ABSTRACT

Title of Dissertation: DIGITAL FORENSIC TECHNIQUES FOR GRAPHIC DATA

Hongmei Gou, Doctor of Philosophy, 2007

Dissertation directed by: Professor Min Wu
Department of Electrical and Computer Engineering

With rapid development of hardware devices and software programs, a large amount of graphic data has been brought to or generated in digital domain, and become increasingly more widely used in our everyday life. Due to the ease of editing and distributing graphic data in the digital domain, protecting graphic data from such fraudulent operations as malicious tampering and unauthorized copying is becoming a major concern. The primary motivation of this dissertation research is to develop novel forensic techniques for digital graphic data to facilitate its proper distribution, authentication, and usage. We investigate two complementary mechanisms for performing forensic analysis on graphic data, namely, the extrinsic and intrinsic approaches.

In the extrinsic approaches, we seamlessly embed into graphic data *extrinsic* watermarks/fingerprints, which shall later be extracted for verifying authenticity

or tracing leak of the graphic data. By utilizing such extrinsic techniques via data embedding, we have studied robust digital fingerprinting for curve-based graphics such as topographic maps and drawings, in which a unique ID referred to as a *digital fingerprint* is robustly embedded for tracing traitors. Through proper transformations between 2-D contour curves and 3-D digital elevation maps, we have also developed an effective fingerprinting technique for digital elevation maps. In order to authenticate such graphic data as critical document and signature images, we have investigated high-payload watermark embedding for binary images, whose authenticity shall be decided through verifying integrity of the hidden watermark.

In the intrinsic approaches, since scanners are a major kind of apparatus to capture graphic data, we develop a new technique of utilizing *intrinsic* sensor noise features for non-intrusive scanner forensics to verify the acquisition source and integrity of digital scanned images. We extract statistical features of scanning noise from scanned image samples, and construct a robust scanner identifier to determine the model of the scanner used to capture a scanned image. We further broaden the scope of acquisition forensics to differentiating scanned images from camera taken images and computer generated images, as well as perform integrity forensic analysis on scanned images using the proposed noise features, including detecting post-processing operations after scanning, and implementing steganalysis on scanned images.

DIGITAL FORENSIC TECHNIQUES

FOR GRAPHIC DATA

by

Hongmei Gou

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

Advisory Committee:

Professor Min Wu, Chair / Advisor
Professor Rama Chellappa
Professor K. J. Ray Liu
Professor Isaak D. Mayergoyz
Professor Yang Tao

DEDICATION

To my parents and Yong.

ACKNOWLEDGEMENTS

With this thesis drawing a period for my education in schools, I first would like to express my sincere gratitude to my advisor, Prof. Min Wu, for her constant guidance, encouragement, and support during my graduate study at University of Maryland. She has played a significant role in both my professional and personal development. Her knowledge, vision, patience, enthusiasm, and endless pursuit of excellence have influenced me with lifetime benefits. I deeply appreciate her for helping me reach this milestone in my life and advising me to be a real researcher and professional.

I would like to thank Prof. K. J. Ray Liu for his vision and support in helping my research endeavors, as well as for introducing me to the field of signal processing through the two excellent courses I took from him. I also would like to thank Prof. Rama Chellappa, from whom I have taken multiple courses in image processing and machine learning, which have brought great benefits to my research. I also would like to thank Prof. Isaak D. Mayergoyz and Prof. Yang Tao for serving in my dissertation committee.

I am grateful to Dr. Jessica Fridrich for the enjoyable discussions and collaborations. The interaction with her and her research group has broadened my research scope and experience. I appreciate Dr. Bertrand Haas and Dr. Robert Cordery for their kindness, support, and collaborations during my internships at Pitney Bowes in summer 2005 and 2006.

I own my gratitude to my colleagues in the MAST group. I would like to thank Dr. Guan-Ming Su, Dr. Yinian Mao, Dr. Meng Chen, Shan He, Ashwin Swaminathan, and Avinash L. Varna for their friendship, unselfish help, and many constructive suggestions on my research. Special thanks to Ashwin Swaminathan for the enjoyable collaborations.

I am grateful to Yong She, my husband, for his love and encouragement during the past years. He is always the wind beneath my wings. Finally, I give my heartfelt gratitude to my parents, who have been giving me unconditional support and full understanding during my whole journey of study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

Graphic data is common in our everyday life. It is often obtained through writing, drawing, and engraving[1]. With rapid development of hardware devices and software programs, a large amount of graphic data has been brought to the digital domain, and become increasingly more widely used in our everyday life. For example, owing to the popularity of scanning devices and pen-based devices, we have seen a large volume of scanned documents and photos, digitized signatures, and digital writings and drawings. The Check Clearing for the 21st Century Act (Check 21), a U.S. federal law effective in 2004, allows banks to electronically exchange scanned check images so that transactions can be performed faster and more efficiently than physically transporting paper checks [1]. With the advancement of software, a lot of graphic data is also directly generated from various computer programs, such as vector/raster maps from map-making software, engineering graphs from computer-aided design (CAD) systems, cartoons from cartoon/2-D anima-

[1]http://www.m-w.com/dictionary/graphical

tion software. In this dissertation, we target at various graphic data and develop novel forensic techniques to facilitate its proper distribution and authentication.

With the wide availability and usage of a large amount of graphic data, it is of utmost importance to protect it from various fraudulent operations. Different kinds of graphic data in various applications raise different requirements on their protection. Such graphic data as military maps contain information with high sensitivity, and therefore prompt the need for preventing information leak and tracing the source of leak. Civilian maps as well as CAD drawing and cartoons often have high commercial values, and the protection from unauthorized copying and illegal re-distribution is highly desirable. Such graphic data as digitized signatures and scanned checks/photos provides evidence in financial systems and/or law enforcement, and therefore it becomes particularly important to establishing its authenticity and/or identifying its acquisition source. Meanwhile, digital graphic data can be easily edited by computer programs, and distributed through computer networks. For example, the World Wide Web Consortium (W3C) developed a standard known as Scalable Vector Graphics (SVG) [2] to facilitate the transmission and sharing of graphic data. The ease of editing and dissemination of graphic data makes the developing of digital forensic techniques more urgent and important.

1.2 Extrinsic and Intrinsic Approaches

In this section, we describe several techniques related to the emerging field of digital forensic research. We focus on two complementary mechanisms that can be utilized for performing forensic analysis on graphic data, namely, the *extrinsic* approaches via data embedding and the *intrinsic* detection via inherent sensor

features.

Extrinsic techniques via data embedding are promising building blocks to construct digital forensic systems for graphic data. Through embedding extrinsic watermarks/fingerprints seamlessly into the graphic data, we can protect different kinds of graphic data in an application-dependent way. For example, to protect graphic data with sensitive information and/or high commercial values (such as maps and drawing) from being leaked or illegally re-distributed, we can embed a unique ID referred to as a *digital fingerprint* into each copy of the graphic data to represent the recipient's identity [19, 92]. When a suspicious copy of the graphic data appears, the fingerprint embedded before can then be extracted for tracing the source of leak. The extrinsic data embedding technique can also be employed to verify the authenticity of such graphic data as digitized signatures and critical scanned documents (such as scanned checks). A pre-determined pattern or some content feature is taken as a digital watermark and then seamlessly embedded into the host graphic data. When the watermarked graphic data is tampered, the integrity of its hidden data shall be violated correspondingly, providing the evidence of tampering [88, 95].

Complementary to the extrinsic technique via data embedding, some recent research on digital forensics are utilizing intrinsic sensor features which are directly associated with the acquisition device of the host data. While extrinsic approaches require some hidden data be inserted into the host data beforehand for future forensic analysis, the intrinsic approaches aim at detecting inherent sensor features non-intrusively by using only host data samples. One example is the component forensic methodology introduced in [81], where some parameters obtained from camera modelling, such as the color filter array (CFA) pattern and the CFA

interpolation coefficients, are used to determine the brand/model of the camera used to capture a photographic image. This camera identifier has been further used to detect multi-segment image splicing by assuming that different segments in a doctored image come from different camera models [80]. It has been also employed to identify post-processing operations on a direct camera output by verifying if a test image violates the conditions imposed by camera modelling [82]. Such non-intrusive method of detecting and leveraging intrinsic sensor features is also beneficial for forensic analysis on such graphic data as scanned images. Certain sensor features unique to scanners shall be identified, and then used to verify the source and integrity of scanned images.

1.3 Thesis Organization and Contribution

This dissertation focuses on studying novel forensic techniques for several important types of graphic data. We investigate the two complementary mechanisms described in Section 1.2, constructing data-embedding based forensic systems for curve-based graphic data, digital elevation maps(DEMs), and binary graphic data, as well as identifying intrinsic sensor features of scanners for verifying the source and integrity of scanned images. The main challenge in forensic analysis on various graphic data is to identify the characteristics unique to each type of graphic data, and then to develop the extrinsic data embedding algorithms or the intrinsic sensor features in a way of tailoring to those unique characteristics.

This dissertation is organized as follows. The first part consists of Chapter 2 to Chapter 4 and focuses on developing extrinsic data embedding algorithms for various graphic data for tracing information leak or verifying data authenticity. Specifically, Chapter 2 studies a robust data embedding method for curves and

investigates its feasibility for digital fingerprinting of such curve-based graphic data as topographic maps and writhings/drawings for traitor tracing. Leveraging a curve modeling method, the B-splines, we achieve high-fidelity curve embedding as well as effective fingerprint detection with strong robustness against a number of challenging attacks. Based on the work presented in Chapter 2, Chapter 3 extends the robust digital fingerprinting of 2-D curves to 3-D digital elevation maps (DEMs), through proper transformations between a 3-D DEM and its 2-D contours. The proposed method enables reliable cross-dimensional fingerprint detection from both the 3-D DEM data set and its 2-D rendering, whichever format that is available to a detector. In Chapter 4, we study the problem of high-payload data embedding for binary images to facilitate the authentication of such graphic data as critical document and signature images. Incorporating a recently proposed steganography approach known as "wet paper coding", we naturally handle the uneven embedding capacity issue in binary graphic data and greatly increase the embedding payload. To maintain good perceptual quality of watermarked binary images under a high embedding payload, we further develop an adaptive trimming method and also introduce a new concept of *super-pixels* to address the pixel flippability issue unique to binary graphic data.

The second part of this dissertation uses scanners as examples to discuss the intrinsic technique via sensor features. We explore in Chapter 5 a novel technique of detecting inherent scanner features to establish the trust-worthiness of scanning devices and scanned images. We extract from scanned image samples informative scanner noise features, building upon which we first construct a robust scanner identifier to determine the model of the scanner used to capture individual scanned images. We further broaden the scope of acquisition source forensics

to differentiating scanned images from camera taken images and computer generated images. Leveraging the intrinsic scanner noise features, we finally perform integrity forensic analysis on scanned images, including detecting post-processing operations after scanning and implementing steganalysis on scanned images.

Finally, the dissertation is concluded in Chapter 6, with discussions on future perspectives.

Chapter 2

Robust Digital Fingerprinting of Curves for Traitor Tracing

In this chapter, we present a new, robust data hiding technique for curves and investigate its feasibility for fingerprinting curve-based graphic data, such as maps, without interfering with its meaningful content. We parameterize a curve using the B-spline model, select B-spline control points of the curve as the feature domain, and add spread spectrum sequences as digital fingerprints to the coordinates of the B-spline control points. A proper set of B-spline control points forms a compact collection of salient features representing the shape of the curve, which is analogous to the perceptually significant components in continuous-tone images [20]. The shape of curves is also invariant to such challenging attacks as printing-and-scanning and format conversions. The additive spread spectrum embedding and the corresponding correlation-based detection generally provide a good tradeoff between imperceptibility and robustness [20], especially when the original host signal is available to the detector, as in most of the digital fingerprinting applications [92].

To determine which fingerprint sequence(s) is(are) present in a test curve, its

registration with the original unmarked curve is an indispensable preprocessing step. B-splines have invariance to affine transformations in that the affine transformation of a curve is equivalent to applying the same affine transformation to its B-spline control points. This affine invariance property of B-splines can facilitate automatic curve registration. Meanwhile, as a curve can be approximated by different sets of B-spline control points, we propose an iterative alignment-minimization (IAM) algorithm to simultaneously align the curves and identify the corresponding B-spline control points with high precision. Through the B-spline based data hiding as well as the IAM algorithm for robust fingerprint detection, the proposed data embedding technique can sustain a number of challenging attacks on marked curves, such as collusion, cropping, geometric transformations, vector/raster-raster/vector conversions, and printing-and-scanning, and is therefore capable of building an effective fingerprinting system for maps and other curve-based graphic data.

2.1 Introduction

Maps are graphic data representing geospatial information ubiquitous in government, military, intelligence, and commercial operations. The traditional way to protect a map from unauthorized copying and distribution is to place deliberate errors in the map as the trademark, such as spelling "Nelson Road" as "Nelsen Road", bending a road in a wrong way, and/or placing a non-existing pond. If an unauthorized user has a map containing basically the same set of errors, this is a strong piece of evidence for piracy that can be presented in court. One of the classic lawsuits is the Rockford Map Pub. vs. Dir. Service Co. of Colorado, 768 F.2d 145, 147 (7th Cir., 1985), where phony middle initials of names in a map

spelled out "Rockford Map Inc." when read from the top of the map to the bottom, and thus copyright infringement was found. However, such traditional protection methods alter the geospatial meanings conveyed by a map, which can cause serious problems in critical government, military, intelligence, and commercial operations that require high-fidelity geospatial information. Furthermore, in the situations where distinct errors serve as fingerprints to trace individual copies, the deliberately placed errors can be easily identified and removed in the digital domain by comparing several map copies with different errors placed in. To overcome all these limitations of the traditional methods, it is desirable to develop some modern ways that can be more effective and less intrusive.

Curves are one of the major components appearing in maps and also in some other graphic data such as writings and drawings. Owing to the popularity of scanning devices and pen-based devices (such as Tablet PCs), a huge amount of curve-based graphic data are being brought to the digital domain. Digital maps and drawings are also generated directly by various computer programs, such as map-making software and computer-aided design systems. Many of these curve-based graphic data, such as maps and CAD drawings, contain information with high sensitivity and/or high commercial values, and therefore prompt the need of preventing information leak in military or unauthorized copying and re-distribution in commerce. Data embedding techniques are promising building blocks to construct digital forensics systems that can protect curve-based graphic data from illegal re-distribution. Before distributing each copy of the graphic data, the authority embeds into it a unique digital fingerprint to represent the recipient's identity. When some recipients leak their copies and these leaked copies are acquired by the authority, the sources of the leak can be identified by examining what IDs

are contained in the suspicious copies.

To be an effective forensic mechanism for tracing the leak source, digital fingerprints must be difficult to remove. This requires the data embedded in curves be robust against common processing and malicious attacks. One example is collusion, where several users combine information from several copies of the protected graphic data, each having the same content but a different fingerprint, so as to generate a new copy in which the original fingerprints are removed or attenuated [92]. Some other distortions/attacks include various geometric transformations such as rotation, scaling, and translation (RST), raster/vector format conversion, and D/A-A/D conversions such as printing-and-scanning. At the same time, the fingerprint should be embedded in a non-intrusive way to preserve the information to be conveyed with high precision. This is because intrusive changes may have serious consequences in critical military and commercial operations, for example, when inaccurate data is given to troops or fed into navigation systems.

There are a very limited amount of existing works on data embedding for maps and graphic data [12], and few works exploit curve features or address fingerprinting issues. As for map watermarking, a text-based geometric normalization method was proposed in [7], whereby text labels are first used to normalize the orientation and scale of the map image, and conventional robust watermarking algorithms for grayscale images are then applied. Since curve-based graphic data can be represented as a binary bitmap image (known as the raster representation) or a set of vectors (known as the vector representation), we review the related prior art from these two aspects accordingly. As for data embedding in general binary images, most of the existing works [52,66,71,90,91] are intended for the purpose of tamper detection. The fragility of these embedding techniques and the dependence on pre-

cise sampling of pixels for correct decoding pose challenges in surviving geometric transformations, printing-and-scanning, and malicious removal in fingerprinting applications. A few other works embed information in dithered images by manipulating the dithering patterns, in fax images by manipulating the run-length [64], and in formatted textual images by changing the line spacing and character spacing [65]. However, these works cannot be easily extended to curve-based graphic data. As for data embedding in vector graphics, two related works perturb vertices through Fourier descriptors of polygonal lines [76] or spectral analysis of mesh models [70] to embed copyright marks. However, the embedding in [76] introduces visible distortions. The approach in [70] has high complexity resulting from the mesh spectral analysis, and it cannot be easily applied to maps beyond urban areas, where curves become essential components in mapping a vast amount of land and underwater terrains.

Several data embedding algorithms on graphic data explore compact representation of curves or surfaces, such as through the non-uniform rational B-spline (NURBS) model. The work in [69] concerns how to embed data in NURBS curves and surfaces without changing the shape or increasing the number of B-spline parameters. The approach demonstrated in [69] relies on reparameterizing a curve or surface using a rational linear function that has an offset determined by the bits to be embedded. The embedded data is fragile and can be removed by perturbing the NURBS parameters or another round of reparameterization. The work in [55, 56] focuses on 3-D surfaces and extracts NURBS features from a 3-D surface to form a few 2-D arrays. Through DCT-domain embedding in these virtual images, a watermark is embedded into the 3-D NURBS surfaces. The work in [74] employs a different domain for 3-D surfaces through multiresolution mesh modelling and

embeds a spread spectrum watermark by perturbing the mesh vertices along the direction of the surface normal. Registration techniques for 3-D NURBS surfaces, such as [51], may be employed to facilitate the alignment of the test surfaces with the original reference surface prior to watermark detection. These prior works provide enlightening analogies for watermarking 2-D curves in the B-spline feature domain. However, as most existing exploration either has limited robustness or targets mainly 3-D surfaces, there are few discussions on robust fingerprinting of curves. To our best knowledge, no existing data embedding work has demonstrated the robustness against curve format conversions and D/A-A/D conversions, or addressed collusion resistance and traitor tracing issues for curves.

In the remaining of this chapter, we present our robust data embedding technique for curves by identifying and manipulating curve parameters based on the B-spline model [36–38]. Section 2.2 discusses the feature domain of B-spline control points, and presents the basic embedding and detection algorithms with experimental results on marking simple curves. To achieve robust fingerprint detection, Section 2.3 develops an iterative alignment-minimization algorithm to perform curve registration and to address non-uniqueness of B-spline control points simultaneously. Experimental results on fingerprinting topographic maps are presented in Section 2.4 to demonstrate the robustness of our method against a number of distortions and attacks. Finally we summarize this chapter in Section 2.5.

2.2 Basic Embedding and Detection

Our proposed algorithm employs B-spline control points of curves as the feature domain, and adopts spread spectrum embedding [20] for robustly watermarking the coordinates of the control points. The fingerprints for different users are approxi-

mately orthogonal and are generated by a pseudo-random number generator with different keys, and the detection is based on correlation statistics. In the following subsections, we explain the main issues of the basic embedding and detection in detail.

2.2.1 Feature Extraction

A number of approaches have been proposed for curve modelling, including using chain codes, Fourier descriptors, autoregressive models, and B-splines [46]. Among them, B-splines are particularly attractive and have been extensively used in computer-aided design and computer graphics. This is mainly because the B-spline model provides a continuous approximation of a curve with excellent local shape control and is invariant to affine transformations [25]. These advantages also lead to our choosing B-splines as the feature domain for embedding data in curves.

B-splines are piecewise polynomial functions that provide local approximations of curves using a small number of parameters known as the *control points* [46]. Let $\{\mathbf{p}(t)\}$ denote a curve, where $\mathbf{p}(t) = (p_x(t), p_y(t))$ and t is a continuous indexing parameter. Its B-spline approximation $\{\mathbf{p}^{[B]}(t)\}$ can be written as

$$\mathbf{p}^{[B]}(t) = \sum_{i=0}^{n} \mathbf{c}_i B_{i,k}(t), \tag{2.1}$$

where t ranges from 0 to $n-1$, $\mathbf{c}_i = (c_{x_i}, c_{y_i})$ is the i^{th} control point ($i = 0, 1, \ldots, n$), and $B_{i,k}(t)$ is the weight of the i^{th} control point for the point $\mathbf{p}^{[B]}(t)$ and known as the k^{th} order B-spline blending function. $B_{i,k}(t)$ is recursively defined as

$$B_{i,1}(t) = \begin{cases} 1 & t_i \leq t < t_{i+1} \\ 0 & otherwise, \end{cases}$$

$$B_{i,k}(t) \quad = \quad \frac{(t-t_i)B_{i,k-1}(t)}{t_{i+k-1}-t_i} + \frac{(t_{i+k}-t)B_{i+1,k-1}(t)}{t_{i+k}-t_{i+1}}, \qquad k = 2, 3, ..., \quad (2.2)$$

where $\{t_i\}$ are parameters known as *knots* and represent locations where the B-spline functions are tied together [46]. The placement of knots controls the form of B-spline functions and in turn the control points.

