Using Reversible Steganography Algorithm to Embed Metadata in Vector Maps

Sheng-Ming Wang
Department of Information Science and Management System, National Taitung University

Fu-Mei Chen
Department of Information Management, National Dong Hwa University

Kai-Wei Chen
Department of Computer Science and Engineering, National Chung Hsing University

Wei-Pang Yang
Department of Information Management, National Dong Hwa University

Abstract

Vector maps are the fuel of many Geographic Information System (GIS) applications. Metadata, which is the data about vector maps, are introduced to provide the details of the vector maps. The usage of metadata is still facing some problems. Especially the storage of metadata is a problem that needs to be resolved. This paper presents an internal metadata storage mechanism for solving metadata storage problem by using a reversible steganographic algorithm to embed metadata in vector maps. Experimental results show that this method provides a solution for metadata embedding with high capacity but low distortion. The capacity of metadata embedding is 2(n-2) bits, where n is the amount of vertices of vector maps. To the best of our knowledge, our method provides the highest capaicty achieved in the literature of steganograhy for vector maps. In considering to the capacity required by the metadata elements of ISO 19115 metadata standard that we have adapted in this paper, a vector map should has at least 5458 vertices so that all mandatory and conditional metadata elements can be embedded in the vector map. Since the conditional elements should not be embedded alone, a vector map should has at least 1998 vertices so that the mandatory metadata elements can be embedded and integrated with the vector map. Experimental results also show that there is an insignificantly difference between the original and the recovery map, which is less than 3.41E-11 of the root mean square error (RMSE) and is imperceptible to the human visual system. Surely, the accuracy of recovery maps satisfies the requirements of all GIS applications development. Meanwhile, our method is

robust against the affine transformation including translation, rotation, uniform scaling, and their combinations.

Key words : metadata, steganographic algorithms, vector maps, embedding capacity, reversible.

應用可逆式資料隱藏法將後設資料嵌入於向量地圖

王聖銘
國立台東大學資訊管理學系

陳富美
國立東華大學資訊管理學系

陳凱威
國立中興大學資訊科學研究所

楊維邦
國立東華大學資訊管理學系

摘要

向量地圖（Vector Maps）是很多地理資訊系統（GIS）發展與應用的基礎。向量地圖的後設資料（Metadata）除提供該地圖相關的詳細資料外，也是GIS發展中非常重要的參考資料。雖然後設資料的應用已有既定的基礎，但卻仍存在一些問題，尤其是其資料儲存的問題。本研究提出一個應用可逆式資料隱藏演算法，可將後設資料嵌入向量地圖。研究結果顯示，本研究可於向量地圖中嵌入2（n-2）位元的資料量，其中n代表此向量地圖的頂點數。與其它已知的研究成果相比，我們所提的方法具有最高的資料嵌入量。而本研究實驗結果顯示，一個向量地圖最少要有5458個頂點數，才能將完整的嵌入本研究所採用的ISO 19115後設資料標準中的所有項目。而最少要有1998頂點數，才能嵌入該標準所訂定的主要後設資料項目。實驗結果也顯示，本研究將後設資料擷取出來後的復原圖與原始圖間僅有肉眼所無法分辨出來的3.41E-11的RMSE誤差量。此結果可滿足任何GIS應用與發展上的精度需求。另我們的方法也具其強韌度，可抗拒外在平移、旋轉、等量縮放，以及其組合性的攻擊。

關鍵字：後設資料、資料隱藏演算法、向量地圖、訊息容量、可逆式

1. INTRODUCTION

Ggeographical information data and services are becoming accessible in various new forms and through a wide range of applications as well as new classes of devices (Wang & Chen 2008). As the technology develops we need to discuss the potentials, problems and technical issues of emerging development trends of geographic information data and services (Goodchild et al. 1997；Mnzis 2000). Recently, online interactive mapping systems are growing in popularity (Fisher 2007；Kaasinen 2003；Tan et al. 2008；Zhang et al. 2008). The evolutions of mashup of different Web Service and data resources have flourished the applications of geographic information systems (GISs). A mashup is a kind of mechanism that combines two or more separate data streams to create application content (Yu et al. 2008). Already, hundreds of mashups overlay maps with everything from such practical information as gas station locations, rice fields, hurricane movements, hot springs sites and crime statistics (Butler 2006). Nowadays, online mapping mashup is evolving into a nexus of technologies and communities that is changing the fundamental use of the Internet, as well as redefining the concept of vector maps used in various applications (Goodman & Moed 2006).

Vector maps, which emphasize spatial variation of one or a small number of geographic distributions, are used to display spatial pattern of a theme or series of attributes. These patterns may be physical phenomena such as administration boundary and climate or human characteristics such as population density and village place. Vector maps are the most important component in an online mapping mashup service system, because they are the fuel of the other layers. Hence, it is important to know the details, such as copyright, scale, quality, accuracy, specifications, date, and usage limitation of these vector maps.

Metadata, which is the data about vector maps, are introduced to provide the details of vector maps. For example, users need metadata to locate appropriate data sets. Meanwhile, metadata provides information about the data available within an organization or from catalog services, clearinghouses, or other external sources. In the data quality issue, metadata not only helps find data, but once data has been found, it also tells how to interpret and use data. Furthermore, publishing metadata will facilitate data sharing. The data sharing mechanism is the core of online mapping mashup service system (Gunther & Voisard 1998).

The Importance of metadata for GIS application development has been emphasized in many literatures (Boll et al. 1998；Gunther & Voisard 1998；Kashyap & Sheth 1996；Turner 2004). There are also metadata standards been created and implemented (Federal Geographic Data Committee (FGDC) 2006, International Organization for Standardization (ISO) 2003). However, the usage of metadata is still facing some problems. The first problem occurs because the domain of vector maps is too wide for any single standard to give an ultimate solution to

it. Furthermore, the storage of metadata is another problem that needs to be resolved. There are two kinds of metadata storage: external and internal storing. In external storing, which nowadays is used by vector maps in GISs, metadata is stored in a separate file from the actual data content. Although, this allows better use of metadata in dedicated metadata management servers. These database servers can then search and store metadata information and only have references to the actual data. It is that the vector map is not guaranteed to exist and this easily leads to the situation where metadata server contains lots of ghost records, that does not have the corresponding content anywhere on the Web available or at least the record can not help to find it. The situation will cause serious defects while implementing data mashup.

For solving the problem, an internal storing mechanism to embed metadata in the same vector map is proposed in this paper to store metadata in the same file as the data and is thus also available if the data itself is available. This paper presents a mechanism by using a reversible steganographic algorithm to embed metadata in vector maps with high capacity but low distortion. In our approach, we can achieve the embedding capacity of 2(n-2) bits where n is the vertex numbers of a vector map. To the best of our knowledge, our capacity is much higher than that can provide by the current state of the art reversible steganography algorithms for vector maps, such as shown in several researches (Shao et al. 2006；Voigt et al. 2004；Wang et al. 2007). In addition, we only need to record extra little information to restore original coordinate value with insignificant distortion that is within the standard vector map accuracy.

We arrange the remains of this paper as follows. Section 2 surveys related works. We present details of our algorithm including the data embedding and data extracting processes in section 3 and 4, respectively. Section 5 gives experimental results. Conclusions and future works are described in the final section.

2. RELATED WORKS

To the best of our knowledge few works have been done on using reversible steganography algorithm to integrate metadata with vector maps. Most of them are focusing on embedding secret message to 2D vector maps (Chen et al. 2007；Shao et al. 2006；Voigt et al. 2004；Wang & Wang 2006) or images (Kim et al. 2008；Tian 2003). Hence the section will begin with an overview of steganography techniques. Further discussion will focus on steganography techniques that apply to vector maps. Finally, the metadata standards and applications, especially for vector maps used in GIS applications will be reviewed.

