the *key*.

Watermarks can be classified as private or blind according to the information needed during the detection algorithm. Private or non-blind watermarks use, along with the key and the data to be tested, the original data that was marked. Examples of non-blind watermarking techniques for vector are found in [2] and [3].

Blind watermarks use only the data to be tested and the key during the detection. Therefore, the detection is public, because it does not reveal any additional information and anyone can extract the watermark. Most watermarking methods for vector maps are blind, as in [4], [5], [6] and [7], because they usually have a wider range of application, while private watermarks are more efficient.

Another, less known, category [8] is that of Semi-Blind watermarks, in which the original map is watermarked and this marked version is used during the detection. Note that the original map is used during the detection, but it is not revealed.

In this paper we propose SB-RAWVec, an alternative version for RAWVec [1], a method for watermarking vector maps in spatial domain, i.e. the embedding algorithm modifies the coordinate of vertices. The main advantage of the present proposal is that it is semi-blind, while the original method uses a private technique.

This paper is organized as follows: in Section II some functions and concepts used on the algorithms are presented; in Section III we describe the embedding and detection algorithms for the original RAWVec method; in Section IV we describe the embedding and detection algorithms for SB-RAWVec; in Section V we analyze and compare the two methods and discuss their robustness against several attacks. Finally, in Section VI, we present conclusions and discuss future work.

II. Basic Concepts

The original RAWVec method and its semi-blind alternative use two functions $w()$ and $v()$ and a *Point Pattern Matching* Algorithm. These concepts are described in the following sections.

A. Functions $w()$ and $v()$

The function $w()$ modifies a square matrix $A_{n\times n}$ by shifting its elements. Let A be a square matrix of order n; then $B = w(A)$, has $b_{ij} = a_{uv}$, where $u = n - i + 1$ and $v = n - j + 1$. Figure 1 shows an example of an application of function w.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} => w(A) = \begin{pmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{pmatrix}$$

Figure 1. An example of function $w(A)$

It is important to notice some proprieties of function $w()$:

(a) Property 1: $w(w(A)) = A$, for A a square matrix.
(b) Property 2: $c * w(A) = w(c * A)$, for c a real number, and A a square matrix.
(c) Property 3: $w(A) + w(B) = w(A + B)$, for A and B squares matrices.

The watermark used in both methods is a bitmap image represented by a bi-dimensional matrix whose

elements store the color information or intensity for each pixel. The vector map must also be represented as a matrix, which can be obtained using a function $v()$, that calculates the point representation P from a vector map M. Vector maps are formed by geometric structures and, based on these structures' points, it is possible to create the point representation $P = v(M)$. Thus, each structure is decomposed into points, which are stored according to their type:

- Punctual objects: Structures described by points, like symbols and text. The coordinates of each point are stored.
- Linear objects: Structures described by point sequences, such as lines and polygons. The coordinates of each point in the sequence are stored.
- Parameterized objects: Structures described by parameters, like circles and ellipses. The object can be segmented in a few points, and these are stored; or the main points can be stored. If the object is a circle, for example, we can store a few points of the circle, or only its center.

Each coordinate must have a reference to its original object, making it possible to rebuild all the structures from the point representation.

B. Point Pattern Matching

A *Point Pattern Matching* algorithm compares two sets of points and determines a transformation that maps one set onto the other under a specified set of criteria. That is, given two sets of points P and Q, it finds a transformation T which takes P to Q; thus $P = T(Q)$. This transformation T can be found even if some random noise is added or some points are removed from the sets of points.

There are several types of Point Pattern Matching algorithms, as described in [9]. These algorithms can be used to detect and remove any geometric attacks on marked vector maps, as well as attacks based on object reordering and insertion. The main disadvantage of these algorithms is their inefficiency. In contrast, the methods presented in this paper use the algorithm described in [10], in which the nearest neighbors of each point are calculated and iteratively several transformations are tested until a global transformation is found. This algorithm is the most efficient we have been able to find in the literature.

III. The Original RAWVec Algorithm

This section describes the original RAWVec method [1]. The embedding algorithm marks the vector map with the raster image by shifting the geometric structures' coordinates. The detection algorithm uses the original vector map to extract a raster image that is then compared with the original one.