As a compact representation, the number of B-spline control points necessary to represent a curve at a desired precision can be much smaller than the number of points that can be sampled from the curve. Thus, given a set of samples on the curve, finding a smaller set of control points for its B-spline approximation that minimizes the approximation error to the original curve can be formulated as a least-squares problem. Coordinates of the $m+1$ samples on the curve can be represented as an $(m+1) \times 2$ matrix

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \cdots \\ \mathbf{p}_m \end{bmatrix} = \begin{bmatrix} p_{x_0} & p_{y_0} \\ p_{x_1} & p_{y_1} \\ \cdots & \cdots \\ p_{x_m} & p_{y_m} \end{bmatrix} \triangleq (\mathbf{p}_x, \mathbf{p}_y)\,. \quad (2.3)$$

The indexing values of the B-spline blending functions corresponding to these $m+1$ samples are $t = s_0, s_1, s_2, \ldots, s_m$, where $s_0 < s_1 < s_2 < \cdots < s_m$. Further, let $\mathbf{C}$ represent a set of $n+1$ control points

$$\mathbf{C} = \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \cdots \\ \mathbf{c}_n \end{bmatrix} = \begin{bmatrix} c_{x_0} & c_{y_0} \\ c_{x_1} & c_{y_1} \\ \cdots & \cdots \\ c_{x_n} & c_{y_n} \end{bmatrix} \triangleq (\mathbf{c}_x, \mathbf{c}_y)\,. \quad (2.4)$$

Then we can write the least-squares problem with its solution as

$$\min_{\mathbf{C}} \|\mathbf{BC} - \mathbf{P}\|^2 \quad \Longrightarrow \quad \mathbf{C} = \left(\mathbf{B^T B}\right)^{-1}\mathbf{B^T P} = \mathbf{B}^{\dagger}\mathbf{P}, \quad (2.5)$$

where $\{\mathbf{B}\}_{ji}$ is the value of the k^{th}-order B-spline blending function $B_{i,k}(t)$ in (2.2) evaluated at $t = s_j$ for the i^{th} control point and † denotes the pseudo inverse of a matrix. Because of the natural decoupling of the x and y coordinates in the B-spline representation, we can solve the problem separately along each of the two coordinates as

$$\begin{cases} \min_{\mathbf{c}_x} \|\mathbf{B}\mathbf{c}_x - \mathbf{p}_x\|^2 \\ \min_{\mathbf{c}_y} \|\mathbf{B}\mathbf{c}_y - \mathbf{p}_y\|^2 \end{cases} \Longrightarrow \begin{cases} \mathbf{c}_x = \mathbf{B}^{\dagger}\mathbf{p}_x \\ \mathbf{c}_y = \mathbf{B}^{\dagger}\mathbf{p}_y \end{cases} . \tag{2.6}$$

2.2.2 Embedding and Detection in Control-Point Domain

Control points of a curve are analogous to perceptually significant components of a continuous-tone image [20] in that they form a compact set of salient features for curves. In such a feature domain, we apply spread spectrum embedding and correlation-based detection, as shown in Fig. 2.1.

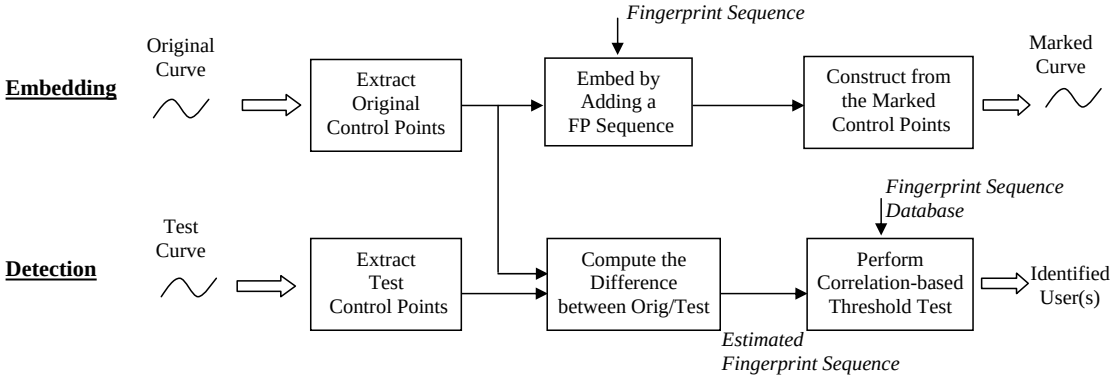


Figure 2.1: The basic embedding and detection process of data hiding in curves.

In the embedding, we use mutually independent, noise-like sequences as digital fingerprints to represent different users/IDs for trace and track purposes. As each of the $n + 1$ control points has two coordinate values x and y, the overall length of the fingerprint sequence is $2(n + 1)$. To apply spread spectrum embedding

on a curve, we add a scaled version of the fingerprint sequence $(\mathbf{w}_x, \mathbf{w}_y)$ to the coordinates of a set of control points obtained from the previous subsection. This results in a set of watermarked control points $\left(\mathbf{c}'_x, \mathbf{c}'_y\right)$ with

$$\begin{cases} \mathbf{c}'_x = \mathbf{c}_x + \alpha \mathbf{w}_x \\ \mathbf{c}'_y = \mathbf{c}_y + \alpha \mathbf{w}_y \end{cases}, \tag{2.7}$$

where α is a scaling factor adjusting the fingerprint strength. A watermarked curve can then be constructed according to the B-spline synthesis equation (2.1) using these watermarked control points.

To determine which fingerprint sequence(s) is(are) present in a test curve, we first need to perform registration using as a reference the original unmarked curve that is commonly available to a detector in fingerprinting applications. After registration, control points $(\tilde{\mathbf{c}}_x, \tilde{\mathbf{c}}_y)$ are extracted from the test curve. The accurate registration and correct extraction of control points are crucial to the detection of fingerprints, which will be detailed in Section 2.3. Assuming we have the set of sample points given by $(\tilde{\mathbf{p}_x}, \tilde{\mathbf{p}_y}) = (\mathbf{B}(\mathbf{c}_x + \alpha \mathbf{w}_x), \mathbf{B}(\mathbf{c}_y + \alpha \mathbf{w}_y))$, we can extract the test control points $(\tilde{\mathbf{c}_x}, \tilde{\mathbf{c}_y})$ from $(\tilde{\mathbf{p}_x}, \tilde{\mathbf{p}_y})$ using (2.6). After getting $(\tilde{\mathbf{c}}_x, \tilde{\mathbf{c}}_y)$, we compute the difference between the coordinates of the test and the original control points to arrive at an estimated fingerprint sequence

$$\begin{cases} \tilde{\mathbf{w}_x} = \frac{\tilde{\mathbf{c}_x} - \mathbf{c}_x}{\alpha} \\ \tilde{\mathbf{w}_y} = \frac{\tilde{\mathbf{c}_y} - \mathbf{c}_y}{\alpha}. \end{cases} \tag{2.8}$$

The estimated fingerprint sequence consists of one or several users' contributions as well as some noise coming from distortions or attacks. The problem of finding out which user(s) has(have) contributed to the estimated fingerprint can be formulated as hypothesis testing [85], which is commonly handled by evaluating the similarity between the estimated fingerprint sequence and each fingerprint

sequence in the database through a correlation-based statistic. Various correlation-based statistics share a kernel term that measures the total correlation , and differ in how they are normalized. To facilitate the evaluation of detection performance, we often normalize the detection statistic to make it have a unit variance and follow approximately a Gaussian distribution under distortions and attacks. There are several ways to do so [97], for example, to normalize using the product of the noise's standard deviation and the watermark's L_2 norm, or through a logarithm-based transformation to be introduced next.

2.2.3 Z statistic for Detection

A nonlinear function of the sample correlation coefficient from the mathematical statistics literature [10] was introduced to the watermarking community by Stone and colleagues of the NEC Research Institute [78]. This is often referred to as the *Fisher's Z statistic*. Among several correlation-based statistics analyzed and compared in [97], the Z statistic shows excellent robustness against different collusion attacks and does not require the explicit estimation of the noise's variance. These advantages make it attractive to handle our problem.

The Z statistic originated from the statistical problem of sampling a bivariate normal population [10], i.e. to obtain i.i.d. samples from a pair of random variables that are jointly Gaussian distributed, with correlation coefficient ρ unknown to the observers. Let r be the sample correlation coefficient computed from L pairs of sample data, and $-1 \leq r \leq 1$. The following function of r

$$\frac{1}{2} \log \frac{1+r}{1-r}$$

has been shown [10] to asymptotically follow a normal distribution when $L \to \infty$, with the mean approximating $\frac{1}{2} \log \frac{1+\rho}{1-\rho}$ and the variance approximating $1/(L-3)$.

Thus, the Z statistic defined below follows a unit-variance Gaussian distribution:

$$Z \triangleq \frac{\sqrt{L-3}}{2} \log \frac{1+r}{1-r} \sim \mathcal{N}\left(\frac{\sqrt{L-3}}{2} \log \frac{1+\rho}{1-\rho}, 1 \right). \tag{2.9}$$

The approximation is found excellent with as few as 10 pairs of samples.

In our fingerprint detection problem in the control-point domain, the effective number of samples for computing the statistic is $L = 2(n+1)$. Denoting the average values of the components in $\tilde{\mathbf{w}}$ and $\mathbf{w}$ as $\tilde{\mu}$ and μ, respectively, we compute the sample correlation coefficient r between $\tilde{\mathbf{w}}$ and $\mathbf{w}$ by

$$r = \frac{\sum_{i=1}^{L} (\tilde{w}_i - \tilde{\mu})(w_i - \mu)}{\sqrt{\sum_{j=1}^{L} (\tilde{w}_j - \tilde{\mu})^2 \sum_{k=1}^{L} (w_k - \mu)^2}}. \tag{2.10}$$

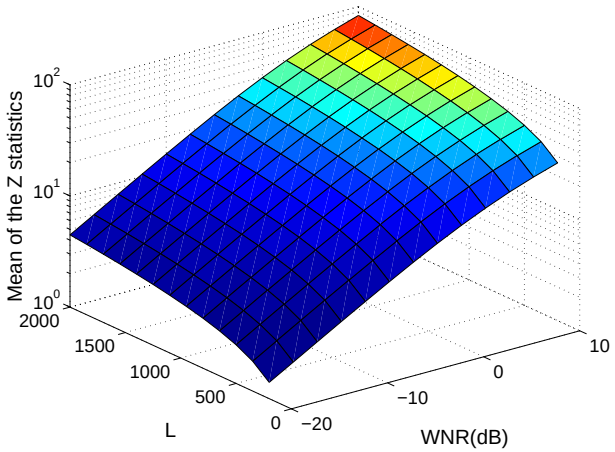
A simplified model considers that the corresponding components of an extracted fingerprint in question and a particular user Alice's fingerprint are i.i.d. samples from a bivariate normal population. When the extracted fingerprint does not have Alice's contribution, the expected correlation coefficient is zero, and the Z statistic will approximately follow a Gaussian distribution with a zero mean and a unit variance. When the fingerprint has Alice's contribution, the Z statistic will have a large positive mean determined by the correlation coefficient ρ. To derive the expression for ρ, we define a random variable $Y \triangleq W + N$ and the extracted fingerprint consists of i.i.d. samples from Y. Here, W is a zero-mean Gaussian random variable representing Alice's fingerprint, and N is a zero-mean Gaussian random variable representing noise. The correlation coefficient of this bivariate normal population (W, Y) is

$$\rho = \frac{cov(W,Y)}{\sqrt{Var(W)Var(Y)}} = \frac{Var(W) + cov(W,N)}{\sqrt{Var(W)\left[Var(W) + Var(N) + 2cov(W,N)\right]}}. \tag{2.11}$$

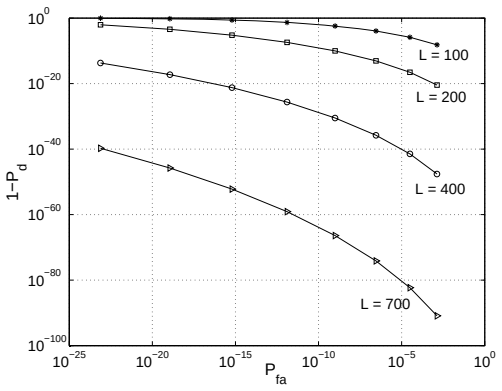
When W and N are uncorrelated, the correlation coefficient becomes

$$\rho = \sqrt{\frac{Var(W)}{Var(W) + Var(N)}} = \sqrt{\frac{1}{1 + \frac{1}{WNR}}}, \tag{2.12}$$

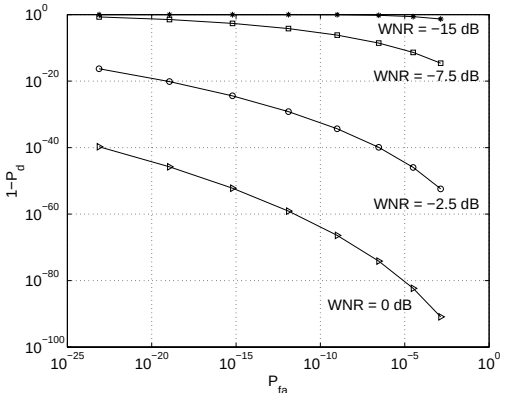
where the watermark-to-noise ratio $WNR \triangleq \frac{Var(W)}{Var(N)}$. In Figure 2.2(a), we plot the mean of the Z statistic computed using the derived ρ for different WNR and L.



(a)



(b)



(c)

Figure 2.2: Z statistics and ROC curves: (a) Mean of Z statistics for different WNR and L; (b) ROC curves for different L at WNR = 0 dB; (c) ROC curves for different WNR at $L = 700$.

Now knowing the distribution of the Z statistic under the presence and absence of Alice's fingerprint, we can compute the probabilities of detection P_d and false alarm P_{fa} for different decision thresholds. A threshold of 3 gives a false alarm probability on the order of 10^{-3}, while a threshold of 6 corresponds to the order

of 10^{-9}. The tradeoff between P_d and P_{fa} can be visualized in terms of Receiver Operating Characteristic (ROC) curves, as shown in Figure 2.2(b) for different L at a fixed WNR and Figure 2.2(c) for different WNR at a fixed L.

In reality, as the noise introduced by attacks does not necessarily follow a Gaussian distribution and/or the noise samples may be mutually correlated, the Z statistics with true traitors may be different from the unit-variance Gaussian distribution. The actual distributions of the Z statistics for the fingerprint presence/absence cases and the detection performance in terms of the detection probability and the false alarm probability will be presented in our experimental results.

2.2.4 Fidelity and Robustness Considerations

Fingerprint Construction: The collusion resistance requirement makes the fingerprinting problem more challenging than robustly embedding a meaningful ID label, as the simple encoding of IDs can be vulnerable to collusion (e.g., different users average their copies of the same content to remove the IDs). Designing a collusion-resistant code is one of the possible approaches and has been studied in [83]. Such a coded approach requires that each code symbol be reliably embedded, which consumes a non-trivial amount of markable features per embedded bit. The markable feature for our curve watermarking problem is the coordinates of control points. As the number of control points is limited and the changes have to be small, orthogonal modulation that uses (approximately) orthogonal signals to represent different users is more attractive than the coded modulation [89]. The general collusion resistance of orthogonal fingerprinting has been studied in [85], which shows the maximum number of colluders that the system can resist is a function of the watermark-to-noise ratio, the number of markable features, the

total number of users, as well as the false positive and negative requirements.

While the orthogonal design of fingerprints is conceptually simple and easy to analyze, the practical implementation often employs a pseudo-random number generator to produce a sequence of independent random numbers as a fingerprint and uses different seeds for different users [50, 83, 97]. In this way, the actual fingerprints would be statistically uncorrelated, but they can have a non-zero correlation. This correlation can accommodate a larger number of fingerprint vectors than the vector's dimension, but it also affects the detection performance to some degree. Since the correlation is very low between the independent fingerprints, the impact is small. This can be seen from our experimental results of the detection performance in Section 2.2.5 and Section 2.4.

Curvature-based Sampling: The overall distortion introduced by the embedding process on a curve consists of two parts: one is from the watermark signals added to the control point coordinates, and the other is from the B-spline modelling. To make the B-spline synthesized curve as close to the original curve as possible and thus keep the modelling error low, the knots connecting adjacent segments of B-splines should be wisely placed, and the sample points should be properly chosen to feed into the least-squares estimator for the control points. Uniform sampling can be used when there are no abrupt changes in a curve segment, while nonuniform sampling is desirable for curve segments that exhibit substantial variations in curvature.

Inspired by [11,45], we employ a curvature-based method to select sample points from raster curves. Formally, the curvature [21] of a point $\mathbf{p}(t) = (p_x(t), p_y(t))$ on a curve $\{\mathbf{p}(t)\}$ is defined as

$$k(t) \triangleq \frac{p'_x p''_y - p'_y p''_x}{({p'_x}^2 + {p'_y}^2)^{3/2}}, \tag{2.13}$$

where $p'_x = \frac{dp_x}{dt}$, $p''_x = \frac{d^2p_x}{dt^2}$, $p'_y = \frac{dp_y}{dt}$, and $p''_y = \frac{d^2p_y}{dt^2}$. In practical implementations, we approximate the curvature of each point on the curve by measuring the angular change in the tangent line at its location. Specifically, we perform a 1^{st}-order polynomial curve fitting on an l-pixel interval before and after the curve point $\mathbf{p}(t)$ to get two slopes, k_1 and k_2. The approximate curvature $\hat{k}(t)$ is computed by $\hat{k}(t) = |\arctan(k_1) - \arctan(k_2)|$. Based on $\hat{k}(t)$, we select more sample points from higher-curvature segments and fewer from lower-curvature segments.

After selecting $m+1$ sample points and put them together into $(\mathbf{p}_x, \mathbf{p}_y)$ as defined in Equation (2.3), we need to determine their indexing values $t = s_0, s_1, s_2, \ldots, s_m$, which will be used to evaluate their B-spline blending function values $B_{i,k}(t)$. In our tests, we employ the uniform non-periodic B-spline blending function of order $k = 3$, and the knot parameters $\{t_i\}$ are determined as $[t_0, t_1, ..., t_{n+3}] = [0, 0, 0, 1, 2, 3, ..., n-2, n-1, n-1, n-1]$. One way of the indexing-value assignment is known as the chord-length method [16], which increases t values of the sample points in proportion to the chord length

$$s_j = s_{j-1} + \|\mathbf{p}_j - \mathbf{p}_{j-1}\| \frac{n-1}{\sum_{i=1}^{m} \|\mathbf{p}_i - \mathbf{p}_{i-1}\|}, j = 1, 2, ...m, \tag{2.14}$$

where $s_0 = 0$ and $\|\mathbf{p}_j - \mathbf{p}_{j-1}\|$ denotes the chord length between points $\mathbf{p}_j$ and $\mathbf{p}_{j-1}$.

In the case of vector curves, we are generally given a set of discrete, non-uniformly spaced points for each curve. Since a vector curve can be rendered as a raster curve by interpolation, we can determine its sample points and their corresponding indexing values by rendering it to be a raster curve and then performing the curvature-based sampling and the chord-length method as described above. A simpler alternative is to directly use the given discrete points as sample points but assign their indexing values according to a curvature-based rule. Specifically, we

approximate the curvature of each sample point $(p_x(s_j), p_y(s_j))$ by

$$\hat{k}(s_j) = \left| \arctan \frac{p_y(s_{j+1}) - p_y(s_j)}{p_x(s_{j+1}) - p_x(s_j)} - \arctan \frac{p_y(s_j) - p_y(s_{j-1})}{p_x(s_j) - p_x(s_{j-1})} \right| \tag{2.15}$$

and then increase t values of the sample points in inverse proportion to their curvature. The higher the curvature, the smaller the increase in the t value. This is equivalent to having more control points for higher-curvature segments. With more "resources" in terms of B-spline control points assigned to segments with more details (i.e., higher-curvature segments), it allows for a better B-spline approximation.

Determining the Fingerprint Length and Strength: The number of control points is an important parameter for tuning. Depending on the shape of the curve, using too few control points could cause the details of the curve to be lost, while using too many control points may lead to over-fitting and bring artifacts even before data embedding. One simple method of determining the number of control points is to compute the approximate curvature of each sample point as in (2.15) and assign higher weights to points with higher curvature. We then determine the number of control points according to the total weights of all sample points on the curve. The number of control points not only affects the distortion introduced by the embedding, but also determines the fingerprint's robustness against noise and attacks. The more the control points there are, the longer the fingerprint sequence is, and in turn the more robust the fingerprint is against noise and attacks. Too many control points, however, may lead to over-fitting and incur visible distortions even before data embedding. In our tests, the number of control points is about 5-8% of the total number of curve pixels.