2.1 Steganography Overview

Steganography is the art and science of communicating in such a way that it hides the existence of the secret message during mutual communication. The main goal of steganography

is to hide a secret message S in a cover medium D, to produce a stego medium D', practically indistinguishable from D by people, in such a way that an eavesdropper cannot detect the presence of S in D'. On the contrary, the main goal of watermarking is to embed watermarks W in the host medium D to obtain a watermarked medium D' in such a way that an attacker cannot remove or replace W in D'.

Bogomjakov have defined that steganography (or, more simply, data-hiding) is the science of hiding messages in media in such a way that even the existence of the message remains undetected to all but the recipient (Bogomjakov et al. 2008). In the aspect of secret communication, steganography is defined as the art and science of communicating in a way which hides the existence of the communication. In the other word, steganography is the technique of hiding secret messages or data within other media to all eyes except those of the sender and intended receiver (Kahn 1996).

Petitcolas mentioned that (Petitcolas et al. 1999), there has been a growing interest, by different research communities, in the fields of steganography, digital watermarking and fingerprinting. This led to some confusion in the terminology. However, Pfitzmann (1996) has proposed a classification of information hiding techniques as shown in Figure 1. According to the figure, the purpose of steganography is having a covert communication between two parties whose existence is unknown to possible attackers; a successful attack consists in detecting the existence of this communication. Copyright marking, which is the root of watermarking techniques, has the additional requirement of robustness against possible attacks.

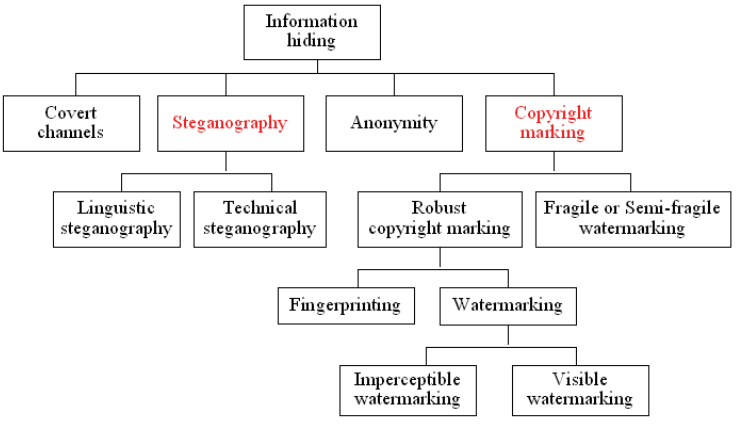

Figure 1: A classification of information hiding techniques (Pfitzmann 1996)

Popa (Cummins et al. 2004；Popa 1998) had proposed another diagram shown in Figure 2. The diagram shows that two general directions are distinguished within steganography: protection against detection and protection against removal. Protection against detection is achieved using schemes that do not modify in a visible way the original unmarked object; the modifications are not visible by the humans or by the computers. Protection against removal

supposes that the scheme should be robust to common attacks; it is impossible to remove the hidden data without degrading the object's quality and rendering it useless.

According to Figure 2, steganography can be used to hide a message intended for later retrieval by a specific individual or group. In this case, the aim is to prevent the message being detected by any other party. The other major area of steganography is copyright marking, where the message to be inserted is used to assert copyright over a document. This can be further divided into watermarking and fingerprinting.

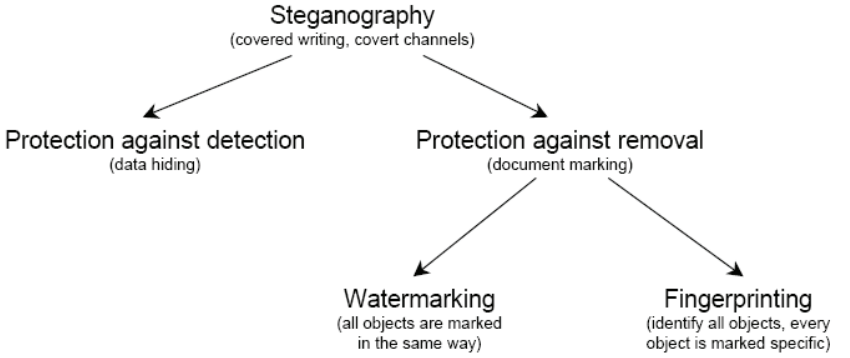

Figure 2: Two general directions distinguished within steganography (Popa 1998)

Furthermore, we compare the differences between steganography and encryption. Although they are both used to ensure data confidentiality, the main difference between them is that with encryption anybody can see that both parties are communicating in secret. Steganography hides the existence of a secret message and in the best case nobody can see that both parties are communicating in secret. This makes steganography suitable for some tasks for which encryption aren't, such as data embedding. Adding encrypted copyright information to a file could be easy to remove but embedding it within the contents of the file itself can prevent it being easily identified and removed.

2.2 Reversible Steganography on Vector Maps

In computer-based steganography, digital media file, such as images, audio, and 3D models, are used as innocuous-looking hosts for embedding secret messages. There are plenty of ways to hide messages within images. This is because an image, being a raster data structure with array of pixels, typically contains an enormous amount of redundant information. In contrast to raster data structure, vector map is a data structure uses sequences of coordinates to represent points, lines, and polygons on a map. It is not easy to hide messages within vector maps, since each of these units is composed simply as a series of one or more coordinate points and have no redundant information being stored. In addition, hiding data in vector maps will generally induce some distortions to the coordinates of the vector data (Wang et al. 2007). Meanwhile,

due to the strict GIS application requirements on accuracy of vector maps, modifications to map data are generally undesired. Therefore, as pointed out by Chen (Chen et al. 2007), reversible schemes are required for applying steganography in vector maps because the distortion can be removed after hiding data been extracted.

There are some works have been done on data hiding with vector maps. Voigt (Voigt et al. 2004) first proposed the method of reversibly hiding data in vector maps. They hide the data by modifying the integer discrete cosine transform mechanism seems to be complex and the data hiding capacity is low in this method. Schulz and Voigt (Schulz & Voigt 2004) proposed a non-reversible data hiding scheme with capacity of 0.1 bit per vertex. Wang (Wang et al. 2007) proposed two reversible data hiding schemes for vector maps based on difference expansion under the concept of watermarking techniques. The first one is coordinate-based scheme where the correlation of map coordinates is utilized for data hiding. The hidden data are embedded by changing the coordinate differences between the adjacent vertices of the map. The scheme can achieve the highest capacity of 0.279 bit/vertex with RMSE value of 0.00295 in the maps with dense vertices whereas the performance could be seriously decreased for those maps whose coordinates exhibit low correlation. The second scheme uses the Manhattan distances between adjacent vertices as the cover data to implement a distance-based scheme. The distance-based scheme can achieve the highest capacity of 0.262 bit/vertex with RMSE value of 0.00055 and is suitable for the maps whose extracted distances exhibit high correlation. Chen (Chen et al. 2007) proposed a reversible data hiding algorithm for vector maps in spatial domain. The algorithm performs high data capacity where the distortion rate is under 0.02%. In this paper, we use Chen's algorithm as a base for developing our method to embed metadata in vector maps.

In our approach, we first adopt PCA to produce two principal axes of the 2D vector map. Next, we sort vertices according to coordinate values of two axes. We then use multi-layer concept to classify the two sorting axes: the odd sorting list and even sorting list. The coordinate value of the modulation is individually embedded into the odd or even sorting lists. In this method we can not only extracting data, but also recover to the original coordinate value while extracting the information. In our approach, we can embed a bit in each interval. Hence we can embed 2(n-2) bits of data where n is the vertex numbers of a vector map. In addition, we only need to record extra little information to restore original coordinate value.