A. Watermark Embedding

Algorithm 1 shows the main steps for embedding a raster image R in the vector map M producing the marked vector map M'. It uses a positive real constant C that controls the maximum shifting of a pixel, so it will stay within the maximum tolerable error of the vector map. We discuss this issue in more detail in Section V-A.

Algorithm 1 Watermark Embedding

INPUT: raster image R, vector map M and constant C.
OUTPUT: marked vector map M'.

1. Calculate the point representation $P \leftarrow v(M)$.
2. Obtain matrices A_x and A_y from P.
3. Resize raster image R, creating the raster image E.
4. Calculate $B_x \leftarrow CE + A_x$; $B_y \leftarrow Cw(E) + A_y$, for C a real constant.
5. Build the marked vector map M' from matrices B_x and B_y.

Steps 1 and 2 build two square matrices – A_x for the x coordinate and A_y for the y coordinates – of point representation $P = v(M)$, as described on Section II-A. Let t be the number of points stored in P, n be the order of the matrices and $p_i = (x_i, y_i)$ be the points stored in P. Then

$$n = \sqrt{t}, \tag{1}$$

$$(A_x)_{ij} = x_{n(i-1)+j} \quad \text{and} \quad (A_y)_{ij} = y_{n(i-1)+j} \tag{2}$$

Step 3 resizes the raster image R, creating a new raster image E that has $n \times n$ pixels, as illustrated in Figure 2.

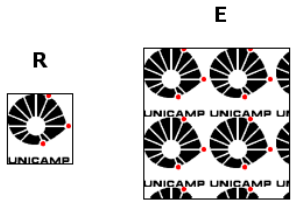

Figure 2. Resizing a raster image R, creating a new raster image E

Now that we have two matrices, A_x and A_y, representing the vector map M, and a matrix E, representing the raster image, we are able to embed E in the vector map. Therefore, Step 4 creates two matrices, calculating B_x and B_y:

$$B_x = CE + A_x, \quad \text{and} \quad B_y = Cw(E) + A_y. \tag{3}$$

Finally, in the last step, the marked vector map M' is built from matrices B_x and B_y, as described in Section II-A.

B. Watermark Detection

The detection algorithm consists in the extraction of the watermark and its comparison with the original one. It extracts the watermark S from the target vector map N, the one that is going to be tested, using the constant C, the original watermark R and the original vector map M. Therefore, this is not a blind watermark method. The basic steps are described below as Algorithm 2.

Algorithm 2 Watermark detection

INPUT: raster image R, vector map M, constant C and the vector map to be tested N.
OUTPUT: watermark S.

1. Embed the watermark R on the vector map M, using the algorithm described in Section III-A and returning the marked vector map M'.
2. Calculate the point representations $P \leftarrow v(M')$ and $Q \leftarrow v(N)$.
3. Compare the point representations P and Q using *Point Pattern Matching*, returning a transformation T, where $Q = T(P)$.
4. Build matrices A_x and A_y from $v(M)$, and B_x and B_y from $T^{-1}(Q)$.
5. Calculate $D_x \leftarrow \frac{B_x - A_x}{C}$; $D_y \leftarrow \frac{w(B_y) - w(A_y)}{C}$.
6. Calculate $D \leftarrow \frac{D_x + D_y}{2}$.
7. Resize the watermark D to its original size, returning the watermark S.

The first step of the detection algorithm is to embed the watermark R on the vector map M, resulting on the marked map M'. This marked vector map must be compared with the vector map N, using the *Point Pattern Matching* Algorithm [10], so that any transformation T used in an attack is found and removed from the vector map N.

Now we are able to build four matrices A_x and A_y from $v(M)$ and B_x and B_y from $T^{-1}(N)$[1]. Matrices A_x and A_y represent the original vector map and matrices B_x and B_y represent the vector map N that may be M'. So, we must calculate the watermark in vector map N and compare it with the original one. The watermark D is calculated using the four matrices A_x, A_y, B_x and B_y, the constant C and the function w:

$$D = \frac{D_x + D_y}{2}, \tag{4}$$

where

$$D_x = \frac{B_x - A_x}{C} \quad \text{and} \quad D_y = \frac{w(B_y) - w(A_y)}{C}. \tag{5}$$

Finally, the watermark D, with size $n \times n$, must be resized to its original size – the size of watermark R – producing watermark S. Watermarks S and R must be compared using a probabilistic algorithm and the human eye. According to this comparison we can conclude whether or not the vector map N was marked.