The scaling factor α also affects the invisibility and robustness of fingerprints. The larger the scaling factor is, the more robust the fingerprint is, but it results

in a larger distortion. For cartographic applications, industrial standards provide guidelines on the maximum allowable changes. Perturbation of 2 to 3 pixels is usually considered acceptable. We use random number sequences with a unit variance as fingerprints and set α to 0.5 in our tests. The difference between two curves can be quantified using such max-min metrics as the Hausdorff distance [9]. More specifically, let $d(a, b)$ be the distance between two points a and b, which are on two curves, A and B, respectively. We further define the distance from point a to curve B as $d(a, B) \triangleq \inf_{b \in B} d(a, b)$, and the distance from curve A to curve B as $d_B(A) \triangleq \sup_{a \in A} d(a, B)$. Thus, the Hausdorff distance between curve A and B is

$$h(A, B) \triangleq d_B(A) + d_A(B) \tag{2.16}$$

Nonblind Detection: The basic fingerprint detection presented earlier makes use of the original unmarked copy and is known as *nonblind detection*. While blind detection is preferred for a few major data hiding applications (such as ownership verification, authentication, and annotation), non-blind detection is considered as a reasonable assumption for many fingerprinting applications [19, 58, 83], which is also the focus of this and the following chapter. The rationale for nonblind detection is that the fingerprint verification is usually handled by the content owner or by an authorized central server, who can have access to the original host signal and use it in the detection to answer the primary question of whose fingerprint is in the suspicious document. The availability of the original unmarked copy in the detection gives a high equivalent watermark-to-noise ratio, thus allowing for high resistance against noise and attacks. Additionally, using the original unmarked copy as a reference copy, the detector can register a test copy that suffers from geometric distortions, which enables the resilience to various geometric

transformations as to be demonstrated later in this chapter.

2.2.5 Experimental Results of Fingerprinting Simple Curves

To demonstrate our basic embedding and detection algorithms, we first present the fingerprinting results on two simple curves, the "Swan" curve in Fig. 2.3(a) and the "W" curve in Fig. 2.4(a). These two curves were hand-drawn on a Tablet PC and stored as binary images of size 329×392 and 521×288, respectively. We use the contour following algorithm in [11] to traverse the curve and obtain a set of ordered curve points. When fingerprinting these two curves, we perform uniform sampling on the curve points and determine the indexing values of the sample points using the chord-length method. We highlight the control points of the "Swan" curve in Fig. 2.3(c). The fingerprinted curves are shown in Fig. 2.3(b) and Fig. 2.4(b), where we have marked 101 control points for each curve. With the marked control points, we construct the fingerprinted curve with the same number of points as the original curve by evaluating the B-spline synthesis formula (2.1) at indexing values uniformly sampled between $t = 0$ and $t = n - 1$. As for the fidelity of the fingerprinted curves, the Hausdorff distance between the original and marked curves is 5.0 for the "Swan" curve, and 3.4 for the "W" curve. The differences are hardly visible to human eyes.

In the detection, we take a fingerprinted curve constructed above as a test curve and apply uniform sampling to it to obtain an approximation of the set of sample points $(\tilde{\mathbf{p}}_x, \tilde{\mathbf{p}}_y) = (\mathbf{B}(\mathbf{c}_x + \alpha\mathbf{w}_x), \mathbf{B}(\mathbf{c}_y + \alpha\mathbf{w}_y))$ assumed in Section 2.2.2. We then estimate the test control points and perform the correlation-based detection. The detection results on the fingerprinted "W" curve are shown in Fig. 2.4(c), which illustrates a high Z value for the correct positive detection with the 1000^{th}

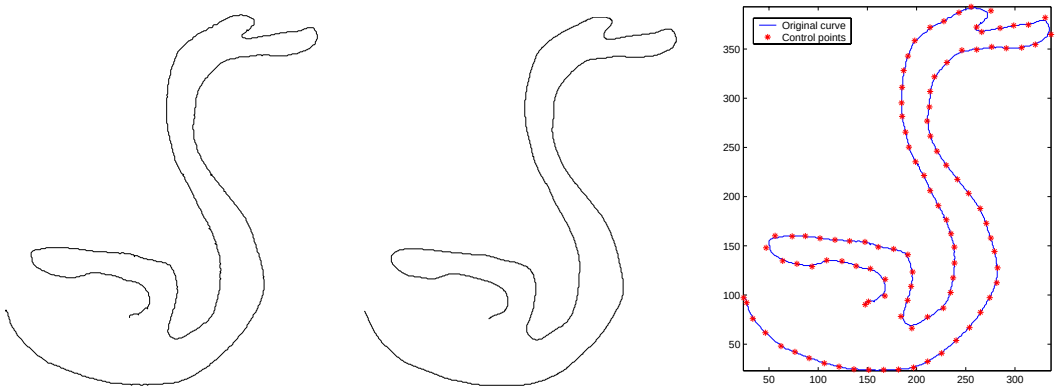
sequence corresponding to the true user, and very small Z statistics for the correct negative detection with other sequences of the innocent users. To quantify the detection performance, we generate 1000 different sets of fingerprint sequences, and each set consists of 1000 independent fingerprint sequences that are approximately orthogonal to each other. For each set, we embed each of the 1000 fingerprint sequences into the original curve to form a fingerprinted curve, and then estimate a fingerprint sequence from the fingerprinted curve and compute the Z values with the 1000 fingerprint sequences. In this way, we collect a total of $1000 \times 1000 = 1 \times 10^6$ values for the fingerprint presence case and $1000 \times 1000 \times 999 \approx 1 \times 10^9$ values for the fingerprint absence case. Using these data, we plot in Fig. 2.5 the histogram of the Z values for both fingerprint presence and absence cases. For each of these two sets of data, we calculate its mean and variance, and then plot the Gaussian distribution with the calculated mean and variance. As shown by the dashed curves in Fig. 2.5, the two Gaussian approximations, $\mathcal{N}(11.22, 0.87)$ and $\mathcal{N}(-0.0009, 1.00)$, fit the observed Z values very well. We can see that we indeed get an approximate Gaussian distribution with a large positive mean under the presence of a fingerprint and a zero mean under the absence.

We further examine the measured and the Gaussian approximated probabilities of detection and false alarm for different thresholds on the Z statistic. We see from the results in Table 2.1 that the Gaussian approximation works well in regions close to the mean, while the measured values slightly deviate from the Gaussian approximation in regions farther from the mean. At the same time, we note that in the tail regions where the Gaussian approximation becomes loose, the approximated order of magnitude matches very well with the measured value from our experiments. The miss probability or false alarm probability in the tail regions

Table 2.1: Measured detection performance and its Gaussian approximation for the "W" curve.

Threshold on Z		*3*	*4.5*	*6*	*7.5*	*9*
$1\text{-}P_d$	Measured	0	0	0	4.1×10^{-5}	8.7×10^{-3}
	Gaussian approx.	6.6×10^{-19}	3.1×10^{-13}	1.1×10^{-8}	3.4×10^{-5}	8.7×10^{-3}
P_{fa}	Measured	1.4×10^{-3}	4.1×10^{-6}	2.0×10^{-9}	0	0
	Gaussian approx.	1.3×10^{-3}	3.4×10^{-6}	0.97×10^{-9}	3.1×10^{-14}	1.1×10^{-19}

is already very small (below 10^{-5}). Therefore, for many practical applications, either the probability may be deemed as zero, or an approximation on the order of magnitude would be sufficient. The table also shows that a threshold of 6 on the detection statistics gives a false alarm probability of 10^{-9}, which is sufficiently low for most applications. Thus, we choose 6 as the detection threshold in our tests. The detection result for the "Swan" curve is similar and will not be repeated here.



(a) (b) (c)

Figure 2.3: Fingerprinting a hand-drawn "*Swan*" curve: (a) original curve; (b) fingerprinted curve; (c) control points overlaid on the original curve.

Furthermore, we examine the survivability of the fingerprints by using our basic scheme, which employs the coordinates of the B-spline control points as the

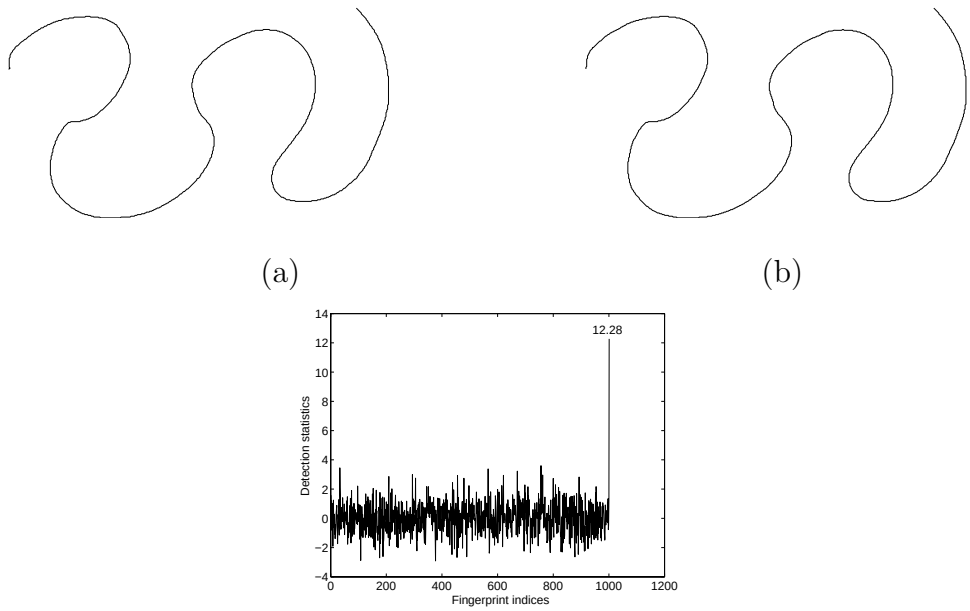


Figure 2.4: Fingerprinting a hand-drawn "*W*" curve: (a) original curve; (b) fingerprinted curve; (c) detection statistics.

embedding domain, and uses approximately orthogonal spread spectrum signals as fingerprints. We perform a printing-and-scanning test with manual registration between the scanned fingerprinted curve and the original unmarked curve. We print out the fingerprinted "W" curve using a HP laser printer, and scan it back as a 527×288 binary image as shown in Fig. 2.6(a). In addition to manual registration, a thinning operation is performed to extract a one-pixel wide skeleton from the scanned curve that is usually several pixels' wide after high-resolution scanning. As we can see from the detection results in Fig. 2.6(b), despite the curve being simple and the number of control points being relatively small, the fingerprint survives the printing-and-scanning process and gives a detection statistic higher than the detection threshold. The issue of automating the registration process will be addressed in the next section.

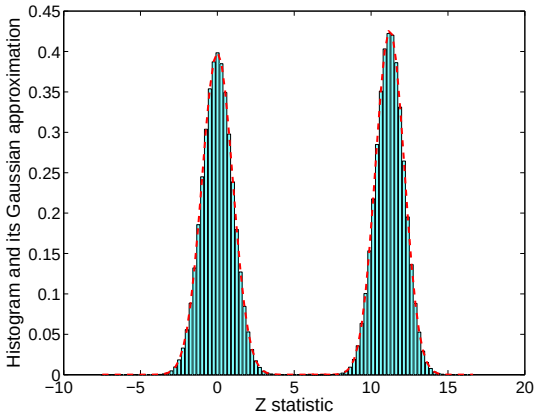
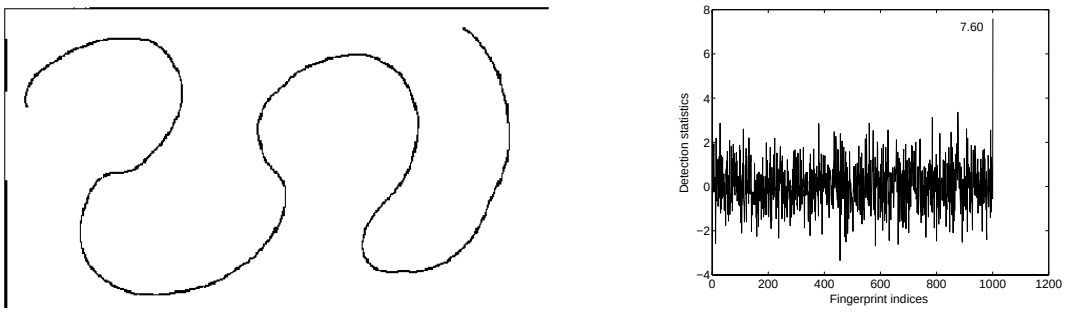


Figure 2.5: Histogram and Gaussian approximation of the Z statistics for the "W" curve.

2.3 Iterative Alignment-Minimization Algorithm for Robust Fingerprint Detection

The set of test sample points $(\tilde{\mathbf{p}}_x, \tilde{\mathbf{p}}_y)$ assumed in Section 2.2.2 is not always available to a detector, especially when a test curve undergoes vector-raster conversion, geometric transformations (such as rotation, translation, and scaling), and/or is scanned from a printed hard copy. A pre-processing step preceding the basic fingerprint detection module is needed to align the test curve with the original one. While manual registration between the test curve and the original unmarked curve shown in Section 2.2.5 is a possible way to overcome simple geometric distortions, automated registration is more desirable to improve the accuracy and efficiency of this indispensable pre-processing step. Note that the test curve should be registered with the original unmarked curve, and any "clean/undistorted" fingerprinted copies known to the detector should not be used as a reference for registering the test curve. This is not only because which fingerprints are present in the test curve still remains to be determined, but also because using a fingerprinted copy as a



(a) (b)

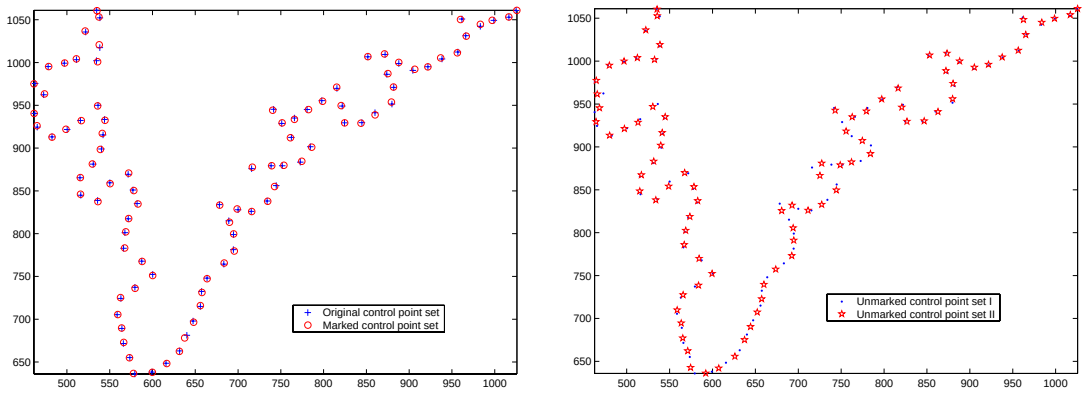
Figure 2.6: Printing-and-scanning test for the "W" curve: (a) fingerprinted curve after printing-and-scanning; (b) detection statistics.

reference for registration may increase the false alarm probability in determining the presence or absence of the corresponding fingerprint.

With the affine invariance property, B-splines have been used in a few existing curve alignment works. In the moment-based approach of [45], two affine-related curves are fitted by two separate B-splines, and the transform parameters are estimated by using weighted B-spline curve moments. This method requires taking integration as well as the second-order curve derivatives to obtain the moments. In a recent method employing a *super-curve* [93], two affine-related curves are superimposed together in the same frame, and then this combined super-curve is fitted by a single B-spline. Through minimizing the B-spline fitting error, both transform parameters and control points of the fitting B-spline can be estimated simultaneously. Since neither integration nor differentiation is needed, this method is robust to noise and will serve as a building block in our work.

Another problem related to the test sample points assumed earlier is the inherent non-uniqueness of B-spline control points, which refers to the fact that a curve can be effectively approximated by different sets of B-spline control points. We have seen from Section 2.2.1 that B-spline control points are estimated from

a set of sample points from the curve. With a different set of sample points or a different indexing-value assignment, we may induce a quite different set of control points that can still well describe the same curve. It is possible for the difference between two sets of unmarked control points to be much larger than the embedded fingerprint sequence, as demonstrated by the example in Fig. 2.7. Therefore, if we cannot find from a test curve a set of control points corresponding to the one used in the embedding, we may not be able to detect the fingerprint sequence. Considering the one-to-one relationship between sample points (including their indexing values $\{s_j\}$) and control points, we try to find the set of sample points from a test curve that corresponds to the set of sample points used in the embedding. We shall refer to this problem as the *point correspondence problem*. As we shall see, the non-uniqueness issue of B-spline control points can be addressed through finding the point correspondence.



(a) (b)

Figure 2.7: Non-uniqueness of B-spline control points: (a) a set of control points for the original unmarked curve and its fingerprinted version; (b) two different sets of control points for modelling the same unmarked curve.

In the following subsections, we first formulate the curve registration and point

correspondence problem in the context of fingerprint detection. We then take the curve alignment method introduced in [93] as a building block and propose an Iterative Alignment-Minimization (IAM) algorithm that can perform curve registration and solve the point correspondence problem simultaneously. Finally, we present a detection example for a single curve using the IAM algorithm, and discuss the robustness issues.

2.3.1 Problem Formulation

We use "View-I" to refer to the geometric setup of the original unmarked curve and "View-II" to refer to the setup of the test curve. Thus, we can register the two curves by transforming the test curve from "View-II" to "View-I", or transforming the original curve from "View-I" to "View-II". We focus on registration under affine transformations, which can represent combinations of scaling, rotation, translation, reflection, and shearing. These are common geometric transformations and can effectively model common scenarios in the printing-and-scanning process.

Under affine transformations, each point (x, y) on one curve is transformed to a corresponding point $(\tilde{x}, \tilde{y})$ on another curve via

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix}, \tag{2.17}$$

where $\{a_{ij}\}$ are parameters representing the collective effect of scaling, rotation, translation, reflection and shearing. The transform parameters can also be represented in a homogeneous coordinate by two column vectors $\mathbf{a}_x$ and $\mathbf{a}_y$, or by a single matrix $\mathbf{A}$:

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{a}_x^T \\ \mathbf{a}_y^T \\ 0\ 0\ 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \quad (2.18)$$

Similarly, the inverse transform can be represented by

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{bmatrix} \triangleq \begin{bmatrix} \mathbf{g}_x^T \\ \mathbf{g}_y^T \\ 0\ 0\ 1 \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{bmatrix}. \quad (2.19)$$

The original curve available to the detector in fingerprinting applications can be a raster curve or a vector curve. The detector also knows the original set of sample points $(\mathbf{p}_x, \mathbf{p}_y) \approx (\mathbf{B}\mathbf{c}_x, \mathbf{B}\mathbf{c}_y)$ that is used for estimating the set of control points upon which spread spectrum embedding is applied. The test curve can be a vector curve with sampled curve points $(\tilde{\mathbf{v}}_x, \tilde{\mathbf{v}}_y)$ or a raster curve with pixel coordinates $(\tilde{\mathbf{r}}_x, \tilde{\mathbf{r}}_y)$. A relatively simple case is that the set of discrete points in a test vector curve corresponds to the set of sample points used in the embedding, except with possible affine transformations and noise addition. In this case, the test vector points $(\tilde{\mathbf{v}}_x, \tilde{\mathbf{v}}_y)$ and the original set of control points $(\mathbf{c}_x, \mathbf{c}_y)$ are related by

$$\begin{cases} \tilde{\mathbf{v}}_x = \begin{bmatrix} \mathbf{B}(\mathbf{c}_x + \alpha\mathbf{w}_x) & \mathbf{B}(\mathbf{c}_y + \alpha\mathbf{w}_y) & \mathbf{1} \end{bmatrix} \mathbf{a}_x + \mathbf{n}_x \\ \tilde{\mathbf{v}}_y = \begin{bmatrix} \mathbf{B}(\mathbf{c}_x + \alpha\mathbf{w}_x) & \mathbf{B}(\mathbf{c}_y + \alpha\mathbf{w}_y) & \mathbf{1} \end{bmatrix} \mathbf{a}_y + \mathbf{n}_y. \end{cases} \quad (2.20)$$

where $(\mathbf{n}_x, \mathbf{n}_y)$ represents additional noise applied to the transformed fingerprinted vector points, and $\mathbf{1}$ is a column vector with all $1's$. With the point correspondence available, the only issue is curve alignment, and it can be solved by directly applying the curve alignment method in [93]. However, in addition to possible affine transformations between the original and the test curves, the correct point correspondence information may not always be available. This is especially the case

after a fingerprinted curve undergoes vector-raster conversions and/or printing-and-scanning. Under these situations, not only transform parameters for the curve alignment but also the point correspondence must be estimated, in order to locate the fingerprinted control points successfully. We consider that both the original and the test curves are represented in raster format since a vector curve can be rendered as a raster curve by interpolation, and that the sample points used in fingerprinting the original curve are known to the detector. The problem can be formulated as follows:

Given an original raster curve with a set of sample points $(\mathbf{p}_x, \mathbf{p}_y)$ *and a test raster curve* $(\tilde{\mathbf{r}}_x, \tilde{\mathbf{r}}_y)$*, we register the test curve with the original curve and extract the control points of the test curve. Both transform parameters* $(\mathbf{a}_x, \mathbf{a}_y)$ *(or equivalently* $(\mathbf{g}_x, \mathbf{g}_y)$*) and a set of sample points* $(\tilde{\mathbf{p}}_x, \tilde{\mathbf{p}}_y)$ *corresponding to the one used in the fingerprint embedding must be found from the test curve.*

2.3.2 Iterative Alignment-Minimization (IAM) Algorithm

To align the test curves with the original curves and in the mean time identify the point correspondence of the sample points, we develop an Iterative Alignment-Minimization (IAM) algorithm. As shown in Fig. 2.8, the IAM algorithm consists of three main steps and the latter two steps will be executed iteratively. We first obtain an initial estimation of the test sample points. With the estimated point correspondence, we then perform super-curve alignment to estimate both the transform parameters and the control points of the test curve. With the estimated transform parameters, we refine the estimation of point correspondence through a nearest-neighbor rule. A detailed block diagram of the proposed IAM-based fingerprint detection is shown in Fig. 2.9.