2.3 Metadata Standard and Definition

Metadata is defined as "data about data". Metadata of vector maps provide the background information of the content, quality, condition, and other appropriate characteristics of the vector maps. The International Organization for Standardization (ISO) proposes several closely related definitions, such as data for describing and documenting data; data about datasets and usage aspects of it; and data about the content, quality, condition, and other characteristics of

data (International Organization for Standardization (ISO) 2003). These definition is similar to that proposed by Federal Geographic Data Committee (FGDC) (Federal Geographic Data Committee (FGDC) 2006).

In GIS applications, metadata is a formal documentation describing the characteristics of a specific geospatial data set. For example, metadata may describe the details of vector maps include: what it is about, where it is to be found, what is the quality of the data for a specified purpose, what spatial location does it cover and over what time period, when and where the data was collected and by whom, what purposes the data has been used for, and what related data sets are available. In general, metadata should at least include dataset names with descriptions of what they contain, coordinate system and datum, theoretical accuracy, source document names with dates and scales, data conversion methods, update history, lists of field names and what they contain, and keys to any codes used in these fields.

ISO 19115:2003 (2003) has formally defined the schema required for describing geographic information and services(International Organization for Standardization (ISO) 2003). It provides information about the identification, the extent, the quality, the spatial and temporal schema, spatial reference, and distribution of digital geographic data. It also defines mandatory and conditional metadata sections, entities, and elements. The details of ISO 19115 core metadata elements are shown in Table 1

Table 1: ISO 19115 Core Metadata Elements

Mandatory Elements	Conditional Elements
1. Dataset title 2. Dataset reference date 3. Dataset language 4. Dataset topic category 5. Abstract 6. Metadata point of contact 7. Metadata date stamp	1. Dataset responsible party 2. Geographic location by coordinates 3. Dataset character set 4. Spatial resolution 5. Distribution format 6. Spatial representation type 7. Reference system 8. Lineage statement 9. On-line Resource 10. Metadata file identifier 11. Metadata standard name 12. Metadata standard version 13. Metadata language

Another metadata standard for geospatial data is defined by Dublin Core Metadata Initiative (DCMI) (Dublin Core Metadata Initiative (DCMI) 1998). The metadata elements fall into three groups which roughly indicate the class or scope of information stored in them, which are elements related mainly to the Content of the resource, elements related mainly to the resource when viewed as Intellectual Property, and elements related mainly to the Instantiation of the resource. These elements are shown in table 2.

Table 2: Dublin Core Metadata Elements

Content	Intellectual Property	Instantiation
1. Title 2. Subject 3. Description 4. Type 5. Source 6. Relation 7. Coverage	1. Creator 2. Publisher 3. Contributor 4. Rights	1. Date 2. Format 3. Identifier 4. Language

Recall that we will present a mechanism by using reversible steganographic algorithm to embed metadata in vector maps with high capacity but low distortion for data mashup service. In our approach, we can embed a bit in each interval constructed from the coordinates of vertex. As a result, we can achieve the embedding capacity of 2(n-2) bits where n is the vertex numbers of a vector map. Although. to the best of our knowledge, our capacity is much higher than that can provide by the current state of art reversible steganography algorithms for vector maps, such as shown in several researches (Shao et al. 2006；Voigt et al. 2004；Wang et al. 2007). The embedding capacity is limited by the vertex numbers of vector maps. For preventing the overflow of embedding capacity, we will just employ the ISO 19115 metadata standard in this paper. For vector maps with small amount vertex, only mandatory metadata elements are embedded. For vector maps with sufficient amount of vertex, all metadata elements are embedded.

The metadata definition and the space defined for the mandatory and conditional elements are shown in table 3 and table 4.

Table 3: The definition of mandatory metadata elements used in this paper

Mandatory Elements	Data Type	Data Space(Byte)	Definition
1. Dataset title	Char	100	Define the name of the map
2. Dataset reference date	Date	10	dataset create date
3. Dataset language	Char	20	dataset in language
4. Dataset topic category	Char	20	defined according to ISO 19115 category
5. Abstract	String	250	brief description of the map
6. Metadata point of contact	Char	90	dataset contact point
7. Metadata date stamp	Date	10	date that acquire the map
Total Space		500	

Table 4: The definition of conditional metadata elements used in this paper

Conditional Elements	Data Type	Data Space(Byte)	Definition
1. Dataset responsible party	String	200	Define the organization that responsible to the vector map
2. Geographic location by coordinates	Number	100	Record the coordinate value the boundary box of the vector map
3. Dataset character set	Char	20	Show the character set used by the vector map
4. Spatial resolution	Char	20	Record the spatial resolution
5. Distribution format	Char	30	Define the format for distribution
6. Spatial representation type	Char	20	Record the Spatial representation type of the vector map
7. Reference system	Char	30	Record the reference system of the vector map
8. Lineage statement	String	200	Record the lineage of the vector map
9. On-line Resource	String	150	Record the address for online access of the vector map
10. Metadata file identifier	String	60	Show the identifier of the metadata
11. Metadata standard name	Char	30	Show the adapted metadata standard
12. Metadata standard version	Char	5	Define the metadata standard version
13. Metadata language	Char	20	Define the metadata language
Total Space		865	

We can see from table 3 and table 4 that the embedding capacity should have at least 500 bytes, which is 4000 bits, in order to embed the mandatory elements in a vector map. Since the conditional elements should not be embedded alone, the embedding capacity should have at least 1365 bytes, which is 10920 bits, in order to embed the conditional elements in a vector map. This will be a challenge and limitation to the method proposed in this paper.

3. METADATA EMBEDDING PROCESSES

This section presents in details our reversible steganographic algorithm for embedding metadata in vector maps. Figure 3 shows the diagram of metadata embedding process, which contains metadata processing phase, cover map transformation phase and metadata embedding phase.

3.1 The Metadata Processing Phase

Given a cover vector map, we calculate its amount of vertex to decide the metadata volume to be embedded. Then we transform the proper mandatory and conditional metadata elements to XML format. The XML file is then transfer to binary format and ready for embedding in its associated cover map.

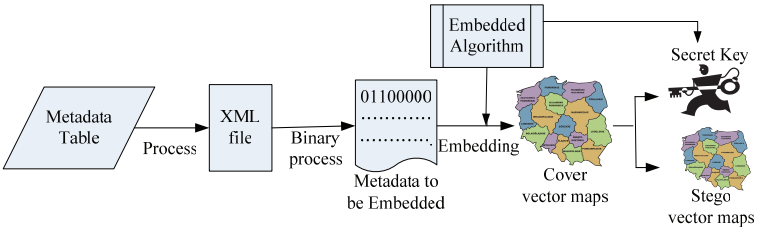

Figure 3: Metadata embedding process diagram

3.2 The Cover Map Transformation Phase

Given a cover vector map in the World coordinates, in the transformation phase, we transform these vertices to the PCA coordinates. We generate two lists, the odd sorting list and the even sorting list in this process. In the data embedding phase, we input the metadata and embed them into these two lists.

The transformation phase is the initial phase in our method. Suppose a cover vector map contains n vertices located originally in the two-dimensional World coordinate system, which has two orthogonal axes, X-axis and Y-axis. We denote these n vertices as $W=\{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$, where the suffix "W" indicates that these vertices are currently in the World coordinate system. We complete the transformation phase using the following four steps of operation.

First, we compute the gravity center (G) of the cover vector map. Then, we employ a principal component analysis (PCA) technique to produce two principal axes for the cover map in the second step. Without loss of the generality, we refer to two principal axes as the PCA X-axis and the PCA Y-axis, respectively. Surely, given these two axes and the gravity center G, we can construct a unique PCA coordinate system. This allows us to transform the coordinates of vertices in the cover vector map to the PCA coordinate system in the third step. After this step, the original n vertices $W=\{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$ becomes in the PCA coordinate system, which can be expressed as $PCA =\{(A_1, B_1), (A_2, B_2), ..., (A_n, B_n)\}$.