[1] $T^{-1}(N)$ here means the set of points of P which are pre-images of points of Q under T

IV. SB-RAWVec Algorithm

This section presents the modifications made on the original RAWVec method, resulting in a semi-blind alternative. Two marked vector maps are created, one must be published as the marked vector map, referred as *public map*, and the other is only used during the detection algorithm, referred as the *detection map*. The original vector map is not needed during the detection and a new constant I is added to help control the maximum shifting of a pixel along with the C constant.

A. Watermark Embedding

Algorithm 3 shows the steps for the watermark embedding of the new method. The main difference appears on the Step 4, in which the raster image is divided in two matrices, resulting in two marked vector maps.

Algorithm 3 Watermark Embedding

INPUT: raster image R, vector map M and two constants C and I.
OUTPUT: public marked vector map M_1 and detection marked vector map M_2.

1. Calculate the point representation $P \leftarrow v(M)$.
2. Obtain matrices A_x and A_y from P.
3. Resize raster image R, creating the raster image E.
4. Decompose matrix E into random matrices E_1 and E_2 such that $E = E_1 + E_2$, and $0 \leq z \leq I$ for each z an element of E_1.
5. Calculate $B_{1x} \leftarrow A_x + CE_1$; $B_{1y} \leftarrow A_y + Cw(E_1)$, for C a real constant.
6. Calculate $B_{2x} \leftarrow A_x - CE_2$; $B_{2y} \leftarrow A_y - Cw(E_2)$, for C a real constant.
7. Build the public marked vector map M_1 using matrices B_{1x} and B_{1y}.
8. Build the detection marked vector map M_2 using matrices B_{2x} and B_{2y}.

Steps 1 to 3 are exactly the same as in the original method. A point representation for the vector map M is created, as well as two matrices A_x and A_y, representing the x and y coordinates respectively. These two matrices must be square and have the same size. Next, the raster image is resized, in order to obtain a square matrix E with the same size as A_x and A_y.

Matrices A_x and A_y represent the positions for the objects in the vector map, and matrix E represents the color for each pixel in the raster image.

Step 4 represents the creation of two matrices E_1 and E_2, using the positive integer constant I and the matrix E. This step is based on Visual Cryptography [11], in

which an image is divided in two, and can be recreated by superimposing the two shares. In our case, the image can be reconstructed by adding the two shares. Note that the image E_2 is not visual, as it can have negative elements.

Using A_x, A_y and E_1, it is possible to obtain two matrices B_{1x} and B_{1y}, and therefore build the public marked vector map M_1:

$$B_{1x} = A_x + CE_1, \quad \text{and} \quad B_{1y} = A_y + Cw(E_1). \tag{6}$$

And, using A_x, A_y and E_2, it is possible to obtain two other matrices B_{2x} and B_{2y}, and build the detection marked vector map M_2:

$$B_{2x} = A_x - CE_2, \quad \text{and} \quad B_{2y} = A_y - Cw(E_2). \tag{7}$$

Note that the sum used in the calculation of the public map M_1 is replaced by a subtraction in the calculation of the detection map M_2.

The error added to M_1 can be controlled by the constants C and I, thus forcing the error added to M_2; therefore, M_2 can only be used during the detection algorithm. This is detailed in Section V-A.

B. Watermark Detection

Although it does not use the original map M, the detection algorithm uses a *Point Pattern Matching* algorithm to detect and remove attacks. This is possible due to the information hidden on the detection map M_2. Algorithm 4 shows the basic steps for this algorithm.

The first two steps are simple, they only create the point representations and the respective matrices for the two maps: B_{2x} and B_{2y} for the detection map M_2 and B_x and B_y for the target map N. The first main difference between this algorithm and Algorithm 2 is Step 4, in which the hidden information from the detection map is obtained.