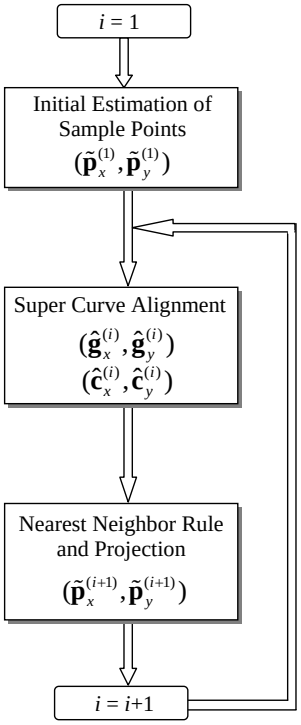


Figure 2.8: Basic flow and main modules of the proposed Iterative Alignment-Minimization (IAM) algorithm.

Step-1 Initial Estimation of Sample Points on the Test Curve We initialize the sample points $(\tilde{\mathbf{p}}_x^{(1)}, \tilde{\mathbf{p}}_y^{(1)})$ on the test curve using the following simple estimator. Let M and $\tilde{M}$ be the number of points on the original and the test raster curves, respectively. From the known indices $\mathbf{J} = [j_0, j_1, j_2, \ldots, j_m]$ of the original curve's $m+1$ sample points, where $j_0 < j_1 < j_2 < \cdots < j_m$ are integers ranging from 0 to $M-1$, we estimate the indices of the test curve's $m+1$ sample points by $\tilde{\mathbf{J}} = round\left(\frac{\tilde{M}-1}{M-1} \cdot \mathbf{J}\right)$. Using this estimated index vector $\tilde{\mathbf{J}}$, we can identify their corresponding sample points from the test curve and take them as the initial estimate.

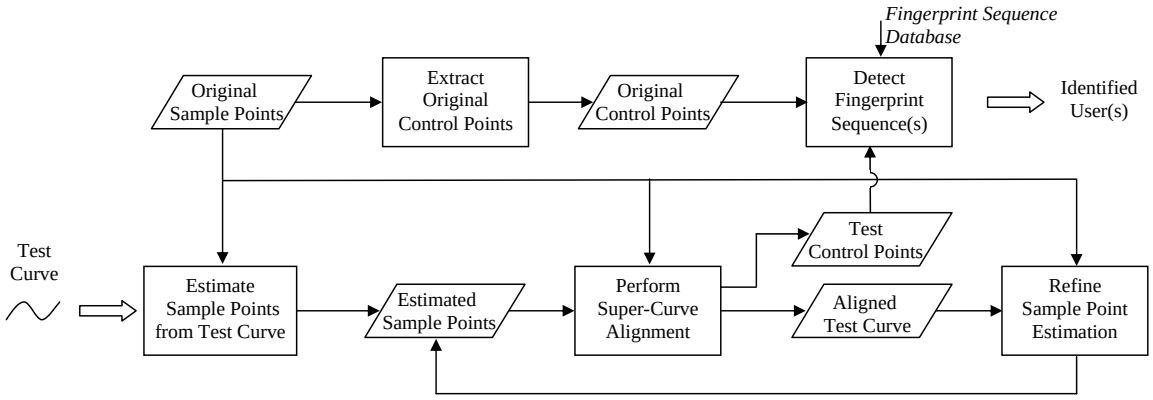


Figure 2.9: Block diagram of curve registration and fingerprint detection using the proposed IAM algorithm.

Step-2 Curve Alignment with the Estimated Sample Points Given the estimated point correspondence with sample points $(\tilde{\mathbf{p}}_x^{(i)}, \tilde{\mathbf{p}}_y^{(i)})$ for the test curve in the i^{th} iteration, we apply the curve alignment method in [93] to estimate the transform parameters and the control points of the test curve. More specifically, let the transform parameters from View-I (the original curve) to View-II (the test curve) be $(\mathbf{a}_x^{(i)}, \mathbf{a}_y^{(i)})$. The sample points on the test curve can be transformed back to View-I by $(\mathbf{g}_x^{(i)}, \mathbf{g}_y^{(i)})$. We then fit these transformed test sample points as well as the original sample points with a single B-spline curve (referred to as a super-curve in [93]), and we search for both the transform parameters $(\hat{\mathbf{g}}_x^{(i)}, \hat{\mathbf{g}}_y^{(i)})$ and the B-spline control points $(\hat{\mathbf{c}}_x^{(i)}, \hat{\mathbf{c}}_y^{(i)})$ to minimize the fitting error

$$f(\hat{\mathbf{c}}_x^{(i)}, \hat{\mathbf{c}}_y^{(i)}, \hat{\mathbf{g}}_x^{(i)}, \hat{\mathbf{g}}_y^{(i)}) = \left\| \begin{bmatrix} \mathbf{B} \\ \mathbf{B} \end{bmatrix} \hat{\mathbf{c}}_x^{(i)} - \begin{bmatrix} \mathbf{p}_x \\ \tilde{\mathbf{P}}^{(i)} \hat{\mathbf{g}}_x^{(i)} \end{bmatrix} \right\|^2 + \left\| \begin{bmatrix} \mathbf{B} \\ \mathbf{B} \end{bmatrix} \hat{\mathbf{c}}_y^{(i)} - \begin{bmatrix} \mathbf{p}_y \\ \tilde{\mathbf{P}}^{(i)} \hat{\mathbf{g}}_y^{(i)} \end{bmatrix} \right\|^2 \quad (2.21)$$

where $\tilde{\mathbf{P}}^{(i)} \triangleq [\ \tilde{\mathbf{p}}_x^{(i)} \quad \tilde{\mathbf{p}}_y^{(i)} \quad \mathbf{1}\ ]$ and $\mathbf{1}$ is a column vector with all $1's$. The partial derivatives of the fitting error function with respect to $\hat{\mathbf{g}}_x^{(i)}, \hat{\mathbf{g}}_y^{(i)}, \hat{\mathbf{c}}_x^{(i)}$, and $\hat{\mathbf{c}}_y^{(i)}$ being zero is the necessary condition for the solution to this optimization problem. Thus, we obtain an estimate of the transform parameters and the B-spline control points

as

$$\begin{cases} \hat{\mathbf{g}}_x^{(i)} = \mathbf{C}^{(i)}\mathbf{D}^{(i)}\mathbf{p}_x, \quad \hat{\mathbf{g}}_y^{(i)} = \mathbf{C}^{(i)}\mathbf{D}^{(i)}\mathbf{p}_y, \\ \hat{\mathbf{c}}_x^{(i)} = \mathbf{D}^{(i)}\mathbf{p}_x, \quad \hat{\mathbf{c}}_y^{(i)} = \mathbf{D}^{(i)}\mathbf{p}_y \end{cases},$$

$$\text{where} \quad \begin{cases} \mathbf{C}^{(i)} \triangleq \left(\tilde{\mathbf{P}}^{(i)T}\tilde{\mathbf{P}}^{(i)}\right)^{\dagger}\tilde{\mathbf{P}}^{(i)T}\mathbf{B} \\ \mathbf{D}^{(i)} \triangleq \left(2\mathbf{B}^T\mathbf{B} - \mathbf{B}^T\tilde{\mathbf{P}}^{(i)}\mathbf{C}^{(i)}\right)^{\dagger}\mathbf{B}^T \end{cases}. \tag{2.22}$$

The estimated control points $(\hat{\mathbf{c}}_x^{(i)}, \hat{\mathbf{c}}_y^{(i)})$ can then be used to estimate the fingerprint sequence and further compute the detection statistic $Z^{(i)}$, as described in Section 2.2.

Step-3 Refinement of Sample Point Estimation on the Test Curve Given the estimated transform parameters $(\hat{\mathbf{g}}_x^{(i)}, \hat{\mathbf{g}}_y^{(i)})$, we align the test raster curve $(\tilde{\mathbf{r}}_x, \tilde{\mathbf{r}}_y)$ with the original curve by transforming it to View-I

$$\begin{cases} \tilde{\mathbf{r}}_{x,I}^{(i)} = [\tilde{\mathbf{r}}_x \quad \tilde{\mathbf{r}}_y \quad \mathbf{1}]\,\hat{\mathbf{g}}_x^{(i)} \\ \tilde{\mathbf{r}}_{y,I}^{(i)} = [\tilde{\mathbf{r}}_x \quad \tilde{\mathbf{r}}_y \quad \mathbf{1}]\,\hat{\mathbf{g}}_y^{(i)} \end{cases}. \tag{2.23}$$

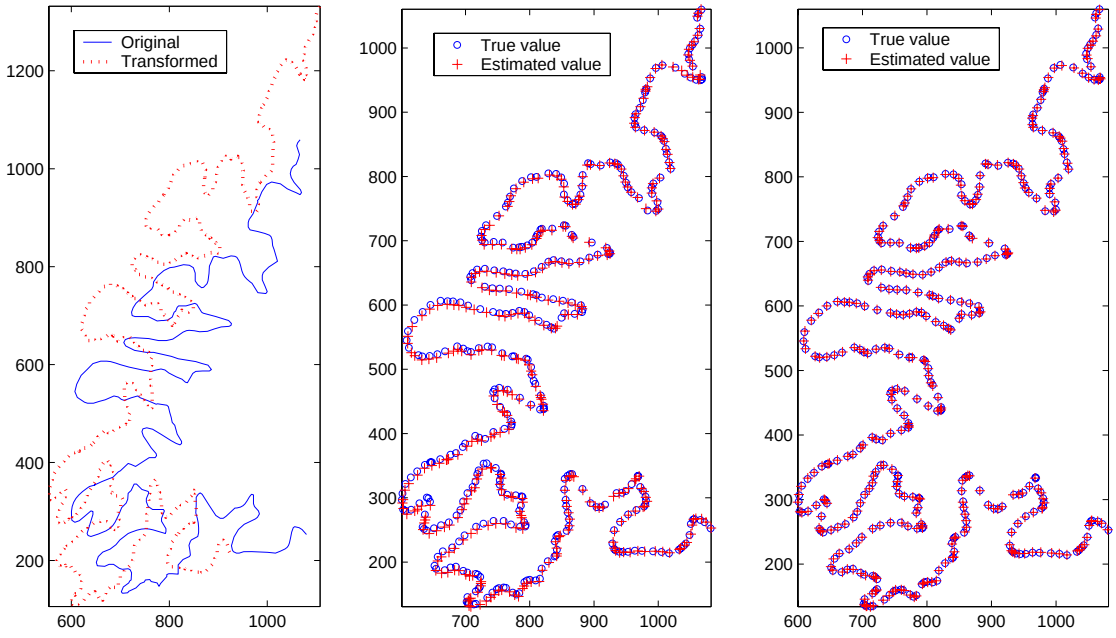
As the fingerprinted sample points $(\mathbf{B}(\mathbf{c}_x + \alpha\mathbf{w}_x), \mathbf{B}(\mathbf{c}_y + \alpha\mathbf{w}_y))$ are located at the neighborhood of their corresponding unmarked version $(\mathbf{B}\mathbf{c}_x, \mathbf{B}\mathbf{c}_y)$, we apply a *nearest-neighbor* rule to refine the estimation of the test curve's sample points. More specifically, for each point of $(\mathbf{B}\mathbf{c}_x, \mathbf{B}\mathbf{c}_y)$, we find its closest point from the aligned test raster curve $(\tilde{\mathbf{r}}_{x,I}^{(i)}, \tilde{\mathbf{r}}_{y,I}^{(i)})$, and then denote the collection of these closest points as $(\tilde{\mathbf{p}}_{x,I}^{(i+1)}, \tilde{\mathbf{p}}_{y,I}^{(i+1)})$. These nearest neighbors form a refined estimate of the test sample points in View-I and are then transformed with parameters $(\hat{\mathbf{a}}_x^{(i)}, \hat{\mathbf{a}}_y^{(i)})$ back to View-II as a new estimate of the test sample points

$$\begin{cases} \tilde{\mathbf{p}}_x^{(i+1)} = \left[\tilde{\mathbf{p}}_{x,I}^{(i+1)} \quad \tilde{\mathbf{p}}_{y,I}^{(i+1)} \quad \mathbf{1}\right]\hat{\mathbf{a}}_x^{(i)} \\ \tilde{\mathbf{p}}_y^{(i+1)} = \left[\tilde{\mathbf{p}}_{x,I}^{(i+1)} \quad \tilde{\mathbf{p}}_{y,I}^{(i+1)} \quad \mathbf{1}\right]\hat{\mathbf{a}}_y^{(i)}. \end{cases} \tag{2.24}$$

After this update, we increase i and go back to Step-2. The iteration will continue until convergence or for an empirically determined number of times. A total of 15 rounds of iterations are used in our experiments.

2.3.3 Detection Example and Discussions

We present a detection example employing the proposed IAM algorithm on a curve taken from a topographic map. Shown in Fig. 2.10(a) are the original curve and a fingerprinted curve having undergone vector-raster conversion and some geometric transformations. The original curve consists of 367 vector points, which are used as sample points to estimate a set of 200 control points for data embedding. Then, a fingerprinted curve with 367 vector points is generated, rendered to be a raster curve, and affinely transformed. After the vector-raster conversion, the point correspondence is no longer directly available from the raster curve representation. We apply the IAM algorithm to align the test curve with the original one and to estimate the correspondence between sample points. The estimated sample points for the test curve after one iteration and 15 iterations are shown in Fig. 2.10(b) and Fig. 2.10(c), respectively. We can see that initially the estimated values deviate from the true values by a non-trivial amount, while after 15 iterations the estimated values converge to the true values. We plot the six estimated transform parameters for each iteration in Fig. 2.11(a), which shows an accurate registration by the proposed IAM algorithm after half of a dozen iterations. Upon convergence, we use the estimated control points $(\hat{\mathbf{c}}_x^{(i)}, \hat{\mathbf{c}}_y^{(i)})$ to perform detection with the fingerprint involved. The high detection statistic value shown in Fig. 2.11(b) suggests the positive identification of the correct fingerprint by using the proposed IAM algorithm.

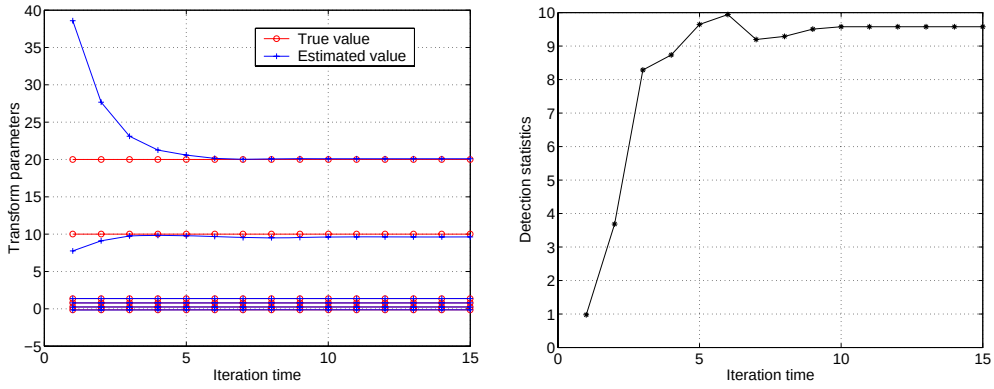


(a) (b) (c)

Figure 2.10: Register a curve using the proposed IAM algorithm: (a) the original curve and a fingerprinted curve undergone vector-raster conversion and affine transformations; (b) estimated sample points after one iteration; (c) estimated sample points after 15 iterations.

The computation time for this experiment is as follows. The IAM algorithm is implemented in Matlab 6.5 and tested on a Pentium-4 2.0GHz PC with 512M RAM. Each iteration of the algorithm takes about 0.5 of a second, and the total 15 iterations plus the initialization take 7.61 seconds. Together with the 0.56 of a second required for computing Z statistics with 1000 fingerprint sequences, the total duration of the detection process is 8.17 seconds.

The above example shows that through the IAM algorithm, we can register the test curve with the original unmarked curve and extract the fingerprinted control points with high accuracy. With good estimation of affine transform parameters, our data embedding method for curves is resilient to combinations of scaling, ro-



(a) (b)

Figure 2.11: Convergence results on estimated transform parameters and detection statistics for the example curve in Fig. 2.10 using the proposed IAM algorithm: (a) estimated transform parameters for each iteration; (b) fingerprint detection statistic for each iteration.

tation, translation, and shearing. The explicit estimation of point correspondence also provides resilience to the vector-raster and vector-raster-vector conversions. In the vector-raster conversion case, a fingerprinted curve stored in vector format is rendered as a raster curve, and thus the point correspondence is no longer directly available from the raster curve representation. In the vector-raster-vector conversion case, the intermediate raster curve is converted to a vector curve with a new set of vector points that are likely to be different from the initial vector points prior to the conversion, even though there is little visual difference between these two vector curves. Again the point correspondence is likely to get corrupted by this conversion, and accurate estimation of point correspondence is a necessary step for the successful detection of the fingerprint. With robustness resulting from the spread spectrum embedding in B-spline control points and the IAM algorithm, our curve fingerprinting approach can resist a number of challenging attacks and

distortions. For example, the distortion from printing-and-scanning involves both vector-raster rendering and a certain amount of rotation, scaling, and translation; a fingerprinted curve in vector format may be rendered as a raster image and then affinely transformed before reaching the detector; in the collusion scenario, colluders may construct a colluded copy, print it out, and then distribute it out of the allowed domain. In the next section, we use our curve-based data hiding approach to fingerprint topographic maps and demonstrate the robustness of our approach against various attacks and distortions.

2.4 Experimental Results on Map Fingerprinting

We now present experimental results of the proposed curve fingerprinting algorithm in the context of tracing and tracking topographic maps. A topographic map provides a two-dimensional representation of the earth's three-dimensional surface. Vertical elevation is shown with contour lines (also known as level lines) to represent the earth's surfaces that are of equal altitude. Contour lines in topographic maps often exhibit a considerable amount of variations and irregularities, prompting the need for non-uniform sampling of curve points in the parametric modelling of the contours. We shall first examine the fidelity of the fingerprinted map, and then evaluate the robustness of the fingerprints against collusion, cropping, geometric transformations, format conversions, point deletion, curve smoothing, printing-and-scanning, and some combinations of these distortions.

Fingerprinted Topographic Maps A 1100×1100 topographic vector map obtained from http://www. ablesw.com is used in our experiment. Starting with

the original map shown in Fig. 2.12(a), we mark nine curves that are sufficiently long. For each of these nine curves, a set of non-uniformly spaced vector points is given by the original dataset. We directly use these points as sample points and determine their indexing values according to the curvature-based rule presented in Section 2.2.4. A total of 1331 control points are used to carry the fingerprint. In Fig. 2.12(b), we overlay the nine original curves and the corresponding marked curves using solid lines and dotted lines, respectively. To help illustrate the fidelity of our method, we enlarge a portion of the overlaid image in Fig. 2.12(c). We can see that the fingerprinted map preserves the geospatial information in the original map with high precision. The perturbation can be adapted to be compliant with cartographic industry standards and/or the need of specific applications.