In the final step, we sort vertices with respect to the PCA X-axis coordinates in the PCA X-axis. This produces a sorting list, L_X, which contains n PCA X-axis coordinates. We referred to each "point" in the LX as $X_1, X_2, ..., X_n$, and the sorting list as $LX = \{ X_1, X_2, ..., X_n \}$. In this expression, X_1 represents the smallest PCA X-axis coordinates, while X_n the largest one. Similarly, we apply the sorting operation on the vertex coordinates with respect to the PCA

Y-axis, producing a list, L_Y, with PCA Y-axis coordinates Y_1, Y_2, ..., Y_n. With the extremes in the PCA X-axis and PCA Y-axis, we can construct a bounding box (BB) encompassing the cover vector map in the PCA coordinate system. The four boundary vertices of the bounding box in a counterclockwise order are $BV_1=(X_1, Y_1)$, $BV_2=(X_n, Y_1)$, $BV_3=(X_n, Y_n)$, and $BV_4=(X_1, Y_n)$. Given two extreme boundary vertices (X_1, Y_1) and (X_n, Y_n), we can compute the diagonal length of the bounding box (DLB) using the common Euclidean distance formula, where

$$DLB = [(X_n - X_1)^2 + (Y_n - Y_1)^2]^{0.5} \quad (1)$$

These four boundary vertices may not be any of vertices in the cover vector map. Instead, they represent the two extreme boundaries in the PCA X-axis and PCA Y-axis with respect to the gravity G.

3.3 The Metadata Embedding and Reversible Mechanism

The data embedding process is used to embed metadata in vector maps. We complete this process using the following two steps.

In the first step, we construct two lists with respect to the PCA X-axis. Recall that we have produced a sorting list, L_X, which contains n PCA X-axis coordinates in the transformation phase, where each "point" in the L_X is referred to as X_1, X_2, ..., X_n, and the sorting list as $L_X = \{ X_1, X_2, ..., X_n\}$. Given this sorting list, we can further classify index into the odd indices or the even indices, generating two lists, the odd sorting list and the even sorting list, respectively. Figure 3 illustrates the generation of the odd sorting list and the even sorting list. We denote the odd sorting list being generated as OL_X and the even sorting list as EL_X, respectively. Figure 4 shows the odd sorting list $OL_X=\{X_1, X_3, X_5, ...,\}$ and the even sorting list $EL_X= \{X_2, X_4, X_6, ...\}$. Observing the original sorting list $L_X = \{ X_1, X_2, ..., X_n \}$ we can compose a number of odd intervals which takes two "points" with odd indices as the boundaries. This interval contains one "point" that has an even index. For example, X_1 and X_3 are two "points" with odd indices and the interval contains another "point" X_2 with an even index. We denote this interval as $\{X_1$-X_2-$X_3\}$. Similarly, we can build a number of even intervals composed by two "points" with even indices and each interval contains another "point" with an odd index.

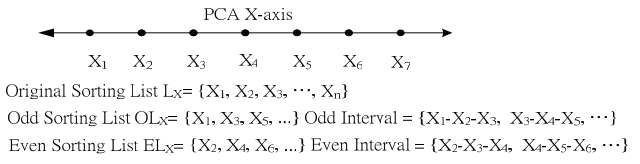

Figure 4: Constructing Odd and even list in the PCA X-axis

In the second step, we embed the metadata into the "point" X_2 in the first interval constructed from the odd sorting list OLX, which can be represented as $\{X_1\text{-}X_2\text{-}X_3\}$ before the data embedding. The data embedding into the "point" X_2 is to shift it to an appropriate position referred to as X'_2 (see the description later). Clearly, we will generate a new interval $\{X_1\text{-}X'_2\text{-}X_3\}$ after the data embedding. Then, we embed another bit into the "point" X_3 in the first interval constructed from the even sorting list EL_X; i.e., we embed a bit into the interval $\{X'_2\text{-}X_3\text{-}X_4\}$. Again, the embedding will shift the "point" X_3 to an approximate position X'_3, resulting a new interval $\{X'_2\text{-}X'_3\text{-}X_4\}$, where each "point" X'_2 or X'_3 has embedded one bit of metadata. Similarly, we embed the third bit into the "point" X_4 of the second interval constructed from the odd sorting list again. Followed this embedding approach, we embed one bit of metadata in the odd sorting list first and then embed another bit in the even sorting list. This alternation embedding approach can allow us to embed one bit of metadata into (n-2) vertices; namely, X_2, X_3, ..., X_{n-1}, keeping the first "point" X_1 and the last "point" X_n intact. Clearly, our algorithm can embed 2(n-2) bits when taking into consideration of odd sorting lists and even sorting list with respect to both the PCA X-axis and PCA Y-axis.

Now, we illustrate how to embed a bit, either "0" or "1", into an interval. We assume the interval is $\{X_i\text{-}X_i\text{+}1\text{-}X_{i+2}\}$, where the boundary points are "X_i" and "X_{i+2}", and the "point" that can be embed a single bit is X_{i+1}, as shown in Figure 5. We first divide the interval into two equal sub-intervals. The left sub-interval is defined as with the status value of "0" and the right sub-interval is with the status value of "1". The status value of the "point" X_{i+1} for data embedding is decided base on the sub-interval it is located. For example, the "point" X_{i+1} shown in Figure 5 is on the status "0" since it locates within the left sub-interval. It is on the status "1" if it locates within the right sub-interval.

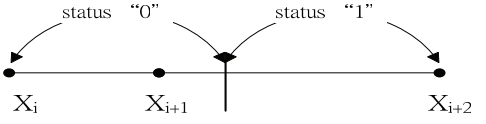

Figure 5: The sub-interval status value of "0" and "1" defined between X_i and X_{i+2}

Next, we embed a bit of metadata into this interval. The scenario behind the embedding is to move X_{i+1} to the appropriate sub-interval which represents the same bit value as the metadata to be embedded. In particular, according to Figure 5, if we intend to embed a bit of metadata "0", we simply do nothing, since X_{i+1} has located at the status of "0". However, if we mean to embed a metadata "1" into this interval, we need to move X_{i+1} to a point at the right sub-interval representing the status of "1". To ease the expression, we use the symbol X'_{i+1} to indicate that it has conveyed a bit of metadata. Also, X_{i+1} is referred to as the cover "point", while X'_{i+1} is called the stego "point." According to the design, the PCA X-coordinates of the cover vertex will be changed after embedding the metadata.

Surely, if an algorithm does not provide the reversibility feature, moving X_{i+1} to any position on the right sub-interval will allow us to embed one bit of metadata "1". However, to consider the reversible mechanism, we need to design the position on which the "point" X_{i+1} needs to be moved to. Meanwhile, for security reason, we need to record a secret key in order to represent the order of intervals on which we embed a bit. The process left is to where we shall move the cover "point" X_{i+1}. Figures 6 and 7 illustrate all the possible position alternation of a cover "point" X_{i+1}, depending on the metadata to be conveyed.