From the embedding algorithm, described in IV-A, we have that $B_{2x} = A_x - CE_2$, $B_{2y} = A_y - Cw(E_2)$ and $E = E_1 + E_2$, so we can extract the information needed for the *Point Pattern Matching* algorithm from B_{2x} and B_{2y}, as follows:

$$\begin{aligned} B'_{2x} &= B_{2x} + CE \\ &= A_x - CE_2 + CE \\ &= A_x + C(E - E_2) \\ &= A_x + CE_1 \\ &= B_{1x} \end{aligned} \qquad \begin{aligned} B'_{2y} &= B_{2x} + Cw(E) \\ &= A_y - Cw(E_2) + Cw(E) \\ &= A_y + Cw(E - E_2) \\ &= A_y + Cw(E_1) \\ &= B_{1y} \end{aligned}$$

If the target map N is a tampered version of public map M_1, we can compare it with M_1 using a *Point Pattern Matching* algorithm. This algorithm will find any transformation or modification that maps N onto M_1, comparing matrices B'_{2x} and B'_{2y} with matrices B_x and B_y.

Algorithm 4 Watermark detection

INPUT: raster image R, detection marked vector map M_2, constant C and the vector map to be tested N.
OUTPUT: watermark S.

1. Calculate the point representation $P \leftarrow v(M_2)$ of detection map M_2 and the point representation $Q \leftarrow v(N)$ of target map N.
2. Build matrices B_{2x} and B_{2y} from P, and B_x and B_y from Q.
3. Resize raster image R, creating matrix E that has the same size as B_{2x}.
4. Calculate $B'_{2x} \leftarrow B_{2x} + CE$ and $B'_{2y} \leftarrow B_{2y} + Cw(E)$
5. Compare the points represented by the matrices B'_{2x} and B'_{2y} with the points represented by the matrices B_x and B_y using a *Point Pattern Matching* algorithm.
6. Remove the transformation returned by the *Point Pattern Matching* algorithm from the matrices B_x and B_y, creating B'_x and B'_y.
7. Calculate $D_x \leftarrow \frac{B'_x - B_{2x}}{C}$; $D_y \leftarrow \frac{w(B'_y) - w(B_{2y})}{C}$.
8. Calculate $D \leftarrow \frac{D_x + D_y}{2}$.
9. Resize the watermark D to its original size, returning the watermark S.

Step 6 removes the transformation and modification found by the *Point Pattern Matching* algorithm from B_x and B_y, resulting in two new matrices B'_x and B'_y. Now we are able to calculate the extract watermark and verify if the target map N has been stolen:

$$D = \frac{D_x + D_y}{2}, \tag{8}$$

where

$$D_x = \frac{B'_x - B_{2x}}{C} \quad \text{and} \quad D_y = \frac{w(B'_y) - w(B_{2y})}{C}. \tag{9}$$

Since the target map N is M_1 tampered, B'_x and B'_y will be equivalent to B_{1x} and B_{1y}, so:

$$\begin{aligned} D_x &= \frac{B'_x - B_{2x}}{C} \\ &= \frac{B_{1x} - B_{2x}}{C} \\ &= \frac{(A_x + CE_1) - (A_x - CE_2)}{C} \\ &= \frac{CE_1 + CE_2}{C} \\ &= E_1 + E_2 \\ &= E \end{aligned}$$

The calculation of D_y is more complex because the $w()$ function is used; therefore the three properties described in II-A are essential:

$$
\begin{aligned}
D_y &= \frac{w(B'_y) - w(B_{2y})}{C} \\
&= \frac{w(B_{1y}) - w(B_{2y})}{C} \\
&= \frac{w(A_y + Cw(E_1)) - w(A_y - Cw(E_2))}{C} \\
&= \frac{(w(A_y) + w(Cw(E_1))) - (w(A_y) - w(Cw(E_2)))}{C} \\
&= \frac{Cw(w(E_1)) + Cw(w(E_2))}{C} \\
&= E_1 + E_2 \\
&= E
\end{aligned}
$$

The matrix D must be resized to its original size and a raster image S must be built from it. The two images, R, representing the original watermark and S, representing the extracted watermark can be compared using the human eye or a comparison algorithm like the Pearson correlation [12]. This fact will be analyzed in Section V-A.

V. Experiments and Results

This section shows two analyses of the methods presented in this work, one for the algorithm complexity and the other for the maximum shifting of a pixel i.e., the error added to the vector map. Furthermore, an attack analysis of the two methods is also shown.

A. Algorithm Analysis

The embedding algorithm for both methods has complexity $O(t)$, where t is the number of points in the vector map. All the new steps of the alternative embedding algorithm have complexity $O(t)$ at most, so the total complexity is increased only by a constant.