Resilience to Collusion To demonstrate the resistance of the proposed method against collusion, we present in Fig. 2.13 the detection statistics under two different types of collusion attacks. Fig. 2.13(a) shows the collusion results under a random interleaving attack, where the control points for each curve are equiprobably taken from two differently fingerprinted maps. The collusion attack for Fig.2.13(b) and (c) is known as averaging, where the coordinates of the corresponding control points from two and five differently fingerprinted maps are averaged, respectively. We assume the correct point correspondence is available in this test, and the cases with unknown point correspondence will be addressed in the later subsections. As we can see from the detection statistics, the embedded fingerprints from all contributing users survive the collusion attacks and are identified with high confidence. For both 2-user random interleaving collusion and 5-user averaging collusion, we use 20 different sets of fingerprint sequences to evaluate the detection performance, using a similar experiment setup to the one discussed in Section 2.2.5. The his-

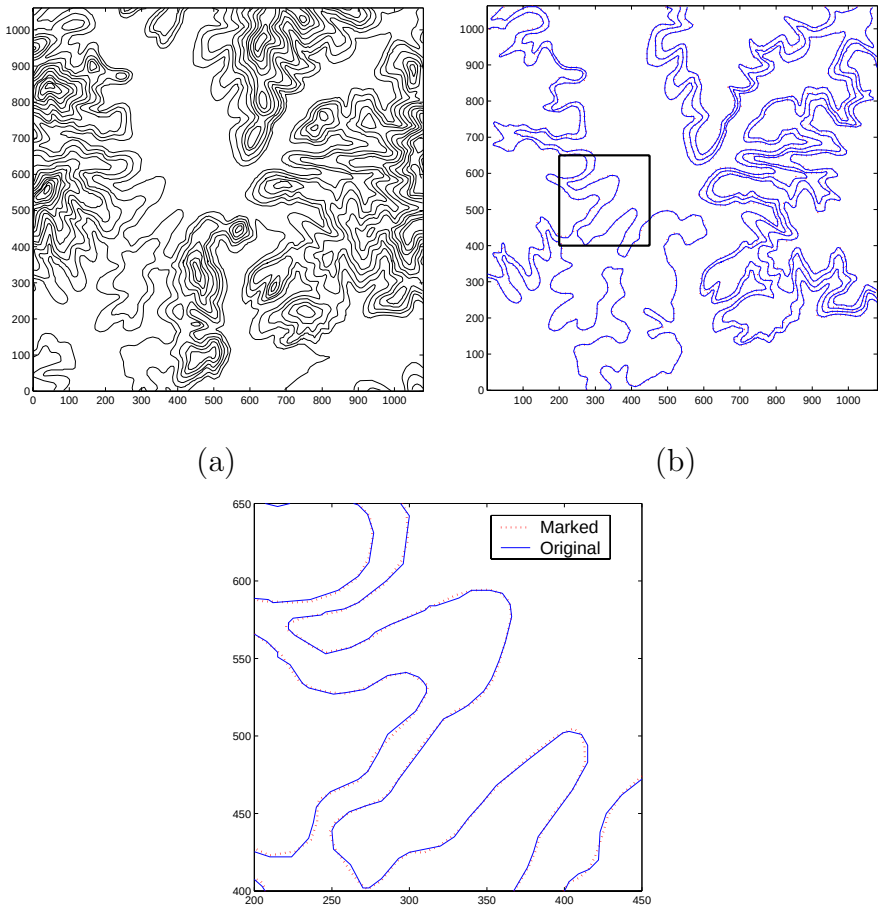


Figure 2.12: Fingerprinting topographic maps: (a) original map; (b) original and fingerprinted curves overlaid with each other; (c) a zoomed-in view of (b).

tograms and the Gaussian approximations in Fig. 2.14 show a very good separation between the Z values for the presence and absence of true fingerprints. Compared with the 2-user averaging collusion, the 2-user random interleaving leads to more substantial coordinate changes in control points when a detector performs B-spline parametrization, and such coordinate changes may follow a distribution different from i.i.d. Gaussian. This is reflected by a reduced mean and a non-unit variance for the fingerprint presence case.

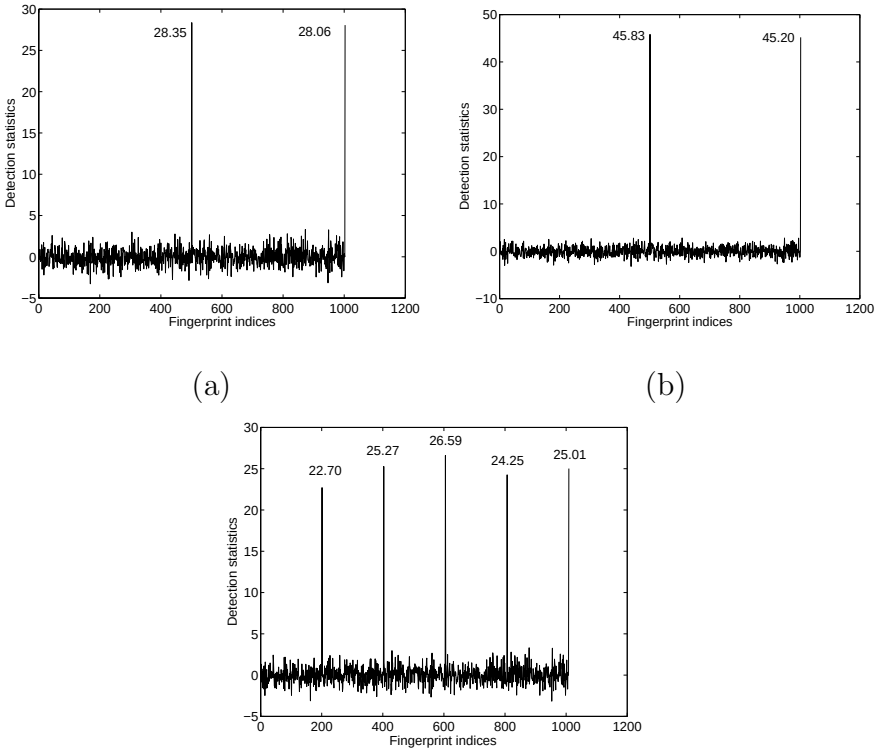
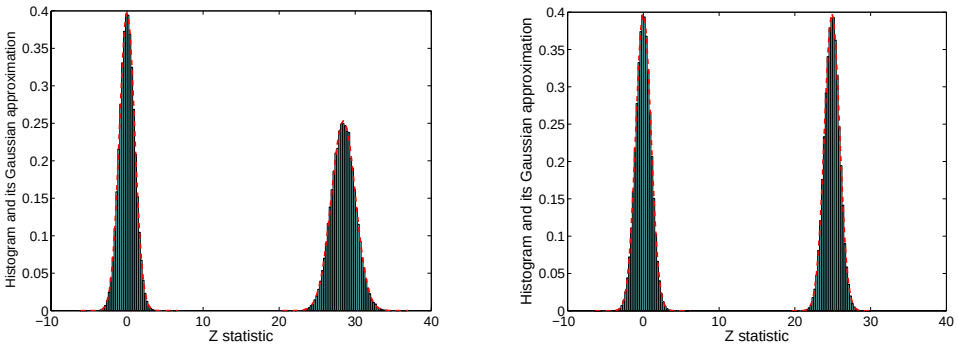


Figure 2.13: Collusion test on fingerprinted vector maps: (a) 2-user random interleaving collusion; (b) 2-user averaging collusion; (c) 5-user averaging collusion.

Resilience to Cropping As shown in Fig. 2.15 (a), we crop an area of a fingerprinted vector map and use it as the test map. Among the nine curves used

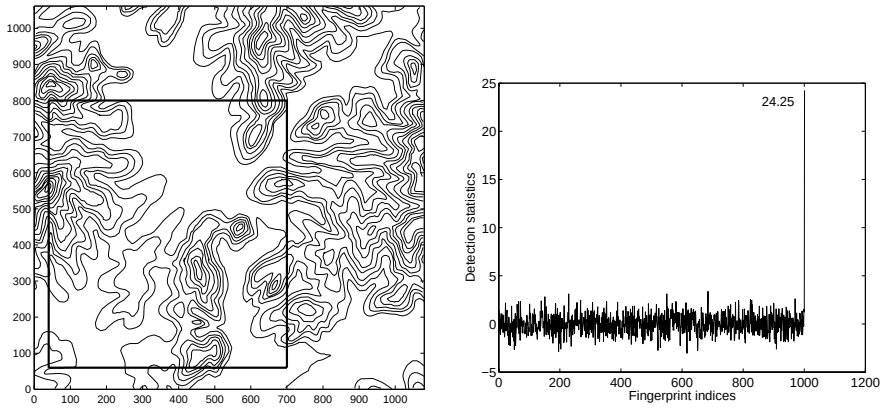


(a) (b)

Figure 2.14: Histogram and Gaussian approximation of the Z detection statistics for collusion attacks: (a) 2-user random interleaving collusion with Gaussian approximations $\mathcal{N}(0, 1.00)$ and $\mathcal{N}(28.45, 2.48)$; (b) 5-user averaging collusion with Gaussian approximations $\mathcal{N}(0, 1.00)$ and $\mathcal{N}(24.90, 1.00)$.

for carrying the fingerprint, only two curves are retained with sufficiently large size. Using the original map as a reference, we perform detection on these two retained segments and obtain the detection result shown in Fig. 2.15(b). As we can see, the detection statistic with the correct fingerprint is still high enough so that its corresponding user can be identified with high confidence.

Resilience to Affine Transformations on Vector Maps To demonstrate the resilience of our approach to a substantial amount of affine transformations, we take a fingerprinted vector map and apply a combination of rotation, scaling, and translation. More specifically, we rotate it by $-30°$, then scale it by 120% or 80% in the X and Y directions, respectively, followed by 100- and 200-pixel translation in the X and Y directions, respectively. The resulting vector map is rendered in Fig. 2.16(a). In this test, we assume that correct point correspondence is available. Thus, the super-curve alignment method can be directly applied to register the

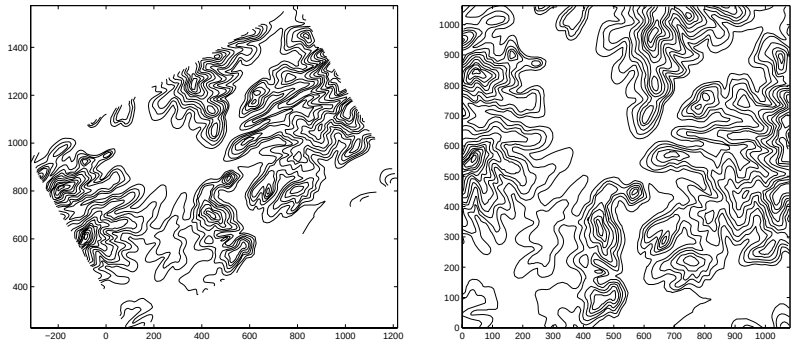


(a) (b)

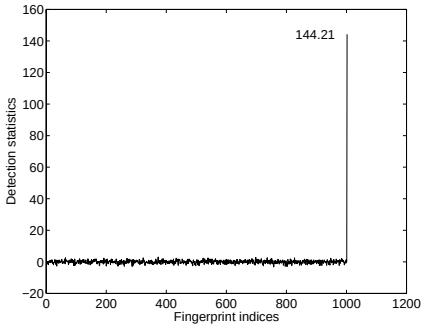
Figure 2.15: Cropping test on fingerprinted vector maps: (a) fingerprinted map with the cropping area; (c) Z statistics.

original and the test curves and to extract the control points. Shown in Fig. 2.16(b) is the registered vector map, and Fig. 2.16(c) shows the detection statistics. We can see that the embedded fingerprint can survive affine transformations, and the detection statistic with the correct fingerprint is high after the registration.

Resilience to Vector-Raster Conversion We now examine the resilience to vector-raster conversion coupled with possible affine transformations. Shown in Fig. 2.17(a) is a fingerprinted vector map after raster rendering as a 1100×1100 image and affine transformations. The affine transformation consists of 10-degree rotation, 80% and 140% scaling in the X and Y directions, respectively, and 10- and 20-pixel translation in the X and Y directions, respectively. As the point correspondence is no longer directly available after the vector-raster conversion, we apply the proposed IAM algorithm to estimate the transform parameters and to locate the sample points on test curves corresponding to those used in the embedding. After 15 iterations, we get the registered raster map as shown in Fig.



(a) (b)



(c)

Figure 2.16: Affine transformation test on fingerprinted vector maps: (a) test map; (b) aligned map; (c) Z statistics.

2.17(b) and the detection statistics as shown in Fig. 2.17(c). The detection statistic results suggest that the embedded fingerprint is identified with high confidence. Using the same settings on the computing system as in Section 2.3.3, we measure the detection time required for this transformed raster map. The total duration of the detection process is 42.43 seconds, including 40.05 seconds for curve registration and fingerprint extraction from nine curves, and 2.38 seconds for evaluating Z statistics with 1000 fingerprint sequences.

Similar to the collusion test, we plot in Fig. 2.18 the histogram and the Gaussian approximation of the Z statistics for this vector-raster conversion test combined with geometric transformations. The transform parameters are randomly

selected from the following ranges with a uniform distribution: $-20 \sim +20$ degrees of rotation, $60\% \sim 140\%$ scaling in the X and Y directions, and $20 \sim 40$ pixels' translation in the X and Y directions. As the errors from registration and re-sampling are not always i.i.d. Gaussian distributed, we observe a variance larger than 1 for the Z values in the fingerprint presence case.

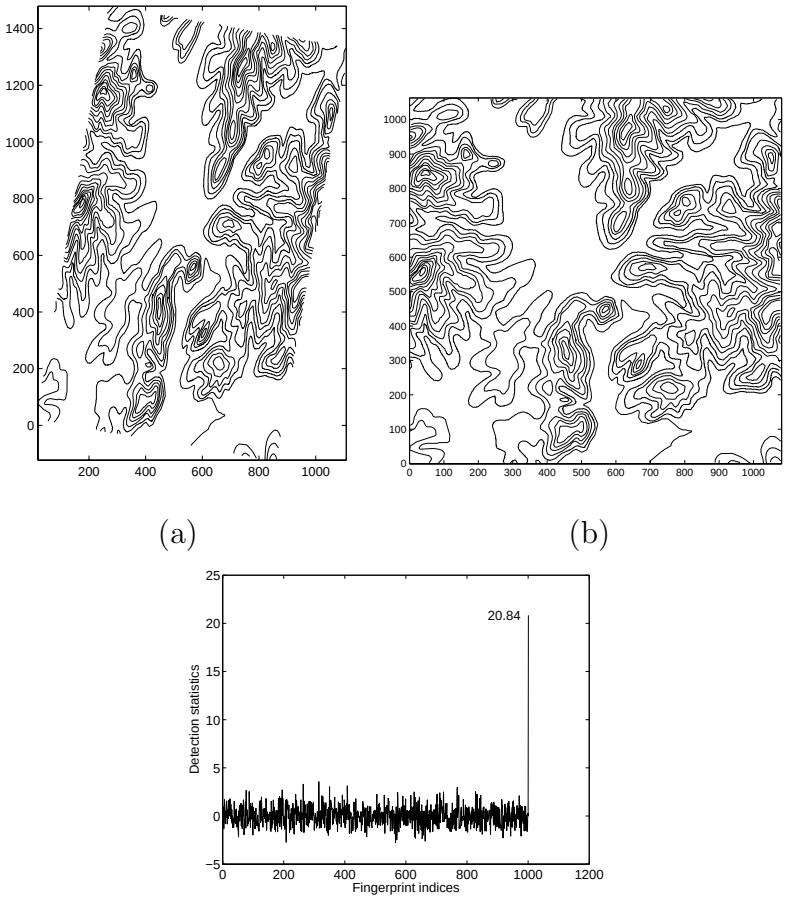


Figure 2.17: Affine transformation test on fingerprinted raster maps: (a) test map; (b) aligned map; (c) Z statistics.

Resilience to Vector-Raster-Vector Conversion To demonstrate the resilience of our method to vector-raster-vector conversion, we first render a fin-

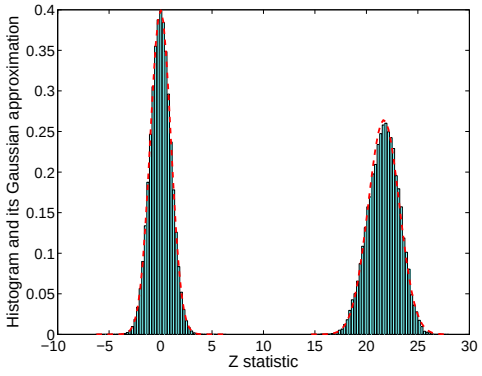


Figure 2.18: Histogram and Gaussian approximation $\mathcal{N}(0, 1.00)$ and $\mathcal{N}(21.69, 2.28)$ of the Z detection statistics for the combined vector-raster conversion and geometric transformations.

gerprinted vector map as a raster map. Then, we uniformly sample it to obtain a new vector map, which has the same number of vector points as prior to the conversion but at different sampling locations. In the detection process, we first render this "new" vector map as a raster map by linear interpolation and then apply our IAM algorithm. From the high detection statistic shown in Fig. 2.19(a), we can see that our approach is robust against the vector-raster-vector attack.

Resilience to Point Deletion in Vector and Raster Maps As we have seen throughout the chapter, traitor tracing applications usually involve adversaries who have strong incentives to remove fingerprints. Attackers may delete a certain number of points from a fingerprinted vector/raster map, while keeping similar shapes of its contour lines. Shown in Fig. 2.19(b) and (c) are detection statistics after point deletion in a fingerprinted vector and raster map, respectively. For the vector map, 20% of points are randomly chosen and removed from each fingerprinted curve, while in the raster map 70% of black pixels on the curve are randomly chosen and removed. We can see that the embedded fingerprints can

survive point deletion applied to both vector maps and raster maps. Similar to the vector-raster-vector conversion test, linear interpolation and the IAM algorithm are used in the fingerprint detection.

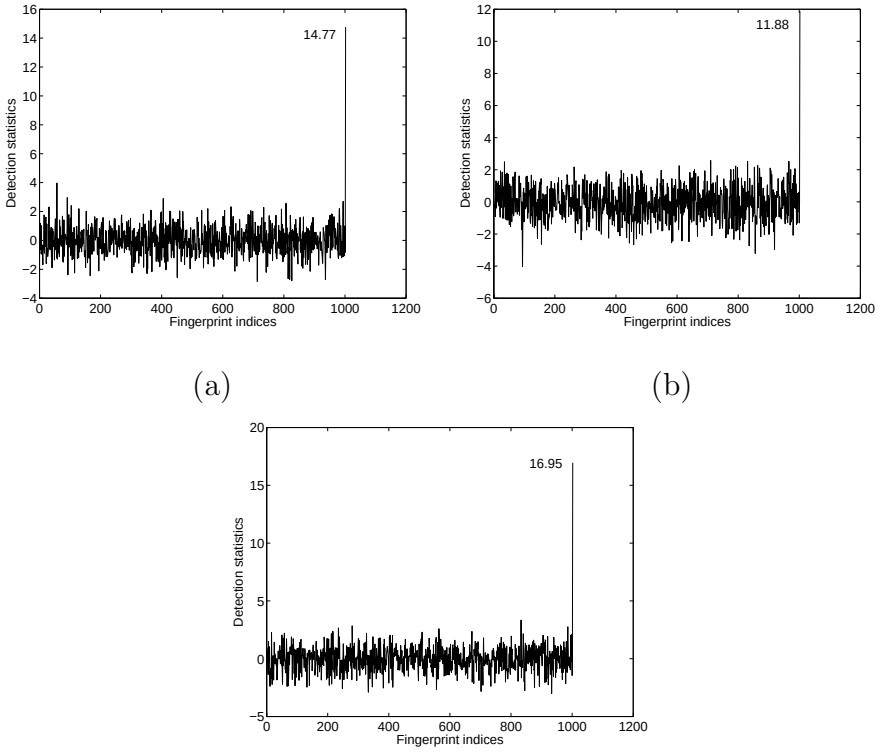


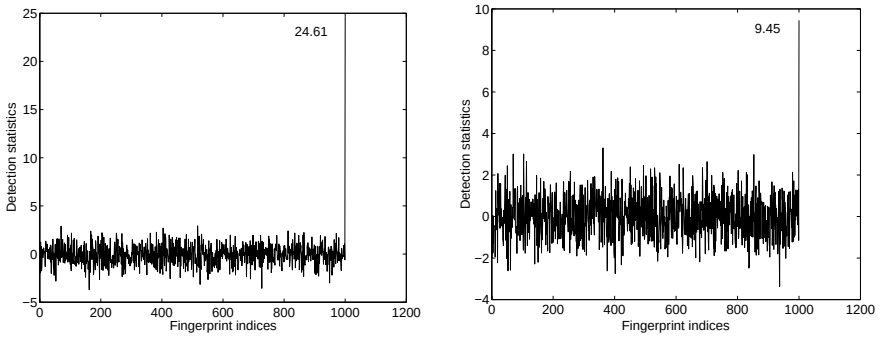
Figure 2.19: Detection results after various attacks: (a) Z statistics for vector-raster-vector conversion; (b) Z statistics for point deletion in vector maps; (c) Z statistics for point deletion in raster maps.

Resilience to Curve Smoothing Similar to lowpass filtering for images, curve smoothing can be applied to topographic maps as an attempt to remove the embedded fingerprint. In order to demonstrate the resilience of the proposed method to curve smoothing, we traverse each marked curve and apply a moving average filter to it. A curve point with coordinates (r_{x_i}, r_{y_i}) will be replaced by a new

point, whose coordinates $(r_{x_i}^{(s)}, r_{y_i}^{(s)})$ are obtained by

$$\begin{cases} r_{x_i}^{(s)} = \frac{1}{2S+1} \sum_{j=-S}^{S} r_{x_{i+j}} \\ r_{y_i}^{(s)} = \frac{1}{2S+1} \sum_{j=-S}^{S} r_{y_{i+j}} \end{cases}, \tag{2.25}$$

where $2S + 1$ is the filter length. Finally, we apply the proposed IAM algorithm to these smoothed curves and compute the detection statistics. Two different filter lengths, 5 and 21, are used in our experiments. As shown in Fig. 2.20, the detection statistic with the correct fingerprint under 5-point averaging is 24.61, and that under 21-point averaging is 9.45. Indeed, the fingerprint is weakened by the smoothing operation, but the detection statistics are still well above the threshold for a correct positive detection. In the latter case when a long filter is used in the smoothing attack, some visual details have been lost from the curves. This study shows that the proposed method is robust against curve smoothing, provided that the smoothing does not severely change the shape of the curve, and that the fingerprint sequence is sufficiently long to help the detector collect information for a positive detection.