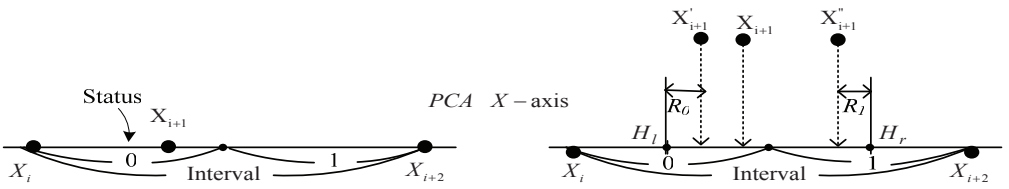

Figure 6: All possible position alternation of cover "point" X_{i+1} where X_{i+1} is originated in the left sub-interval with the status value "0"

In Figure 6, we show the case that embeds a bit of metadata into the cover point "X_{i+1}" within an interval with two boundary "points" X_i and X_{i+2}. In light of the description given above, the cover "point" X_i will become the stego "point" X'_{i+1} after embedding one bit of message. There are two possible cases that are needed to be considered. We can first assume that we intend to embed a bit "0". To convey a metadata we need to shift X_{i+1} to an "appropriate" position so that X_{i+1} becomes X'_{i+1}, indicating that it has now conveyed a metadata of "0". In particular, the final X'_{i+1} is at a position that has a distance R_0 away from H_l. Here, H_l represents the middle position of the left sub-interval, and R_0 is one fourth of the distance between H_l and the original X_{i+1}. Equations (2) and (3) present mathematical expression for R_0 and X'_{i+1}. Using this approach, we convey one secret bit of message "0" at the stego "point" X'_{i+1}.

$$R_0 = (X_{i+1} - H_l) /4 \tag{2}$$

$$X'_{i+1} = H_l + (X_{i+1} - H_l) /4 \tag{3}$$

Now, we consider the case when we intend to embed one bit of metadata "1". Clearly, we need to move the cover "point" X_{i+1} to become a stego "point" X''_{i+1}, which must be located on the right half of the sub-interval with the status of "1". Followed the approach described above, the "appropriate" position of X''_{i+1} has a distance R_1 away from H_r. shown in Equations (4). Again, H_r represents the middle position of the right sub-interval. The new coordinate of X''_{i+1} is shown in Equations (5).

$$R_1 = (X_{i+1} - H_r)\ /4 \tag{4}$$

$$X'_{i+1} = H_r + (X_{i+1} - H_r)\ /4 \tag{5}$$

In contrast to Figure 6, we illustrate in Figure 7 the case when the original cover "point" X_{i+1} is located on the sub-interval with the status value of "1". Similarly, if we intend to embed a metadata of "0", we shift the cover "point" X_{i+1} to X'_{i+1} on the left sub-interval with a distance R_0 away from the middle position of the left sub-interval. Otherwise, we embed a metadata of "1" by moving the stego "point" X_{i+1} to X''_{i+1} on the right sub-interval with a distance R_1 away from the middle position.

We apply the same embedding step to every interval in the odd sorting list. This means that the "points" with the even index X_2, X_4, X_6, ..., X_{n-2} will convey one bit of metadata. With regard to the even sorting list, similarly, we embed one bit of metadata at each "point" X_3, X_5, X_7, ..., X_{n-1}. However, we do not change the first "point" X_1 and the last "point" X_n. We need these two points fixed to ensure that the point in each interval can be reversed to its original position. As a result, we embed a total of (n-2)/2 bits in the odd sorting list and another (n-2)/2 bits in the even sorting list in the 2D cover vector map, where n represents numbers of vertices in the map. Similarly, we embed n-2 bits of message in the PCA Y-axis. As a result, we can embed the total capacity of 2(n-2) bits in a vector map, where n is the amount of vertex number of the map.

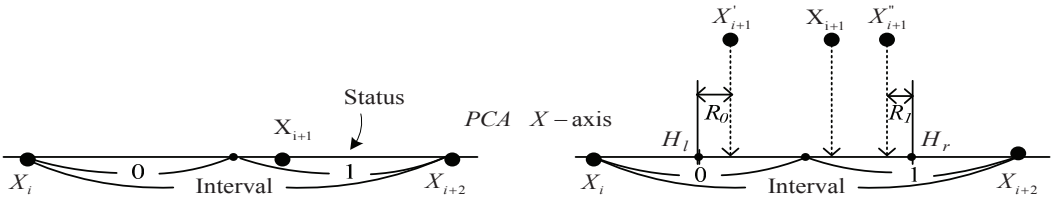

Figure 7: All possible position alternation of cover "point" X_{i+1} where X_{i+1} is originated in the right sub-interval with the status value "1"

Our approach need to hold some side information during the embedding processes. The information required to be held includes two principal axes plus the gravity center of both the cover model and stego model, the length of the bounding volume, and a secret key. The first two are used to against translation, rotating, uniform scaling for the stego map. The last one is to enhance the security of the data hidden in stego map. It is also used for extracting the hidden data from stego map.

4. METADATA EXTRACTION AND RESTORING PROCESSES

There are three phases in the metadata extraction and restoring processes. They are stego map rectifying phase, metadata extraction phase and metadata restoring phase. Figure shows the diagram of these processes. The details of these phases are discussed in this section.

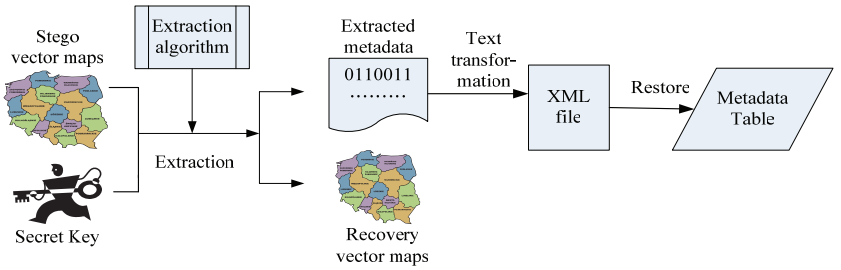

Figure 8: Metadata extracting and restoring processes diagram

4.1 Stego Map Rectifying Phase

Before data extraction, the stego map is needed to be rectified by using two side information derived during metadata embedding process. The side information used for the rectification includes two PCA axes and the gravity center value of both the cover map and stego map. The stego map rectifying phase is used to prevent the errors from displacement and transformation attacks to the stego map.

4.2 Metadata Extraction Phase

Once the stego maps have been rectified, we still need to implement the preprocessing phase which is exactly the same as used in metadata embedding processes. After processing on the stego map, the metadata extraction must begin from the even sorting list first, since we need to ensure that the interval in the even sorting list is reversed to its original form. Once metadata extraction in even sorting list is done, we can then extract the metadata from the odd sorting list.

We use the processing on the PCA X-axis to illustrate the data extraction process. We then take the next three steps for each PCA axis to extract the secret message and produce the recovered cover "point".

Step-1: Find an interval $\{X'_i\text{-}X'_{i+1}\text{-}X'_{i+2}\}$, which has been embedded with the metadata according to the secret key K.

Step-2: Extract a data bit from the position of X'_{i+1} in the interval $\{X'_i\text{-}X'_{i+1}\text{-}X'_{i+2}\}$. In particular, the data bit is set to 0 if X'_{i+1} is located on the left of the sub-interval and 1 otherwise.

Step-3: Restore the recovered cover "point" X_{i+1} by Equation (6), where H_l is the middle position of the left sub-interval. The derivation of Equation (6) is using a simple mathematical transposition from the original embedding expression for the stego "point" X'_{i+1} where $X'_{i+1} = H_l+(X_{i+1}-H_l)/4$. Clearly, we can derive a similar recovered X_{i+1} using the middle position of the right sub-interval H_r.

$$X_{i+1} = 4\times X'_{i+1} - 3H_i \quad (6)$$

Step-4: Repeat Step-1 and 3 above for all the intervals on a given stego model until the embedded data have been extracted from each stego "point" and the cover "point" is recovered. We apply the same processes to the PCA Y-axis to extract the embedded data and recover the cover "point".

Finally, we apply PCA inverse transformation to the coordinate value of each vertex. This transforms the vertices in the PCA coordinate system back to the World coordinate system, generating the recovered map.

4.3 Metadata Restoring Phase

During the metadata extraction process, a binary file is extracted from the stego map. The binary file contains the metadata that had been embedded in cover map. We then transform the binary file to XML file metadata elements. Finally the XML file is restored to metadata table.

5. EXPERIMENTAL RESULTS

We now present our experimental results. The software architecture for implementing the experiments, the factors that are commonly used in GISs to evaluate accuracy of vector maps, the features of the vector maps for our experiments and evaluation of the experimental results are described in this section.