The detection algorithms are more complex due to the steps for detection and attack removal. The *Point Pattern Matching* algorithm fulfils this role, and its complexity is $O(t(\log\ t)^{3/2})$. All the other steps have complexity $O(t)$ at most, so the total complexity of the two detection algorithms is $O(t(\log\ t)^{3/2})$. Hence, the original and the new method have the same complexity.

It is also important to analyze the quality of the marked vector map returned, i.e. the maximum shifting of a point on the marked vector map must not be greater than the tolerable maximum error for the original vector map.

Figure 3 shows the maximum shifting of a point for the two methods. In the original RAWVec method, this shifting is controlled only by the constant C, and some manipulation of the intensities of pixels in the raster image (like normalization) may be needed as seen in [1]. The maximum shifting of a pixel is given by $\pm CR_{max}\sqrt{2}$, where $R_{max} = \max(e)$ and $e \in R$.

On the other hand, in the SB-RAWVec method, this property can be controlled by the constants, C and I, and no change on the intensities of the pixels of the raster image is needed. The maximum shifting of a pixel in this case is given by $\pm CI\sqrt{2}$. This control refers only to the public map M_1; the error added to the detection map M_2 cannot be controlled, so this map cannot be published, as it can be visually degraded by the watermark.

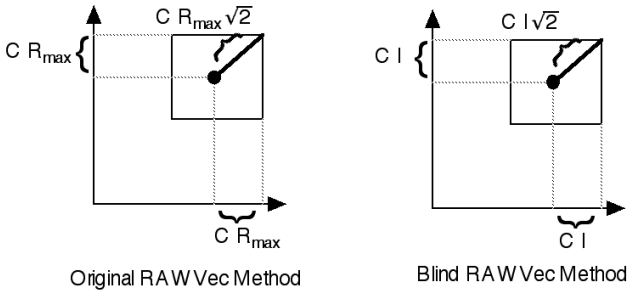

Figure 3. The maximum shift of pixel for the two methods presented.

The comparison between the original watermark R and the extracted watermark S uses two coefficients: r, the Pearson correlation coefficient [12] and h, the quality coefficient. The quality coefficient h is based on human observation. It can vary from 0 to 5 and it represents the answer from 5 people to the question "Do you think that the image R is the image D after some modifications?" Each positive answer represents 1 and each negative answer represents 0, and h is the sum of the answers. The Pearson correlation coefficient r is based on the pixels intensities and it is calculated as follows:

Let I_{m_R} and I_{m_S} be the average intensity of images R and S, respectively, and $I_{r_{ij}}$ and $I_{s_{ij}}$ be the intensity of pixels $r_{ij} \in R$ and $s_{ij} \in S$, respectively. Then

$$
r = \frac{\sum_i \sum_j (I_{r_{ij}} - I_{m_R})(I_{s_{ij}} - I_{m_S})}{\sqrt{(\sum_i \sum_j (I_{r_{ij}} - I_{m_R})^2)(\sum_i \sum_j (I_{s_{ij}} - I_{m_S})^2)}}. \quad (10)
$$

B. Attacks

Most attacks on watermarks for vector maps are based on the objects and vertex information of the vector maps [13]. Geometric attacks consist on transformations, such as rotations and translations, on the marked vector map with the goal of removing or degrading the watermark without damaging the vector map. Other attacks consist of: reordering or removal of some objects of the vector map; reordering and removal of vertices of some objects of the vector map; and adding random noise.

All these attacks can be detected and removed, to a certain extent, by a *Point Pattern Matching* algorithm that compares the vector map to be tested with a marked vector map. The transformations are found and returned by the algorithm, as well as added random information and points or objects removed from one of the maps. Therefore, the *Point Pattern Matching* is essential to the robustness of these methods.

Figures 5 and 6 show the extracted watermark of a simple test, without any attacks, for the original RAWVec method and the SB-RAWVec method, respectively. Figures 7 and 8 show the extracted watermark of a test with a combined attacked vector map for the same two methods. In this case, the marked vector map was rotated 10^o, translated, scaled to double of its original size, 11% of its points were cropped, 37% of its objects were reordered, and some random noise was added. The coefficients r and h are calculated for all the tests, as described in V-A. Figure 4 shows the original watermark used during the tests. Several other tests can be found in [14] (in portuguese).

Figure 4. The original watermark used during the tests.