(a) (b)

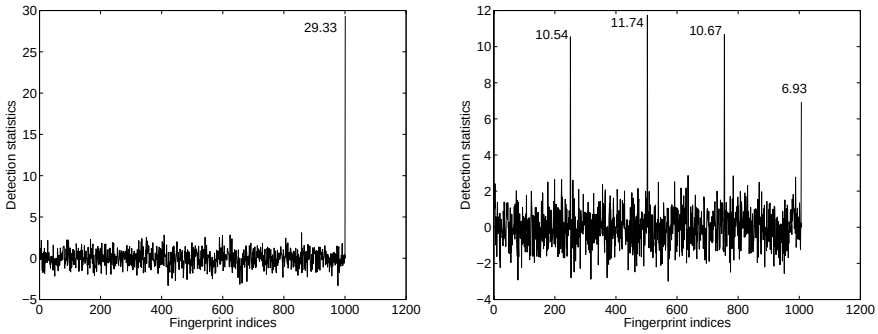
Figure 2.20: Curve smoothing test on fingerprinted raster maps: (a) Z statistics for filter window size 5; (b) Z statistics for filter window size 21.

Resilience to Printing-and-Scanning To show the robustness of our approach against the printing-and-scanning attack, we render a fingerprinted vector map by taking a screen shot of its display in Matlab, print out the image using a HP laser printer, and then scan it back as a binary image using a Canon scanner with 360dpi resolution. Preprocessing before detection includes a thinning operation to extract one-pixel wide skeletons from the scanned curves that are usually several-pixel wide after high-resolution scanning. As we can see from the detection results in Fig. 2.21(a), the fingerprint survives the printing-and-scanning test and gives reliable positive detection with the detection statistic much higher than the detection threshold.

To further examine our approach under the combined attack of collusion and printing-and-scanning, we first generate a colluded map by averaging coordinates of the corresponding control points from four users' fingerprinted maps, then render and print it out, and scan it back as a binary image. From the detection statistics in Fig. 2.21(b), we can see that the embedded fingerprints from all the four colluders can be correctly identified after this combination of attacks involving collusion, vector-raster conversion, filtering, and affine transformations.

2.5 Chapter Summary

In summary, we have proposed a new data embedding algorithm for curves by parameterizing a curve using the B-spline model and adding spread spectrum sequences to B-spline parameters. In conjunction with the basic embedding and detection techniques, we have proposed an iterative alignment-minimization algorithm to allow for robust fingerprint detection under unknown geometric transformations and in the absence of explicit point correspondence. We have demon-



(a) (b)

Figure 2.21: Printing-and-scanning test: (a) Z statistics for printing-and-scanning; (b) Z statistics for 4-user averaging collusion combined with printing-and-scanning.

strated the fidelity of our method as well as its robustness against collusion, cropping, affine transformations, vector-raster and vector-raster-vector conversions, point deletion, curve smoothing, printing-and-scanning, and their combinations. Our work has shown the feasibility of the proposed algorithm in fingerprinting applications for tracing and tracking topographic maps as well as some other curve-based graphic data such as writings/drawings from pen-based inputs. Protecting the above graphic data to ensure its proper distribution has increasing importance to the emerging digital operations in government, military, and commercial agencies.

Chapter 3

Fingerprinting Digital Elevation Maps (DEMs)

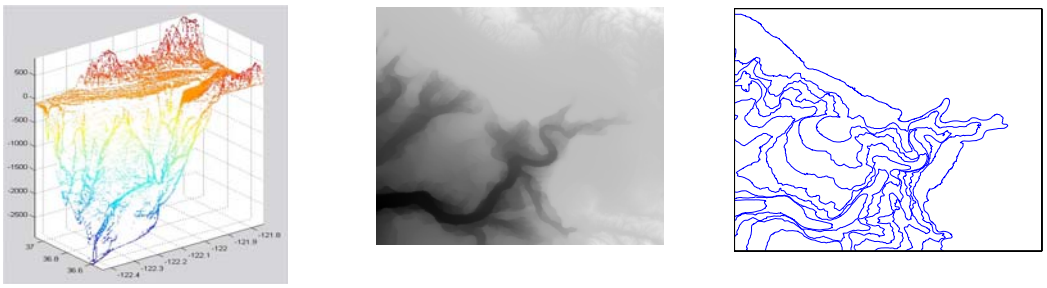
In addition to 2-D topographic maps, geospatial data is also acquired and archived as digital elevation maps (DEMs). A DEM provides a digital representation of surface terrains with a set of points in the three-dimensional space. In civilian applications, high-precision DEMs carry a high commercial value owing to the large amount of effort in acquiring them; and in military applications, DEMs are often used to represent critical geospatial information in sensitive operations. These call for new technologies to prevent unauthorized distribution of DEMs and to trace traitors in the event of information leak. In this chapter, we propose a new digital fingerprinting technique to protect DEM data from illegal re-distribution. The proposed method enables reliable detection of fingerprints from both the 3-D DEM data set and its 2-D rendering, whichever format that is available to a detector. Our method starts with extracting from a DEM a set of critical contours either corresponding to important topographic features of the terrain or having application-dependent importance. Fingerprints are then embedded into these critical con-

tours by employing parametric curve modeling and spread spectrum embedding discussed in Chapter 2. Finally, a fingerprinted DEM is constructed to incorporate the marked 2-D contours. Through experimental results, we demonstrate the cross-dimensional fingerprint detection capability of the proposed method, as well as its robustness against a number of challenging attacks applied to either DEMs or their contour representations.

3.1 Introduction

Digital elevation maps (DEMs) provide three-dimensional (3-D) measurements (x, y, z) of surface terrains. The coordinates (x, y) indicate the spatial location of a point, while the z coordinate represents its elevation. An example DEM for the Monterey Bay region is shown in Fig. 3.1(a), where the ocean-floor depth data are obtained from the U.S. National Geophysical Data Center (NGDC) [3]. In recent years, acquisition systems used to obtain DEM data have been improved to achieve better resolutions in both the spatial plane and the elevation coordinate. With such improvement, DEMs have become more widely used in military and commercial operations, such as navigation, landing, petroleum exploration, and land use planning. Because of the large amount of efforts that are put in acquiring DEM data as well as their critical role in practical applications, many DEMs have high commercial values and need to be protected from unauthorized copying and re-distribution. Furthermore, DEM data used in military applications often contain sensitive information, which prompts the need of preventing information leak as well as tracing the source of leak.

The data points of DEMs are generally acquired and archived using a rectangular grid. If we take the spatial location (x, y) as the image coordinates, and the



(a) (b) (c)

Figure 3.1: Digital representation of Monterey Bay, California: (a) the 3-D DEM; (b) the corresponding representation in 2-D gray-scaled image; (c) a 2-D topographic map of the 3-D DEM.

elevation value z as the image intensity, a 3-D DEM can be represented by a gray-scaled image, as shown in Fig. 3.1(b). Both the 3-D DEM and its corresponding gray-scaled image representation can convey the same elevation information. A DEM generally contains much redundant information in that neighboring points in many areas have gradually changing elevations. Therefore, applications of the DEM often focus on elevation contours extracted from the DEM, as opposed to the DEM itself [53]. From a given DEM, a set of contours corresponding to certain elevations can be extracted, forming a two-dimensional (2-D) topographic map as shown in Fig. 3.1(c). Generally, topographic maps can convey typical geographic features of surface terrains. Such meaningful cross-dimensional rendering of DEMs prompts the need of cross-dimensional fingerprint detection capability, that is, fingerprints shall be detected from both marked 3-D DEMs and their 2-D contour rendering.

Since DEMs can be represented as gray-scaled images, approaches developed for robustly watermarking images may be employed to fingerprint DEMs. One candidate is spread spectrum embedding in the DCT (Discrete Cosine Transform)

or DFT (Discrete Fourier Transform) domain [20]. However, such image watermarking methods do not explicitly take the geographic meanings of DEMs into consideration, and cannot satisfy the requirement of cross-dimensional fingerprint detection. When an aggressive attacker renders his/her marked 3-D DEM to be some meaningful 2-D contours but removes all the other information, the fingerprint information embedded in the DCT or DFT coefficients is most likely to be destroyed along with the dimension reduction.

In order to combat such attack of rendering 3-D DEMs to 2-D contours as well as other attacks and distortions to either DEMs or their contours, we extend our curve fingerprinting method in Chapter 2 to fingerprint DEMs through hiding fingerprints in the 2-D contours of a 3-D DEM data set. We start with extracting from a DEM a set of critical contours either corresponding to important topographic features of the terrain or having certain application-dependent importance. Fingerprints are then embedded into these critical contours, and finally a fingerprinted DEM is constructed to incorporate these marked 2-D contours. Our approach has the potential to achieve cross-dimensional fingerprint detection, making the fingerprint detectable from both the DEMs and their 2-D contour based rendering, whichever available to a fingerprint detector.

The rest of this chapter is organized as follows. Section 3.2 provides a framework of fingerprinting a DEM through hiding data in its contours. Section 3.3 describes the transformation between a 3-D DEM and its 2-D contour curves. In Section 3.4, we discuss the tradeoff between fingerprint robustness and imperceptibility, and address a special attack named contour replacement. Experimental results are presented in Section 3.5 to demonstrate the effectiveness of the proposed approach. Finally, we summarize this chapter in Section 3.6.

3.2 Framework of Cross-Dimensional DEM Fingerprinting

When fingerprinting a DEM through hiding data in its 2-D contours, the first step is to determine which contours to select for hiding fingerprints. If we select some contours to embed the fingerprint but adversaries are likely to be interested in other elevations, they may carefully re-render the DEM data into a different set of 2-D contours to avoid the marked elevations. Just like a natural image has perceptually significant components, a DEM has some critical contours corresponding to either important topographic features of a terrain such as the maxima, the minima, and the saddle points [77], or some objects with application-dependent importance such as oil wells. As critical contours are essential to a DEM for largely determining terrain features or conveying valuable geospatial information, attackers are often not willing to completely remove or seriously distort them. Therefore, we propose to extract from a given DEM those critical contours to carry the fingerprint.

After identifying the contours to carry the hidden data, we fingerprint a DEM through proper transformations between the 3-D DEM and its 2-D contours, and leveraging a curve fingerprinting method developed in Chapter 2. A framework of the proposed cross-dimensional DEM fingerprinting scheme is illustrated in Fig. 3.2, and the main steps of fingerprint embedding and detection [39] are summarized below.

In the embedding process, we first identify and extract from the given DEM a set of critical 2-D contours. Then, we embed the fingerprint sequence into these contours by employing B-spline based parametric curve modelling and spread spectrum embedding discussed in Chapter 2. Finally, we construct a fingerprinted DEM

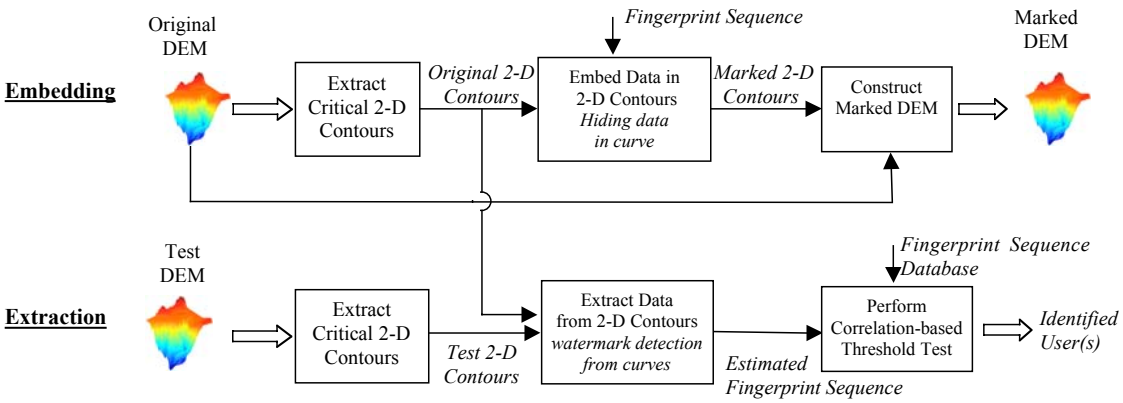


Figure 3.2: The block diagram of fingerprinting 3-D DEMs through hiding data in 2-D contours.

by modifying the elevation values of the original 3-D DEM to incorporate the marked 2-D contours. To detect the fingerprint(s) from a suspicious DEM, we first extract from it the critical contours at the elevations corresponding to those used in the embedding. If the DEM data set in question has already been rendered in 2-D contours, these rendered contours will be passed directly to the detector. Then, we estimate a fingerprint sequence from these 2-D extracted/rendered contours, using the IAM algorithm proposed in the Chapter 2. Finally, correlation-based detection is performed to identify the sources of information leak.

3.3 Transformation between 3-D DEM and 2-D Curve

Taking the fingerprint embedding and detection in the curve domain as building blocks to hide/extract fingerprints in/from the 2-D contours of the 3-D DEM data, the new issue remaining to be addressed in our scheme is how to carry out proper transformations between a 3-D DEM and its 2-D contours. In the following

subsections, we discuss how to extract 2-D contours from a 3-D DEM data set as well as how to construct a fingerprinted 3-D DEM from marked 2-D contours.

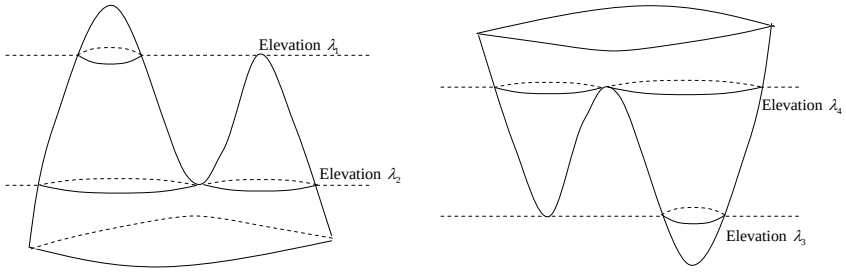
3.3.1 Extracting Critical 2-D Contours from a DEM

Critical contours of a DEM either identify certain topographic features of the terrain or depend on its usage in particular applications. Correspondingly, elevations of critical contours, or *critical elevations* in short, can be obtained from analyzing terrain features, or have been specified by the applications. In literature, the Morse theory has been used to identify critical elevations corresponding to such topographical features as maxima, minima, and saddle points [77].

Formulate a DEM as a function $u : D \to \mathbb{R}$, where $D \subset \mathbb{R}^2$ represents (x, y) locations, and the function u specifies the elevation value of each location. The upper level set of an elevation λ, denoted as $[u \geq \lambda]$, consists of locations where the height is greater than or equal to λ, i.e., $[u \geq \lambda] \triangleq \{(x, y) \in D : u(x, y) \geq \lambda\}$. Similarly, the lower level set of an elevation λ is defined as $[u \leq \lambda] \triangleq \{(x, y) \in D : u(x, y) \leq \lambda\}$. More generally, for $\lambda, \mu \in \mathbb{R}$, and $\lambda \leq \mu$, we can define a set $[\lambda \leq u \leq \mu] \triangleq \{(x, y) \in D : \lambda \leq u(x, y) \leq \mu\}$. Further, the contour set of an elevation λ is defined as $[u = \lambda] = [\lambda \leq u \leq \lambda]$.

Shown in Fig. 3.3 are two major types of terrain, where maxima appear in the Type-I terrain, and minima appear in the Type-II terrain. There are also two kinds of saddle points, one with each type of terrain. To identify maxima/minima and saddle points from these two types of terrain, we consider a DEM with elevation values in the range of $[a, b], a \leq b$, i.e., $u : D \to [a, b]$, and examine how the contour set $[u = \lambda]$ deforms as λ continually decreases from b to a. For the Type-I terrain shown in Fig. 3.3(a), each time when we come across a maximum, a small curve

appears in the contour set, as the situation for elevation λ_1. If there is a saddle point, as for elevation λ_2, two relatively short curves merge into a single curve. For the Type-II terrain shown in Fig. 3.3(b), when decreasing the elevation from b to a, a small curve disappears at each minimum, e.g. at elevation λ_3, while a single curve splits into two short ones at a saddle point, e.g., at elevation λ_4. By observing the above topology changes along with the moving contour set, critical elevations corresponding to maxima, minima, and saddle points can be identified.



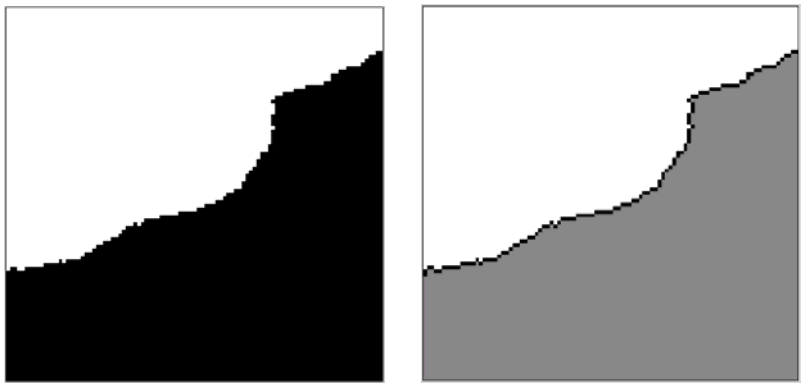
(a) (b)

Figure 3.3: The two types of terrain and their critical elevations: (a) Type-I with maxima and saddle points; (b) Type-II with minima and saddle points.

As there may be many small oscillations in a terrain, directly employing the above method may result in many critical elevations, some of which may not represent significant topographic features of the terrain. In order to find major critical elevations, Extrema filters are utilized in [77] to remove small oscillations while preserving major topographic changes. Applying Extrema filtering to a DEM, elevation values around maxima(mimina) are decreased(increased) by a certain amount so that small oscillations can be flattened. From such a filtered DEM, major critical elevations can then be identified using the Morse theory described above.

After obtaining the critical elevations, either directly specified by particular

applications or found using the Morse theory, we now extract their corresponding critical contours. For each critical elevation λ, we first identify its upper level set $[u \geq \lambda] \triangleq \{(x, y) \in D : u(x, y) \geq \lambda\}$. Assigning $0's$ to the locations in the upper level set and $1's$ to the other locations, we then generate a binary image B_λ. Convolving a 3×3 filter with 1's at the 4-connected neighbors $h = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ with B_λ, we identify boundary points of the upper level set $[u \geq \lambda]$ as those in $[u \geq \lambda]$ and with a positive value in the filtered image $B_\lambda \otimes h$, and take them as the contour at elevation λ, i.e., $[u = \lambda] = \{(x, y) \in D : B_\lambda(x, y) = 0 \wedge B_\lambda \otimes h(x, y) > 0\}$. In Fig. 3.4, we show an example of contour detection using the above method. Finally, we employ a curve following algorithm in [11] to traverse the contour and represent it using a set of ordered curve points.



(a) (b)

Figure 3.4: An example of contour detection: (a) binary image B_λ with black points in the upper level set; (b) detected contour points represented in the black color.

3.3.2 Constructing Fingerprinted DEM from Marked 2-D Contours

In the final step of fingerprint embedding, we construct a fingerprinted DEM to incorporate those marked critical contours so that the fingerprint embedded in 2-D contours is also detectable from this newly constructed 3-D DEM. In order to reliably detect the fingerprint embedded in the 2-D contours, we need to make the marked 2-D contours recoverable from this newly constructed DEM with high accuracy. Additionally, in order to preserve the geospatial information conveyed by the DEM, we need to make the distortion between this newly constructed DEM and the original DEM as small as possible.

As shown in Fig. 3.5, the original contour at elevation λ is present as the boundary of the upper level set $[u \geq \lambda]$ for the original DEM $u : D \rightarrow \mathbb{R}$. After curve-based fingerprinting, the original contour is slightly deviated to be a marked contour (we exaggerate the difference between the original and the marked contours in Fig. 3.5 for illustration). Such 2-D contour deformation can be implemented through appropriately modifying elevation values in the 3-D digital elevation map. In the modified DEM $u' : D \rightarrow \mathbb{R}$ (i.e., the fingerprinted DEM in our context), the marked contour shall be detected as the boundary of a new upper level set $[u' \geq \lambda]$ for the same elevation λ. To accomplish this, we modify elevation values in the non-overlapping regions of the two upper level sets $[u \geq \lambda]$ and $[u' \geq \lambda]$. Specifically, we increase the elevation value to be λ for locations that are in $[u' \geq \lambda]$ but not in $[u \geq \lambda]$ ("+" region in Fig. 3.5), decrease it to be slightly smaller than λ for those that are in $[u \geq \lambda]$ but not in $[u' \geq \lambda]$ ("-" region in Fig. 3.5), and keep

it the same as in the original DEM for all the other locations:

$$u'_{\lambda\to\lambda}(x,y) = \begin{cases} \lambda & (x,y) \in [u' \geq \lambda] \wedge (x,y) \notin [u \geq \lambda] \\ \lambda - \delta & (x,y) \notin [u' \geq \lambda] \wedge (x,y) \in [u \geq \lambda] \\ u(x,y) & \text{otherwise} \end{cases}, \tag{3.1}$$

where δ is a positive number to specify the amount of elevation adjustment. One possible choice of δ is to use the smallest unit of the elevation coordinate, which is determined by the DEM vertical resolution.