5.1 Software Architecture

We use Java programming language to implement the main program for experiments. Results were collected on a personal computer with a 3.4GHz processor and 2 GB memory. We also use the concept of "mashup" to include some open source programs and application programming interface (API) to develop the system for metadata embedding. The software architecture is shown in Figure 9.

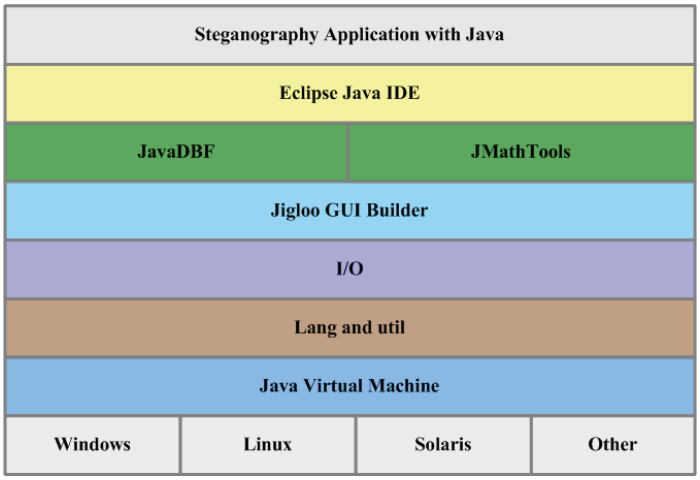

Figure 9: The software architecture of metadata embedding system

The details of each component in the software architecture are described as follows:

- Steganography application with Java: The benefits to use Java language to develop application are such as encapsulation, abstraction, inheritance and polymorphism. Rely on above characters, it is flexible and can reuse the object to reduce develop cost.
- Eclipse Java IDE: It is free, open-source integrated development environment (IDE). And there are many Java-related plug-in for Eclipse.
- JavaDBF: This is a plug-in package for the application. Because metadata can be accessed by DBF file (*.dbf), this package can access format of DBF file.
- JMathTools: This package collection of independent packages designed to fit common engineering or scientific computing needs. To plug-in this package the main function is deal with original coordinate transform to PCA coordinate.
- Jigloo GUI Builder: This is a plug-in for the Eclipse Java IDE which to build and manage both Swing and SWT GUI classes. It implements a set of components for building graphical user interfaces and adding rich graphics functionality and interactivity to Java applications.
- I/O: Provides for system input and output through data streams, serialization and the file system. In the system, the main function is input, read, write cover map, stego map, recover map, and DBF file.
- Lang and util Packages: Provides classes that are fundamental to the design of the Java programming language such as String, Math.
- JVM and Platforms: Programmer wrote a source code must compile to machine code for execute in general program language. But the different situation in Java, the source code compiles result is byte code that machine and operation system

is unknown. Therefore, JVM main decode byte code to machine code for execute program. Besides that, JVM also need to deal with across platform problem for achieving write one run any where.

5.2 Accuracy Evaluation Factors

We performed experiments to evaluate the capacity, imperceptibility, reversibility of the algorithm. We describe more about several important factors for vector maps in the following before we present our experimental results.

Recall that vector maps are one of fundamental data representations for geographic information system (GIS), which depends upon the abstraction and classification of real-world phenomena. Five factors are commonly used in GIS to evaluate the accuracy of 2D vector maps, which include maps scale, map accuracy standard (τ), root mean square error (RMSE), average displacement (σ), and the diagonal length of bounding box (DLB). We describe each factor in more detail.

1. Map scale: The ability to show how detail in a map is determined by the scale. A map with a scale of 1:1000 can illustrate much finer points of data than a smaller scale map of 1:250000. The mathematical expression for the map scale is the ratio of the map distance in a map over and the ground distance on the surface, shown in Equation (7).

$$\text{Map scale} = \text{map distance/ground distance} \tag{7}$$

2. Map accuracy standard (τ): The map accuracy standard is a function of the scale at which a map was created. The Specification and Committee of the American Society for Photogrammetry and Remote Sensing (ASPRS) have developed the Planimetric Coordinate Accuracy Requirement of the term "well-defined point" which pertains to features that can be sharply identified as discrete points (Chao et al. 2009). Points which are not well-defined (that is poorly-defined) are excluded from the map accuracy test. ASPRS defines the limiting root mean square error (RMS error) is 0.025% of a typical ground distance in map scale. As an example, for a typical map scale of 1:20000, the ground distance is 20000 meters, thus the limiting RMS error is 5 meters. Equation (8) is a mathematical expression of the map accuracy standard in meter.

$$\text{Map accuracy standard } (\tau) = \text{ground distance in map scale x } 0.025\% \tag{8}$$

3. Root mean square error (RMS error): The RMS error is defined to be the square root of the average of the squared discrepancies. Since the connectivity of the cover and stego maps as well as cover and recovery maps are identical, the RMS error of each 2D vector

map is calculated to measure the distortion. We use RMSEE to indicate the embedding distortion, which represents the average distortion per vertex between cover and stego vector maps. Similarly, we denote $RMSE_R$ to indicate the recovering distortion, which represents the average distortion per vertex between the recovered map and the original map when applying our reversible algorithm.

4. Average displacement (σ): The average displacement represents the displacement occurs in real world. This is used as a quantitative measure for representing the average error in real world caused by the distortion after embedding secret message to the vector maps. We denote σ_E as the average displacement per vertex between cover and stego vector maps caused by data embedding. Similarly, σ_R represents the average displacement per vertex between the recovered map and the original map used in our reversible algorithm. The equation of this factor is calculated by Equation (9).

$$\text{Average displacement } (\sigma) = \text{RMS error x map scale} \tag{9}$$

5. The Diagonal Length of the Bounding Box (DLB). The DLB represents the diagonal length of the bounding box defined in the transformation phase. The bounding box was constructed with the extremes in the X-axis and Y-axis. The DLB shows the magnitude of vector maps with respect to the center of gravity (G).

5.3 Evaluation of Experimental Results

We now present our experimental results. In our experimental test, we employ three vector maps and employ the above factors to validate the feasibility of proposed methods for metadata embedding in vector maps. Table 5 lists characteristics of three vector maps together with important features. Observing Table 1 indicates that three vector maps have different amounts of vertices, ranging from 2,685 to 9,199 vertices. The Taitung map has the largest Map Scale, which also has the largest Map Accuracy Standard (125 meters) and nearly 155 km of DLB. The Taitung Village map, however, has the smallest Map Scale, containing smallest Map Accuracy Standard (25 meters) and approximately 155 km of DLB. Finally, Pmax represents numbers of digitals that are allowed in the mantissa of the decimal value.

Table 5: Features of three vector maps employed for experiments

Name of Vector Maps	Vertex Amounts	Map Scale	τ (m)	DLB (m)	Pmax
Taitung Village	9,199	1:100000	25	155,127	6
Taitung County	4,905	1:250000	62.5	155,127	6
Taitung	2,685	1:500000	125	155,127	6

Figure 10 shows three vector maps with a proportional scale, measured in meters, which is displayed at the bottom of the image. We present the ground distance for each map in kilometers, which is derived from the first short black line of the proportional scale displayed at the bottom left. As an example, the first short black line in Taitung Village represents approximately 37.5 kilometers of the ground distance.