Figure 5. Test without attacks using the original RAWVec method. $r = 99.9648\%$ $h = 5$

Figure 6. Test without attacks using the SB-RAWVec method. $r = 99.9637\%$ $h = 5$

Figure 7. Test with a combined attack using the original RAWVec method. $r = 98.1838\%$ $h = 5$

Figure 8. Test with a combined attack using the SB-RAWVec method. $r = 89.6466\%$ $h = 4$

The original images used as watermarks showed in Figures 4, 5, 6, 7 and 8 are small (60×68 pixels), so that several copies can be added to a single map as pictured in Figure 2. This redundancy allows for an easier retrieval of the right information about a pixel, because during the extraction, more than one copy of each pixel is analyzed.

The original vector map used in this paper was produced by Robert Ford, which can be found in http://www.go.dlr.de/pdinfo_dv/xfig_Examples/transit.fig.

VI. Conclusions and Future Work

In this work, the RAWVec method for watermarking vector maps is revised and an alternative SB-RAWVec method is presented. The new method is as efficient and robust as the original one, since they both make use of a *Point Pattern Matching* algorithm and redundancy to detect and remove attacks.

This new method has the same advantages as the original one - such as the use of an image as the watermark - with the addition of the semi-blinding feature. Although it may seem a marginal improvement, blind and semi-blind schemes are more practical, since the original map may not be available for a number of reasons, limiting the application of the non-blind method. Another advantage of our new method is a better control of the maximum shifting of a pixel, i.e. the quality of the marked vector map, now controlled by two constants.

Other types of attacks on SB-RAWVec remain to be studied; such as those more commonly seen in watermarking for other medias like video, audio and bitmap images.

References

[1] D. A. Marques, K. M. de Magalhães, and R. Dahab, "Rawvec – a method for watermarking vector maps," in *SBSeg 2007: Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, 2007.

[2] R. Ohbuchi, H. Ueda, and S. Endoh, "Robust watermarking of vector digital maps," in *Proc. IEEE Conference on Multimedia and Expo 2002*, 2002.

[3] ——, "Watermarking 2d vector maps in the mesh-spectral domain," in *SMI '03: Proceedings of the Shape Modeling International 2003*. Washington, DC, USA: IEEE Computer Society, 2003, p. 216.

[4] H. Kang, K. Kim, and J. Choi, "A vector watermarking using the generalized square mask," vol. 00. Los Alamitos, CA, USA: IEEE Computer Society, 2001, p. 0234.

[5] M. Voigt and C. Busch, "Watermarking 2d-vector data for geographical information systems," in *Proc. SPIE, Security and watermarking of Multimedia Content*, 2002, pp. 621–628.

[6] G. Schulz and M. Voigt, "A high capacity watermarking system for digital maps," in *MM&Sec '04: Proceedings of the 2004 workshop on Multimedia and security*. New York, NY, USA: ACM Press, 2004, pp. 180–186.

[7] V. Solachidis, N. Nikolaidis, and I. Pitas, "Fourier descriptors watermarking of vector graphics images," in *ICIP00*, 2000, pp. Vol III: 9–12.

[8] R. Ohbuchi, A. Mukaiyama, and S. Takahashi, "A frequency-domain approach to watermarking 3d shapes," in *EUROGRAPHICS 2002*, vol. 21, 2002.

[9] G. Cox and G. de Jager, "A survey of point pattern matching techniques and a new approach to point pattern recognition." [Online]. Available: citeseer.ist.psu.edu/96624.html

[10] P. van Wamelen, Z. Li, and S. Iyengar, "A fast expected time algorithm for the point pattern matching problem," Louisiana State University, Dept. of Mathematics, Tech. Rep., 1999.

[11] M. Naor and A. Shamir, "Visual cryptography," *Lecture Notes in Computer Science*, vol. 950, pp. 1–12, 1995. [Online]. Available: citeseer.ist.psu.edu/naor95visual.html

[12] E. K. Yen and J. G. Roger, "The ineffectiveness of the correlation coefficient for image comparisons."

[13] X. Niu, C. Shao, and X. Wang, "A survey of digital vector map watermarking," *International Journal of Innovative Computing, Information and Control (IJICIC)*, vol. 02, pp. 1301–1316, 2006.

[14] K. M. de Magalhães, "Uma alternativa pública para o método de marcas d'Água raster em mapas vetoriais (rawvec)," Master's thesis, UNICAMP, 2009.