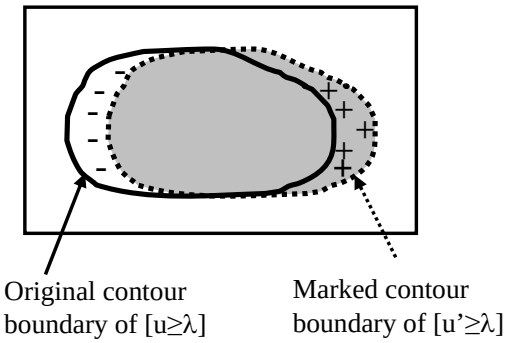


Figure 3.5: Construct the fingerprinted 3-D DEM.

When the marked contour maintains high fidelity with respect to its original version, non-overlapping regions between the two upper level sets $[u \geq \lambda]$ and $[u' \geq \lambda]$ are actually very small. Using (3.1) with the smallest elevation unit for δ, we further make the minimal amount of changes in the fingerprinted DEM to incorporate the marked contours. Therefore, our fingerprinted DEM can preserve the geospatical information conveyed by the original DEM with high precision.

3.4 Robustness and Fidelity Considerations

In this section, we discuss the tradeoff between embedding fidelity and fingerprint's robustness when fingerprinting a 3-D DEM through hiding data in its rendered 2-D contours. We also propose a new attack named contour replacement and discuss the robustness of our method against this challenging attack.

3.4.1 Embedding Fidelity and Fingerprint's Robustness

When fingerprinting a DEM, we hide the fingerprint sequence into its critical contours. If the total length of contours that are used in the embedding is larger (e.g., we may use more contours and/or longer contours), most likely we can have a larger number of B-spline control points that can host a longer fingerprint sequence to achieve stronger resilience against distortions. However, when more and/or longer contours are manipulated to carry a longer fingerprint sequence, larger distortions may be introduced to the fingerprinted DEM.

We now examine the fidelity and robustness issues in the construction of fingerprinted 3-D DEMs. To achieve cross-dimensional fingerprint detection, we construct the 3-D fingerprinted DEM $u' : D \rightarrow \mathbb{R}$ in a way that the original 2-D contour can be accurately deformed to be a marked 2-D contour, as in (3.1) of Section 3.3.2. During the contour deformation, elevation values in the non-overlapping regions ($C \subset \mathbb{R}^2$) of the two upper level sets $[u \geq \lambda]$ and $[u' \geq \lambda]$ are either increased or decreased. A non-trivial elevation difference may be observed when a point $(x, y) \in [u' \geq \lambda] \wedge (x, y) \notin [u \geq \lambda]$ has an elevation value much smaller than λ, because it shall be increased to λ in the fingerprinted DEM u'. Similarly, for a point $(x, y) \in [u \geq \lambda] \wedge (x, y) \notin [u' \geq \lambda]$ with an elevation value much larger than λ, its elevation will be decreased to $\lambda - \delta$ in the fingerprinted DEM u', which also

results in a non-trivial distortion. To preserve high fidelity of the fingerprinted DEM, we propose to employ thresholding on the point-wise elevation distortion to determine the points in the non-overlapping regions C that should actually change their elevation values. Setting a threshold $\tau > 0$, a point $(x, y) \in C$ gets its elevation changed from the original $u(x, y)$ to $u'(x, y)$ only when $|u(x, y) - u'(x, y)| \leq \tau$ is satisfied:

$$u'_{\lambda \to \lambda, \tau}(x, y) = \begin{cases} \lambda & (x, y) \in [u' \geq \lambda] \wedge (x, y) \notin [u \geq \lambda] \wedge \lambda - u(x, y) \leq \tau \\ \lambda - \delta & (x, y) \notin [u' \geq \lambda] \wedge (x, y) \in [u \geq \lambda] \wedge u(x, y) - (\lambda - \delta) \leq \tau \\ u(x, y) & \text{otherwise} \end{cases} . \quad (3.2)$$

The thresholding with τ can effectively limit the maximal point-wise elevation distortion in the fingerprinted DEM to be no more than τ. However, it may affect the accuracy of deforming the original contour to the target marked contour, i.e., deviation may exist between the contour extracted from the fingerprinted DEM and the marked contour generated from fingerprinting the original contour. Such deviation has a negative effect on the detection accuracy of fingerprints that are virtually embedded in contours. Therefore, we need to make a tradeoff between embedding fidelity and fingerprint robustness, and the threshold τ can be tuned according to application requirements.

3.4.2 Robustness against Contour Replacement Attack

Through hiding data in critical 2-D contours of a DEM, our method is effective in combating the attack of rendering a fingerprinted DEM to a topographic map, which contains some or all of the marked 2-D contours because of their critical role in conveying geospatial information. As neighboring points in a DEM often have gradually changing elevation values, a fingerprinted contour can be approximated

by or inferred from its neighboring contours. Taking advantage of this property, an aggressive attacker may move one step further to remove the fingerprinted contour itself but preserve some of its neighboring contours during the 2-D rendering. We refer to this attack as a contour replacement attack, and discuss how to make our method resilient to this challenging attack.

Band Deformation to Combat Contour Replacement Attack Use the contour at elevation λ to carry the fingerprint. Now, we consider the following contour replacement attack. In the 2-D rendering of the 3-D fingerprinted DEM, the attacker removes the marked contour at elevation λ, but preserves one of its neighboring contours at elevation $\beta \in [\lambda_{min}, \lambda_{max}]$ ($\lambda_{min} \leq \lambda \leq \lambda_{max}$). To combat such a contour replacement attack, we modify the construction of the fingerprinted 3-D DEM so that all contours in the "elevation band" of $[\lambda_{min}, \lambda_{max}]$ will be deformed to be the target marked contour. Considering the property of $[u \geq \lambda_1] \supset [u \geq \lambda_2]$ and $[u < \lambda_2] \supset [u < \lambda_1]$ when $\lambda_1 < \lambda_2$, such band deformation can be obtained through jointly deforming the two contours at the two elevation bounds λ_{min} and λ_{max}. Correspondingly, we modify DEM data values as follows:

$$
\begin{aligned}
u'_{\lambda \to \lambda_{min}}(x,y) &= \begin{cases} \lambda_{min} & (x,y) \in [u' \geq \lambda] \wedge (x,y) \notin [u \geq \lambda_{min}] \\ \lambda_{min} - \delta & (x,y) \notin [u' \geq \lambda] \wedge (x,y) \in [u \geq \lambda_{min}] \\ u(x,y) & \text{otherwise} \end{cases} \\
u'_{\lambda \to \lambda_{max}}(x,y) &= \begin{cases} \lambda_{max} & (x,y) \in [u' \geq \lambda] \wedge (x,y) \notin [u \geq \lambda_{max}] \\ \lambda_{max} - \delta & (x,y) \notin [u' \geq \lambda] \wedge (x,y) \in [u \geq \lambda_{max}] \\ u(x,y) & \text{otherwise} \end{cases}
\end{aligned}, \quad (3.3)
$$

where δ is still a positive number that can be as small as the smallest unit of the elevation coordinate. Because $[u \geq \lambda_{min}] \supset [u \geq \lambda_{max}]$ and $[u < \lambda_{max}] \supset [u <$

$\lambda_{min}]$ for $\lambda_{min} < \lambda_{max}$, we can further combine the two operations in (3.3) so that the fingerprinted DEM $u' : D \rightarrow \mathbb{R}$ can be constructed as follows:

$$u'_{\lambda \rightarrow [\lambda_{min}, \lambda_{max}]}(x, y) = \begin{cases} \lambda_{max} & (x, y) \in [u' \geq \lambda] \wedge (x, y) \notin [u \geq \lambda_{max}] \\ \lambda_{min} - \delta & (x, y) \notin [u' \geq \lambda] \wedge (x, y) \in [u \geq \lambda_{min}] \\ u(x, y) & \text{otherwise} \end{cases} . \quad (3.4)$$

Through (3.4), for a point (x, y) in the upper level set prescribed by the marked contour at elevation λ, i.e., $(x, y) \in [u' \geq \lambda]$, we increase its elevation value to λ_{max} if its original elevation $u(x, y) < \lambda_{max}$. On the contrary, for a point $(x, y) \notin [u' \geq \lambda]$, we decrease its elevation to be $\lambda_{min} - \delta$ if its original elevation $u(x, y) \geq \lambda_{min}$. All the other points keep their elevation values the same as their original correspondences. With such band deformation, the contour at any elevation $\beta \in [\lambda_{min}, \lambda_{max}]$ is deformed to be the target marked contour. This enables us to successfully combat the contour replacement attack within the elevation range of $[\lambda_{min}, \lambda_{max}]$. However, along with this robustness achievement, the elevation modification according to (3.4) may introduce a large distortion to the fingerprinted DEM u', especially when values of λ_{min} and λ_{max} deviate from λ with a large amount. Thus, here we still need to make a tradeoff between embedding fidelity and fingerprint robustness, and the values of λ_{min} and λ_{max} can also be tuned according to specific application requirements.

To improve the embedding fidelity, one way is to apply the thresholding technique as discussed in Section 3.4.1. A point (x, y) will get its elevation changed from $u(x, y)$ to $u'(x, y)$ according to (3.4) only when $|u(x, y) - u'(x, y)| \leq \tau$. In order to achieve the band deformation so that any contour in $[\lambda_{min}, \lambda_{max}]$ is deformed in the same way toward the target marked contour, the threshold τ shall be no less than the elevation range, i.e. $\tau \geq \lambda_{max} - \lambda_{min}$. Since the value of

τ determines the maximal point-wise elevation distortion after fingerprinting, this again indicates the tradeoff between embedding fidelity and fingerprint robustness. When a contour replacement attack in a larger elevation range is to be combatted, the maximal point-wise elevation distortion to be introduced is also larger.

Gradual Band Deformation to Improve Embedding Fidelity Considering that a marked contour may be more likely to be replaced with a neighboring contour closer to it, it is desirable to gradually adjust the amount of deformation applied to the nearby contours. When a contour has an elevation farther away from the target elevation λ, it may be deformed toward the target marked contour with lower accuracy. In our work, we modify the construction of the fingerprinted DEM u' as follows to combat the contour replacement attack in the range of $[\lambda_{min}, \lambda_{max}]$ with gradual deformation adjustment:

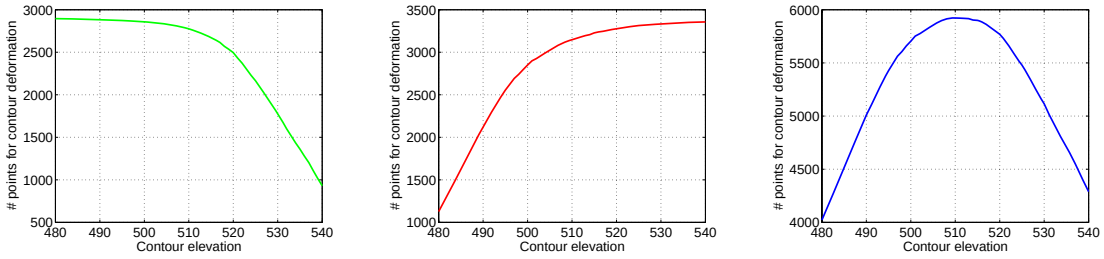
$$u'_{\lambda\to[\lambda_{min},\lambda_{max},\tau]}(x,y) \tag{3.5}$$

$$= \begin{cases} \min\{\lambda_{max}, u(x,y)+\tau\} & (x,y)\in[u'\geq\lambda]\wedge(x,y)\notin[u\geq\lambda_{max}] \\ \max\{\lambda_{min}-\delta, u(x,y)-\tau\} & (x,y)\notin[u'\geq\lambda]\wedge(x,y)\in[u\geq\lambda_{min}] \\ u(x,y) & \text{otherwise} \end{cases}, \tag{3.6}$$

where $\tau > 0$ is a desirable threshold and δ can still be the smallest unit of the elevation coordinate. Specifically, for locations that are in $[u' \geq \lambda]$ but not in $[u \geq \lambda_{max}]$, we increase the elevation value to be the minimal of λ_{max} and $u(x,y)+\tau$, instead of λ_{max} in (3.4). Similarly, for those in $[u \geq \lambda_{min}]$ but not in $[u' \geq \lambda]$, we decrease the elevation value to be the maximal of $\lambda_{min}-\delta$ and $u(x,y)-\tau$.

Through the construction in (3.6), we can achieve a better tradeoff between embedding fidelity and fingerprint robustness. First, we keep the maximal point-wise elevation distortion after fingerprinting to be no larger than a threshold τ.

This is because the maximal increment/decrement happened in (3.6) is limited by τ. Meanwhile, τ can be independent of $[\lambda_{min}, \lambda_{max}]$ since we are not targeting at uniform band deformation. Second, the above construction can achieve gradual contour deformation as follows. As shown in Fig. 3.6(a) by using one contour at elevation $\lambda = 510$ with $\lambda_{min} = 480$, $\lambda_{max} = 540$, and $\tau = 25$ as one example, after the specified elevation increment to $\min\{\lambda_{max}, u(x, y) + \tau\}$, for an elevation value $\beta \in [\lambda_{min}, \lambda_{max}]$, more points are available in the area of $[u' \geq \lambda] \wedge [u < \lambda_{max}]$ deforming the contour at β toward the target marked contour when β is further away from λ_{max} (540 in Fig. 3.6(a)). Similarly, as shown in Fig. 3.6(b) for the elevation decrement case, we have more points in the area of $[u' < \lambda] \wedge [u \geq \lambda_{min}]$ to deform the contour when β is further away from λ_{min} (510 in Fig. 3.6(b)). Since both of the above two point sets contribute to the contour deformation at $\beta \in [\lambda_{min}, \lambda_{max}]$, collectively more points are available for an elevation in the middle range of $[\lambda_{min}, \lambda_{max}]$ (as shown in Fig. 3.6(c) around $\lambda = 510$), and therefore a contour in the middle range can be deformed toward the target marked contour with higher accuracy.



(a) (b) (c)

Figure 3.6: Number of points available for deforming the contour at elevation $\beta \in [\lambda_{min}, \lambda_{max}]$: (a) in the area of $[u' \geq \lambda] \wedge [u < \lambda_{max}]$; (b) in the area of $[u' < \lambda] \wedge [u \geq \lambda_{min}]$; (c) combining the two cases in (a) and (b) together.

3.5 Experimental Results

In this section, we apply the proposed 3-D DEM fingerprinting method to fingerprint two data sets from NGDC [3], namely, the Monterey Bay DEM shown in Fig. 3.1 and the Hawaii DEM shown in Fig. 3.7. We present experimental results to demonstrate the effectiveness of our cross-dimensional fingerprinting method, showing fidelity of fingerprinted DEMs as well as fingerprint robustness against various challenging attacks.

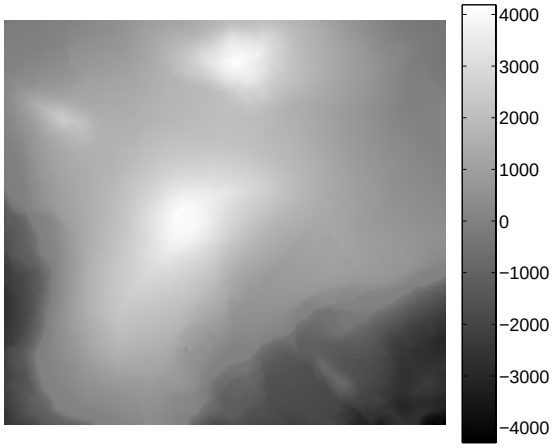


Figure 3.7: Hawaii DEM data set represented as a gray valued image.

For the Monterey Bay DEM data set, we uniformly sample 341×411 points in an area of latitude $36°30'\text{n} \sim 37°4'\text{n}$ and longitude $122°26'\text{w} \sim 121°45'\text{w}$. In our experiments, we focus on its ocean part, whose deepest point has an elevation of -2918 meter. For the Hawaii data set, 551×601 points are uniformly sampled in an area of latitude $19°\text{n} \sim 19°55'\text{n}$ and longitude $155°\text{w} \sim 156°\text{w}$. Here, the land terrain is focused, and elevation of the highest point is 4188 meter. For both data sets, the smallest unit in the elevation coordinate is 1 meter, which is used in our

tests to determine the adjustment parameter $\delta = 1$. Using the Morse theory and the concept of upper level sets as discussed in [77] and presented in Section 3.3.1, we extract from the Monterey Bay DEM three critical contours corresponding to saddle points at elevation -802, -494, and -220 respectively, for carrying fingerprint sequences. For the Hawaii DEM data set, we embed fingerprint sequences into three contours at elevation 510, 891, and 1528, respectively. To generate a fingerprint sequence, we employ a pseudo-random number generator to produce a sequence of independent random numbers and use different seeds for different users. The scaling factor upon fingerprint sequences is set as 1.

Effectiveness of Cross-dimensional DEM Fingerprinting: In our test, we first examine the tradeoff between embedding fidelity and fingerprint robustness by using different thresholds τ in the process of constructing a fingerprinted DEM $u'_{\lambda\to\lambda,\tau}(x,y)$, as discussed in Section 3.4.1. The thresholding rule of $|u(x,y) - u'(x,y)| \leq \tau$ naturally limits the maximal point-wise elevation distortion in the fingerprinted DEM to be at most the threshold τ. In our tests, we also evaluate a Root Mean Square Error(RMSE) of the fingerprinted DEM u' with the original DEM u via

$$RMSE(u,u') = \sqrt{\frac{\sum_{x=1}^{M}\sum_{y=1}^{N}|u'(x,y) - u(x,y)|^2}{M \times N}}, \tag{3.7}$$

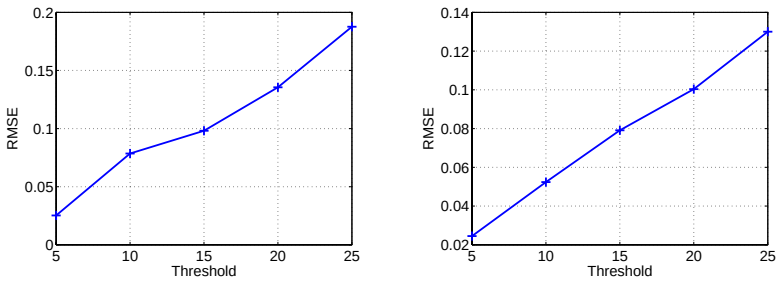
where $M \times N$ is the size of the DEM. As the square root of the mean square error, the RMSE indicates the average elevation difference between two DEMs. As shown in Fig. 3.8(a) for the Monterey DEM, and Fig. 3.8(b) for the Hawaii DEM, with a smaller threshold, the fingerprinted DEM has a smaller RMSE besides a smaller maximal elevation distortion specified by the threshold itself. This is because a smaller threshold can keep more elevation values in the original DEM unchanged during the construction of the fingerprinted DEM. In our experiment,

we test several thresholds in [5, 25]. When the maximal threshold value of 25 is used, the RSME for the Monterey DEM is 0.19, and that for the Hawaii DEM is 0.13. With the minimal threshold of 5, the RMSE can be as small as 0.025 for both the Monterey DEM and the Hawaii DEM. This demonstrates that our method can obtain fingerprinted DEMs with high fidelity. While maintaining the fingerprint invisibility using the thresholding technique, we further examine if the embedded fingerprint can be extracted with high confidence. We compute a Z statistic between the extracted fingerprint with the true fingerprint. As we know from Chapter 2, different thresholds on the Z statistic correspond to different probabilities of false alarm P_{fa}, and we re-list several of them in TABLE 3.1. The selection of the threshold on the Z statistic depends on application requirements, and our following tests shall set it as 4.5 which gives us a P_{fa} on the order of 10^{-6}. As shown in Fig. 3.8(c) for the Monterey DEM and Fig. 3.8(d) for the Hawaii DEM, the larger the threshold τ, the larger the Z statistic with the true fingerprint. With the minimal $\tau = 5$, a Z detection statistic of 5.34 is obtained from the fingerprinted Monterey DEM, and a Z value of 9.25 is from the fingerprinted Hawaii DEM. Combining performance on both embedding fidelity and fingerprint robustness, we demonstrate the effectiveness of our cross-dimensional fingerprinting method in that we can reliably extract the fingerprint embedded into 2-D rendering of a fingerprinted DEM while preserving its high fidelity with the original DEM.

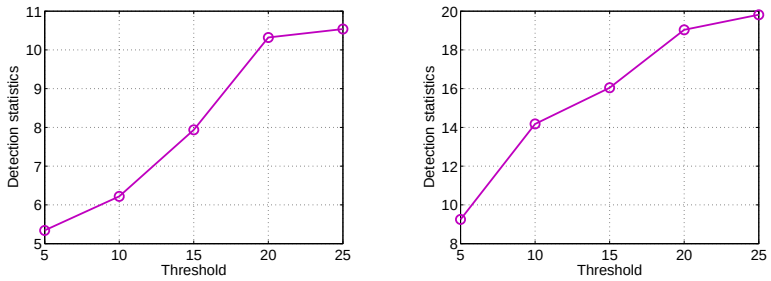
Table 3.1: Thresholds on the Z statistic with corresponding probabilities of false alarm.