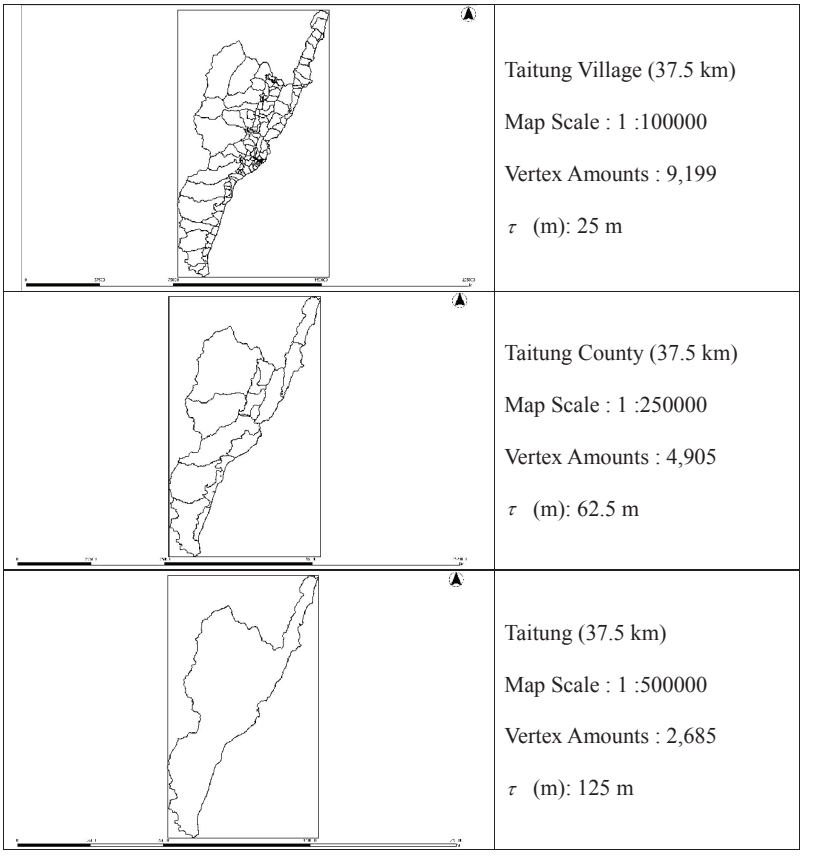

Figure 10: Features of three vector maps employed for experimental test

We present our experimental results by performing the maximum embedding capacity of metadata. There are 2(n-2) bits to be embedded in each vector maps where n is the amount of vertex number of vector maps. The results are evaluated based on the capacity, reversibility and imperceptibility as follows.

1. Capacity Results

Table 6 presents the embedding capacity of our algorithm. Recall that our methods can embed (n-2) bits of metadata in the odd-list and another (n-2) bits in the even-list of a cover vector map, where n represents the amount of vertices in the vector map. This leads to the theoretical data capacity of our algorithm is 2(n-2) bits. The data capacity shown in Table 6

comply the theoretical data capacity. The embedding capacity of our algorithm is nearly 2 bits per vertex (bpv). According to the embedding capacity shown in byte, only Taitung village is capable to embed both mandatory and conditional metadata elements. Taitung county map and Taitung map can only be embedded mandatory metadata elements.

The results show in Table 6 proves that our method has small embedding distortion. The $RMSE_E$ is in the range of 0.000057 and 0.000087. The small range of the $RMSE_E$ implies that while embedding a large amount of metadata, our algorithm produces insignificant distortion in the stego vector map. Also shown in Table 6 are values of the average displacement of the embedding (σ_E) in the stego maps. All σ_E values are smaller than the map accuracy standard. These values show a significant benefit of our methods, which is the positional error caused from the distortion in the stego map is still within the map accuracy standard and the stego map can still be employed in the GIS applications. Finally, observing σ_E and τ allows us to increase the capacity to a vector map which has larger difference between σ_E and τ, because it can still tolerate more distortion due to the embedded metadata.

Table 6: Experimental results for metadata embedding and recovery of vector maps

Name of the Map	Vertex Amounts	Capacity (bits)	Capacity (bytes)	τ (m)	Embedding		Recovery	
					$RMSE_E$	σ_E (m)	$RMSE_R$	σ_R (m)
Taitung Village	9,199	18,394	2299	25.0	0.000087	8.68	3.41E-11	3.41E-06
Taitung County	4,905	9,806	1225	62.5	0.000064	15.97	1.38E-11	3.45E-06
Taitung	2,685	5,366	670	125.0	0.000057	28.38	7.04E-12	3.52E-06

2. Reversibility Results

Our algorithm has the reversibility manner, which means we can produce a recovered vector map once the metadata is extracted. We compare the recovered vector map and the original cover map to check the success of the reversibility that our methods can provided. The final column in Table 6 shows the comparison in terms of the root mean square error for the recovery process (RMSER) and the corresponding average displacement of the recovery (). All values of RMSER are small, being less than 3.41E-11, and values of are smaller than 3.52E-06 meter. These statistics demonstrate that the success of our reversible mechanism. While the recovered vector maps have small root mean square error values, the positional errors occurred in all recovery maps are insignificant and within the map accuracy standard. Surely, the accuracy of recovery maps satisfies the requirements of GIS applications development.

3. Imperceptibility Analysis

Figure 11 illustrates a visualization of the cover vector maps (left) and the stego vector maps (right) for the most complex test vector map-Taitung Village. Within the vector maps, it

has very small embedding distortion (small RMSEE and values). Clearly, observing the images it is hard to identify any difference between cover map and stego map. As a result, our method can be used to embed a large amount of metadata, yet produces insignificant distortion that imperceptible to human visual system.

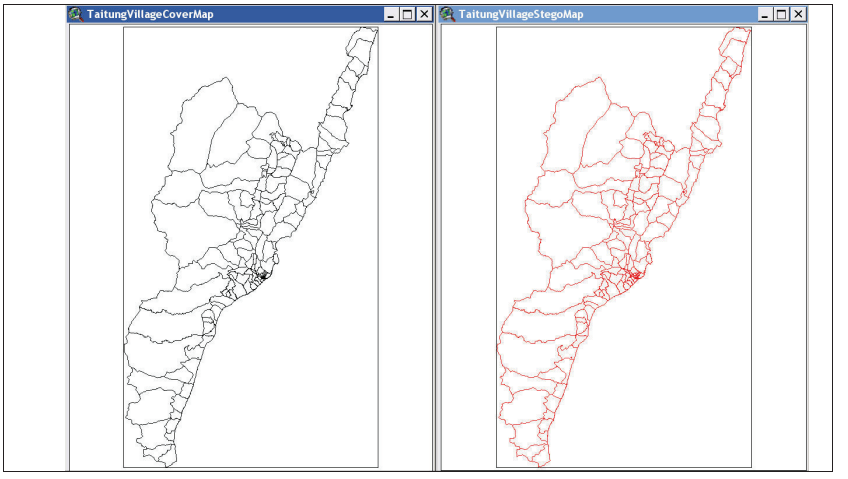

Figure 11: A visualization of the cover vector map (left) and the stego vector maps (right).

6. CONCLUSIONS AND FUTURE WORKS

In this paper, we proposed a metadata internal storage method by using reversible steganographic algorithm to embed metadata in vector maps. Experimental results shows that the maximum metadata embedding capacity of our method is 2(n-2) bits, where n is the amount of vertices in a cover vector map. To the best of our knowledge, this is the highest capaicty achieved in the literature of reversible steganograhy method for vector maps. The results aslo show that our method produces a stego vector map with negligible distortion, being only 0.000057 of the root mean square error. In addition, our method has the capability of reversibility. This means that once the metadata is extracted, we can produce a recovery map with high accuracy that can be used in GIS applications.We only need three secet keys for data extracton, including the gravity of the cover map, the major and minor axes generated by applying the principal component analysis on the cover vector map. Experimental results also show that there is an insignificantly difference between the original and the recovered map, which is less than 3.41E-11 of the root mean square error and is imperceptible to the human visual system. Meanwhile, our method is robust against the affine transformation including translation, rotation uniform scaling, and their combinations. Our method is also secure because embedded metadata can not be extracted correctly without the legal secret keys provided.

In considering to the capacity required by embedding the mandatory and conditional metadata elements of ISO 19115 metadata standard that we have adapted in this paper, a vector map should has at least 5458 vertices so that all mandatory and conditional metadata elements can be embedded in the vector map. Since the conditional elements should not be embedded alone, a vector map should has at least 1998 vertices so that the mandatory metadata elements can be embedded and integrated with the vector map. Any vector map with vertex points less than 1998 is suggested to embed some metadata elements only, such as dataset title, metadata point of contact and the information link of metadata. Furthermore, our method can aslo be used to embed data with less capacity, such as copyright declaration, watermarking information, and user authentication information.

There are some possible future works deserved to be investigated. The first is to extend the method proposed in this paper to increase the embedding capacity. So that the limitation caused by the amount of vertex of vector maps can be improved. The second is to investigate the effects of vector maps' features to our method, such as the complexity of cover vector maps, the smoothness of boundary, and the included angle between vertices. So that better results in embedding capacity and recovery accuracy can be achieved. Finally, a distortion free method for internal metatdata storage mechanism should be considered as the utimate goal of all future works.

Reference

1. Bogomjakov, A., Gotsman, C. and Isenburg, M. "Distortion-Free Steganography for Polygonal Meshes," *Computer Graphics Forum* (27:2), April, 2008, pp. 637-642.
2. Boll, S., Klas, W. and Sheth, A.P. "Overview on Using Metadata to manage Multimedia Data," *Multimedia Data Management*, McGraw-Hill, New York, 1998, pp. 1-17.
3. Butler, D. "Mashups mix data into global service," *Nature* (439:7072), 2006, pp. 6-7.
4. Chao, M. W., Lin, C. H., Yu, C. W. and Lee, T. Y. "A High Capacity 3D Steganography Algorithm," *IEEE Transaction on Visualization and Computer Graphics* (20:3), May/June 2009, pp. 1-11.
5. Chen, K. W., Wang, S. M. and Wang, C. M. "A Reversible Data Hiding Algorithm for 2D Vector Map," *Communications of the CCISA*, 2007,
6. Chen, K. W., Wang, S. M. and Wang, C. M. "A Reversible Data Hiding Algorithm for Vector Maps," *Information Security Conference 2007*, Chiayi, Taiwan, 2007.
7. Cummins, J., Diskin, P., Lau, S. and Parlett, R. *Steganography And Digital Watermark*, 2004. (http://www.cs.bham.ac.uk/~mdr/teaching/modules03/security/students/SS5/Steganography.pdf)
8. Dublin Core Metadata Initiative (DCMI). *Dublin Core Metadata Element Set, Version 1.0: Reference Description, 1998*. (http://dublincore.org/documents/1998/09/dces/#)

9. Federal Geographic Data Committee (FGDC). *Geospatial Metadata*. (2006). (http://www.fgdc.gov/metadata)
10. Fisher, M. "Hotmap: Looking at Geographic Attention," *IEEE transactions on Visualization and Computer Graphics* (13:6), November/December, 2007, pp.1184-1191.
11. Goodchild, M. F., Egenhofer, M. J., Fegeas, R. and Kottman, C. A. "Interoperating Geographic Information Systems," *International Conference and Workshop on Interoperating Geographic Information Systems*, Santa Barbara, California, USA, 1997.
12. Goodman, E. and Moed, A. "Community in Mashups: The Case of Personal Geodata," *Web Mash-ups and CSCW: Opportunities and Issues, The 20th ACM Conference on Computer Supported Cooperative Work*, Banff, Alberta, Canada, 2006.
13. Gunther, O. and Voisard, A. "Metadata in Geographic and Environmental Data Management," *Managing Multimedia Data: Using Metadata to Integrate and Apply Digital Data*, Mc-Graw Hill, 1998, pp. 57-87.
14. International Organization for Standardization (ISO). *ISO 19115:2003* (2003). (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=26020)
16. Kaasinen, E. "User needs for location-aware mobile services," *Personal Ubiquitous Computing* (7:1), 2003, pp. 70-79.
17. Kahn, D. "The history of steganography," *Lecture Notes In Computer Science: Proceedings of the First International Workshop on Information Hiding* (1174), 1996, Springer Berlin / Heidelberg, pp. 1-5.
18. Kashyap, V. and Sheth, A. "Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context and Ontology," *Cooperative Information Systems: Current Trends and Directions*, 1996, Academic Press, pp. 139-178.
19. Kim, H.J., Sachnev, V., Shi, Y. Q., Nam, J. and Choo, H. G. "A Novel Difference Expansion Transform for Reversible Data Embedding," *IEEE Transactions on Information Forensics and Security* (3:3), 2008, pp. 456-465.
20. Mnzis, A. H. "The Road to Ubiquitous Geographic Information Systems Roam Anywhere -Remain Connected," *SIRC 2000-The 12th Annual Colloquium of the Spatial Information Research Centre*, University of Otago, Dunedin, New Zealand, 2000.
21. Petitcolas, F.A.P., Anderson, R.J. and Kuhn, M. G. "Information Hiding-A Survey," *Proceedings of the IEEE, special issue on protection of multimedia content* (87:7), 1999, pp. 1062-1078.
22. Pfitzmann, B. "Information Hiding Terminology - Results of an Informal Plenary Meeting and Additional Proposals," *Lecture Notes In Computer Science* (1174), Springer-Verlag, London, 1996.
23. Popa, R. "An Analysis of Steganographic Techniques," 1998. (http://ad.informatik.uni-freiburg.de/mitarbeiter/will/dlib_bookmarks/digital-watermarking/popa/popa.pdf)
24. Schulz, G. and Voigt, M. "A high capacity watermarking system for digital maps," *ACM*

International. Workshop on Multimedia and Security, Magdeburg, Germany, 2004, pp. 180-186.

25. Shao, C. Y., Wang, H. L., Niu, X. M. and Wang, X. T. "A Shape-Preserving Method for Watermarking 2D Vector Maps Based on Statistic Detection," *IEICE-Transactions on Information and Systems*, 2006, pp. 1290-1293
26. Tan, X., Zhou, M., Zuo, X. and Cui, Y. "Integration WebGIS with AJAX and XML Based on Google Maps," *The First International Conference on Intelligent Networks and Intelligent Systems*, 2008, pp. 376-379.
27. Tian, J. "Reversible data embedding using difference expansion," *IEEE Transactions on Circuits Systems and Video Technology* (13:8), 2003, pp. 890-896.
28. Turner, J. M. "Data about metadata: beating the MetaMap into shape," *American Society for Information Science & Technology, Special Interest Group on Classification Research (SIG/CR) Workshop*, 2004.
29. Voigt, M., Yang, B. and Busch, C. "Reversible watermarking of 2d-vector data," *ACM Int. Workshop on Multimedia and Security*, Magdeburg, Germany, 2004, pp. 160-165.
30. Wang, C. M. and Wang, P. C. "Steganography on Point-Sampled Geometry," *Computers & Graphics* (30:2), 2006, pp. 244-254.
31. Wang, S. M. and Chen, F. M. "The Development of a Ubiquitous Geographic Information Service by Using Service-Oriented Architecture," *IEEE Asia-Pacific Services Computing Conference*, Yilan, Taiwan, 2008, pp. 453-457.
32. Wang, X. T., Shao, C. Y., Xu, X. G. and Niu, X. M. "Reversible Data-Hiding Scheme for 2-D Vector Maps Based on Difference Expansion," *IEEE Transactions on information forensics and security* (2:3), 2007, pp. 311-320.
33. Yu, J., Benatallah, B., Casati, F. and Daniel, F. "Understanding Mashup Development," *IEEE Internet Computing* (12:5), 2008, pp. 44-52.
34. Zhang, D., Qian, D., Wu, W., Liu, A., Yang, X. and Han, P. "Spatial Map Data Share and Parallel Dissemination System Based on Distributed Network Services and Digital Watermark," *Network and Parallel Computing* (4672/2008), Berlin / Heidelberg, Springer, 2008.