Threshold on Z	*3*	*4.5*	*6*	*7.5*	*9*
Probabilities of false alarm P_{fa}	1.3×10^{-3}	3.4×10^{-6}	0.97×10^{-9}	3.1×10^{-14}	1.1×10^{-19}

Fingerprint Robustness against Further Processing: In our second test,



(a) Monterey DEM (b) Hawaii DEM



(c) Monterey DEM (d) Hawaii DEM

Figure 3.8: Tradeoff between imperceptibility and robustness of fingerprints with different thresholds τ used in constructing the fingerprinted DEM: (a-b) RMSE vs. thresholds for the Monterey and Hawaii DEM, (c-d) detection statistics vs. thresholds for the Monterey and Hawaii DEM.

we examine fingerprint robustness against common signal processing operations as well as geometric distortions. In the embedding, we employ the thresholding rule as in (3.2) with a threshold of 20, and obtain a fingerprinted Monterey DEM with RMSE = 0.14, and a fingerprinted Hawaii DEM with RMSE = 0.10. We then act as attackers and apply different distortions to the fingerprinted DEM. From this distorted fingerprinted DEM, we extract the contours corresponding to those used in the embedding, and then perform correlation-based detection. The attacks we test include additive Gaussian noise with a zero mean and a unit standard deviation, 3×3 Gaussian low-pass filtering with a standard deviation $\sigma = 0.5$, and some affine transformations (10° rotation, 1.2× scaling, and 0.8× scaling). In TABLE 3.2, we list the Z detection statistics with the true fingerprint sequences after applying the above attacks to each of the two DEM data sets. We can see all these Z values are higher than the detection threshold of 4.5 (corresponding to a probability of false alarm on the order of 10^{-6}) and suggest positive identification of the correct fingerprint. When more and/or longer contours or larger DEM data sets are used to carry fingerprints, we can have a larger number of markable features. This in turn enables us to obtain higher detection statistics that lead to higher robustness against attacks, and/or to assign orthogonal fingerprint sequences to more users. When attacks and distortions are applied to the 2-D rendering of the DEM consisting of the fingerprinted contours, Chapter 2 has demonstrated the fingerprint's robustness against many challenging attacks such as cropping, geometric distortions, curve smoothing, and printing-and-scanning.

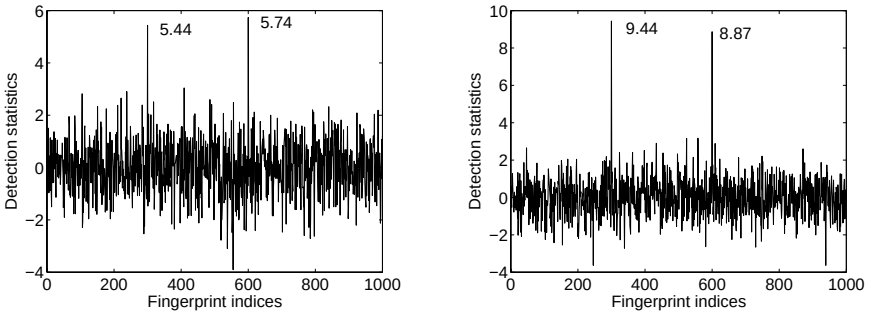
Fingerprint Robustness against Multi-user Collusion: Here we demonstrate the robustness of the proposed method in combating collusion attacks in fingerprinting applications. Using two different fingerprinting sequences, which

Table 3.2: Detection statistics with true fingerprints after various attacks on the fingerprinted DEMs.

Attacks	**Z statistics with fingerperints**	
	Monterey DEM	**Hawaii DEM**
Additive Gaussian noise (σ = 1)	5.89	11.95
Gaussian lowpass filtering (3x3, σ = 0.5)	5.18	8.10
Rotation (10 degree)	6.82	16.86
Scaling (1.2x)	9.41	17.85
Scaling (0.8x)	6.88	16.86

are generated using a random number generator with different keys, we create two fingerprinted copies for each DEM data set. In the embedding, we still apply the thresholding rule in (3.2) with a threshold of 20. On the attacker side, the two fingerprinted DEMs are averaged to be a colluded DEM. Taking this colluded DEM as the test data set, we then extract the target contours and perform correlation-based detection. The detection results are shown in Fig. 3.9(a) for the Monterey DEM, and Fig. 3.9(b) for the Hawaii DEM, respectively. For both data sets, high Z statistics are obtained for the correct positive detection with the fingerprint sequences corresponding to both colluders, while small Z statistics are returned for the correct negative detection with other sequences of innocent users.

Effectiveness of Combating Contour Replacement Attack: Finally, we evaluate the embedding fidelity and detection reliability when using the method proposed in Section 3.4.2 to combat the contour replacement attack. In the following, for a given elevation λ, we use a parameter called elevation tolerance $\triangle\lambda$ to specify the upper and lower elevation bounds (i.e.,λ_{max} and λ_{min}) of its neighboring contours that may be used to replace the fingerprinted contour at elevation



(a) Monterey DEM (b) Hawaii DEM

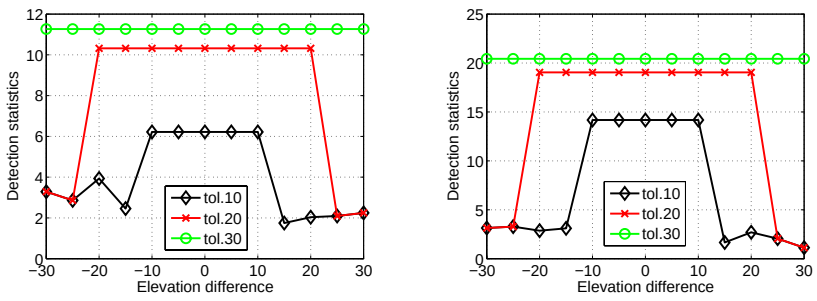
Figure 3.9: Detection statistics with true fingerprint sequences under 2-user averaging collusion.

λ:

$$\begin{cases} \lambda_{min} = \lambda - \triangle\lambda \\ \lambda_{max} = \lambda + \triangle\lambda \end{cases} . \tag{3.8}$$

After getting a marked contour at elevation λ, we use (3.4) to construct a fingerprinted DEM u' to overcome the contour replacement attack in a range of $[\lambda_{min}, \lambda_{max}]$. To maintain fidelity of the fingerprinted DEM, we further enforce the thresholding procedure so that a point (x, y) will keep its elevation unchanged if $|u(x, y) - u'(x, y)| > \tau$. In order to obtain uniform band deformation for combating the contour replacement attack in the range of $[\lambda_{min}, \lambda_{max}]$, it is required that the threshold of $\tau \geq \lambda_{max} - \lambda_{min}$, which is $2\times\ \triangle\lambda$ in our case. Varying the elevation tolerance $\triangle\lambda$ from 10 to 30 with an increment of 10 while setting the threshold $\tau = 2\times\ \triangle\lambda$, we first evaluate the detection performance. Under each value of $\triangle\lambda$, for a marked contour at elevation λ, we extract several of its neighboring contours in the elevation range of $[\lambda - 30, \lambda + 30]$ with an increment of 5. Then, we estimate a fingerprint sequence from each of these neighboring contours, and calculate its Z statistic with the true fingerprint. As shown in Fig. 3.10(a) for

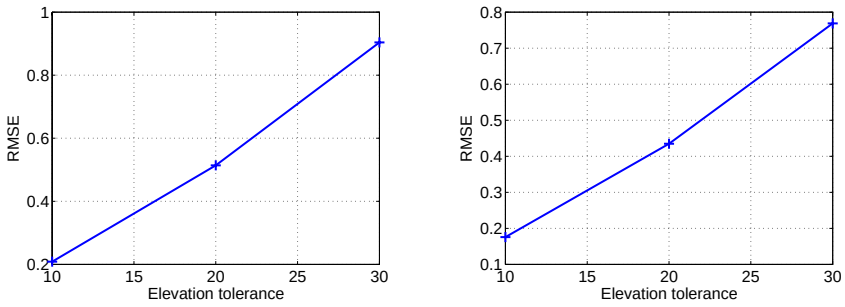
the Monterey DEM, and Fig. 3.10(b) for the Hawaii DEM, respectively, we obtain high and uniformly equal Z statistics with the true fingerprint within each of the designed tolerance ranges. This demonstrates a uniform band deformation which can successfully combat the challenging contour replacement attack. Obviously, a larger elevation tolerance $\triangle\lambda$ enables us to combat a contour replacement attack applied to a larger elevation band. In our test, we also evaluate fidelity of the fingerprinted DEM when the elevation tolerance $\triangle\lambda$ varies. As shown in Fig. 3.11 for both DEM data sets, the RMSE of the fingerprinted DEM with the original DEM increases with the elevation tolerance $\triangle\lambda$. When $\triangle\lambda$ takes the value of 30, the RMSE for the Monterey DEM is 0.90 while that for the Hawaii DEM is 0.77. As for the maximal point-wise elevation distortion, it is limited by the threshold τ which is 20, 40, and 60 corresponding to $\triangle\lambda = 10$, 20, and 30, respectively.



(a) Monterey DEM (b) Hawaii DEM

Figure 3.10: Detection statistics from neighboring contours for different values of the elevation tolerance $\triangle\lambda$.

In the following, we evaluate the performance of using gradual band deformation to combat the contour replacement attack. With an elevation tolerance of $\triangle\lambda = 30$ and a threshold of $\tau = 25$, we construct a fingerprinted DEM u' using the gradual band deformation in (3.6). Then, we apply the same detection procedure from neighboring contours as above, and obtain detection statistics with the true



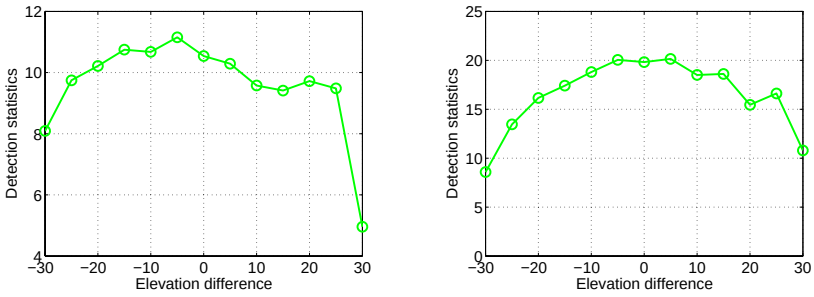
(a) Monterey DEM (b) Hawaii DEM

Figure 3.11: Fidelity of the fingerprinted DEM for different values of the elevation tolerance $\triangle\lambda$.

fingerprints. As shown in Fig. 3.12, the proposed gradual band deformation indeed returns higher detection statistics from neighboring contours that are closer to the target elevation ($\triangle\lambda = 0$). As for the embedding fidelity, we obtain a fingerprinted copy with RMSE = 0.74 for the Monterey DEM, and a fingerprinted copy with RMSE = 0.70 for the Hawaii DEM. Both of them are smaller than their corresponding RMSEs under the uniform band deformation (0.90 for Monterey DEM, and 0.77 for Hawaii DEM). An important improvement of the gradual band deformation over the uniform band deformation is at the maximal point-wise elevation distortion, which is largely decreased from the previous 60 to the current 25. This is because that the threshold τ in the gradual band deformation is no longer required to be larger than $\lambda_{max} - \lambda_{min}$, and a smaller τ gives us a smaller maximal point-wise elevation distortion.

3.6 Chapter Summary

In summary, we have extended the curve-based data embedding to fingerprinting digital elevation maps for preventing them from illegal re-distribution. The



(a) Monterey DEM (b) Hawaii DEM

Figure 3.12: Detection statistics from neighboring contours for gradual band deformation with the threshold of 25.

proposed cross-dimensional data embedding method enables reliable detection of fingerprints from both a 3-D DEM data set and its 2-D rendering, whichever format that is available to a detector. Both embedding and detection of the fingerprint are virtually performed in the domain of contour curves, and the new issue of proper transformations between a 3-D DEM and its 2-D contours has been raised and addressed. We have examined the tradeoff between fidelity of the fingerprinted DEM and robustness of fingerprints, and have also introduced several fine-tuning techniques to adjust the tradeoff.

Chapter 4

Watermarking Binary Images for Authentication and Annotation

A large amount of graphic data are stored and archived in the raster format, i.e., as a binary bitmap image with a 2-D array of black/white pixels. Due to the ease of digital editing on images, we are no longer in a world of "pictures never lie". For a variety of binary images, such as digitized signatures and scanned checks/documents, it is of utmost importance to verify their authenticity and to detect tampering. Extrinsic techniques via data embedding can be used for this authentication purpose [90]. A pre-determined pattern or some content feature is taken as the watermark and then seamlessly embedded into the original image. When the content of the watermarked image is altered, the embedded watermark will change accordingly, providing evidence that the image has been tampered. Such watermarking-based authentication relies on fragility of the embedded data, and generally requires a high embedding payload to accommodate the pre-determined patterns or content features. As there is no "true" image before the authentication, blind detection is required, i.e., the hidden data shall

be extracted without using the original image. In this chapter, we incorporate a recently proposed steganography framework known as the wet paper coding to achieve high-payload data embedding in binary images for the authentication purpose. To maintain good perceptual quality of watermarked binary images under a high embedding payload, we further develop an adaptive trimming method and introduce a new concept of *super-pixels* to address the pixel flippability issue unique to binary graphic data. Having the capability of hiding a large amount of data in binary images with high fidelity can also facilitate steganography and annotation of important raster graphic data in the digital domain.

4.1 Introduction and Prior Art

Hiding data in binary images is a challenging problem. Different from the wide range of brightness levels or colors appearing in gray-valued or color images, black and white are the only two colors in a binary image and they are drastically different to our eyes. Therefore, we cannot rely on minor tuning on the pixel colors to embed data into a binary image, and the only operation that can be taken is black-white or white-black flipping. Further, in order to preserve perceptual quality of marked binary images, those pixels to be flipped for carrying the hidden data must be carefully chosen. It has been shown that such flippable pixels have an uneven spatial distribution in most non-dithered binary images [90]. Studies have shown that when maintaining a simple embedding module to embed one bit in one image block (such as through quantization based embedding), this uneven distribution makes it difficult to hide a large amount of data [89]. This is because the decoder would have to know precisely how many bits are hidden in each part of an image, but there is no room for conveying such overhead information. In order

to hide into a binary image a large amount of data that can be extracted without using the original image (i.e., blind detection), this uneven distribution issue must be properly addressed.

Hiding data in binary images has received a considerable amount of attention since the 1990s. The bi-level constraint limits the extension of many approaches proposed for gray-scale or color images to binary images, and hiding a large amount of data and detecting without the original binary image is particularly difficult for an additive approach [57]. Several methods for hiding data in specific types of binary images have been proposed by taking advantage of unique characteristics of these images. For example, information is embedded in dithered images by manipulating the dithering patterns [42, 64], in other kinds of halftone images by tuning the process of generating a corresponding kind of halftone images [32, 47, 72]. Maxemchuk et al. changed line spacing and character spacing to embed information in textual images for bulk electronic publications [65]. Our B-spline based method in Chapter 2 can also be applied to hiding data in binary images consisting of curve components. However, these approaches cannot be easily extended to binary images which may not have specific structures. In addition, the amount of data that can be hidden using many of these methods is quite limited, and the original host data is often required in the detection.

Applications of authentication, annotation, and steganography generally require high-payload embedding with blind detection. A common framework of such data embedding in binary images is first identifying flippable pixels, which can be flipped from white to black or vice versa without introducing noticeable artifacts. Then, the watermark is embedded through enforcing properties of a group of pixels via local manipulation of a small number of flippable pixels. For example, Matsui

et al. embed data in fax images by manipulating the run-length features [64]. Mei et al. first scan the image in a pre-determined order to extract objects (such as a connected letter or stroke) and then the data is embedded in the outer boundary of an object at a rate of one bit per a five-pixel long boundary if the center pixel of such a boundary pattern is deemed flippable [66]. Koch and Zhao proposed a data hiding algorithm through changing some pixels around contours to enforce the ratio of black vs. white pixels in a block to be larger or smaller than 1 [52]. Enforcing block-based feature to hide data is also used in the work by Wu et al. [90,91], where the uneven distribution of flippable pixels is identified and handled by applying random shuffling to equalize the distribution prior to embedding.

As we shall see later in this chapter, the shuffling approach in [90, 91] can be viewed as a special case of the general paradigm of wet paper coding [27–29] to be employed in our work. Here, we shall review this shuffling-based data hiding technique in more details. Taking the human perception into account, the shuffling approach first assigns a flippability score in $[0, 1]$ to each pixel, quantifying how unnoticeable the flipping of a pixel is to human observers. Pixels with scores high enough can be considered as flippable pixels. In most binary images, the distribution of flippable pixels vary dramatically across the image, with no flippable pixels in the uniformly white or black regions, while quite a few on rugged boundaries. In order to handle such uneven distribution to hide more data while maintaining a simple block-based parity-enforcing embedding, random shuffling is performed prior to embedding. After the shuffling, the formerly identified flippable pixels now occur much more evenly across the image. Then, the shuffled pixels are divided into groups of appropriate size, and if necessary the pixel with the highest flippability score in that group is flipped to enforce the block parity. A bit "0" (or

"1") is embedded by enforcing the total number of black pixels in the group to be an even (or odd) number. Through shuffling, flippable pixels in complex regions and along rugged boundaries can be dynamically assigned to carry more data than less active regions, without the need of specifying much side information that is image dependent. Shuffling also enhances security since the receiver side needs the shuffling table or a key for generating the table to correctly extract the hidden data. A disadvantage of shuffling, however, is that the number of hidden bits is considerably smaller than the total number of flippable pixels and the utilization of flippable pixels is still quite limited. This limitation is in part due to retaining a relatively simple embedding and detection technique for hiding each bit, and to ensure that every group has at least one flippable, the average number of flippables per group needs to be sufficiently large.

In this chapter, we investigate how to incorporate a recently introduced approach to steganography called "writing on wet paper" [27–29] to solve the problem of high-payload, high-fidelity data embedding in binary images. In writing on wet paper, the encoder first identifies a set of k changeable (flippable) pixels and modifies them to embed a secret message. Although the decoder has no knowledge about the location of flippable pixels, the encoder can communicate on average k bits by applying a "wet paper code" (WPC), also known as codes for memory with defective cells [84]. The basic idea of WPC-based embedding is to establish and solve a set of linear equations taking the flippables as unknowns and a pseudo-random binary matrix D as coefficients. Through randomized projections brought by D, the WPC scheme can naturally handle the uneven distribution of flippables in a binary image. By jointly considering the embedding of multiple bits as opposed to sticking with the simple technique of hiding one bit at a time,

the average number of hidden bits can approach the number of flippable pixels, suggesting a potentially significant gain in the embedding payload over the prior art. Along with the high utilization of flippable pixels, a large number of flippable pixels shall be actually flipped in the embedding process and this may affect the perceptual quality of watermarked binary images. To achieve a better tradeoff between the embedding payload and the embedding imperceptibility, we propose an adaptive trimming method to perform flippability assignment on individual pixels in a binary image, and also introduce a new concept of *super-pixels* to address the dependencies among the flippability of a group of pixels. Since a certain level of resilience against noise is desirable in some content-based authentication applications, we also analyze the robustness of WPC-based data embedding against minor processing and distortions.

The rest of this chapter is organized as follows. In Section 4.2, we discuss the incorporation of wet paper codes for data hiding in binary images, including the basic embedding and detection scheme, analysis on the embedding payload, and the relationship between the previous shuffling approach and the new wet paper codes. In Section 4.3, we propose two mechanisms, adaptive trimming and *super-pixels*, with the aim of achieving a better tradeoff between the embedding payload and the perceptual impact of data hiding. Experimental results are provided in Section 4.4 to demonstrate the advantages of using the proposed techniques in hiding data in binary images. In Section 4.5, we discuss the watermark's robustness in the WPC-based data embedding, and compare it with the shuffling approach. Finally, we summarize this chapter in Section 4.6.

4.2 Data Embedding in Binary Images Using Wet Paper Codes

The wet paper coding (WPC) was proposed as a solution for a scenario that frequently occurs in steganography called "writing on wet paper" [27–29]. It is also known in the information theory literature as "writing to a computer memory with defective cells" [41, 84, 96]. To explain this metaphor, imagine that the cover object $\mathbf{x}$ is an image that was exposed to rain and the encoder can only slightly modify the dry spots of $\mathbf{x}$ but not the wet spots. During transmission, the stego image $\mathbf{y}$ dries out and thus the decoder does not know which pixels the sender has modified for carrying the stego information. The task of wet paper coding is to enable both parties to exchange secret messages. The problem of data embedding in binary images fits this "writing on wet paper" paradigm quite well, if we consider that flippable pixels can be viewed as dry spots, and non-flippable ones as wet spots. Since embedding may modify the flippability scores of neighboring pixels, the watermark detector will not be able to correctly identify the pixels that were used for embedding.

Using WPC embedding to replace the shuffling and block-based embedding module in [90, 91], we can build an improved data hiding system for binary images [87]. Shown in Figure 4.1 is a block diagram of the embedding and extraction process of the proposed approach. In the embedding procedure, a set of k changeable (flippable) pixels is first identified, as in the shuffling approach. Then a set of linear equations are established and solved to assign values to the k changeable pixels. To establish the equation set, a pseudo-random binary matrix D that is shared by the data embedder and detector is used. In the detection, the hidden

data can be extracted using the simple matrix multiplication. In the following subsections, we first present a detailed mathematical formulation applying WPC embedding to binary images. We then evaluate the embedding payload, and draw some insights on the relationship between the wet paper codes and the shuffling approach in [90].

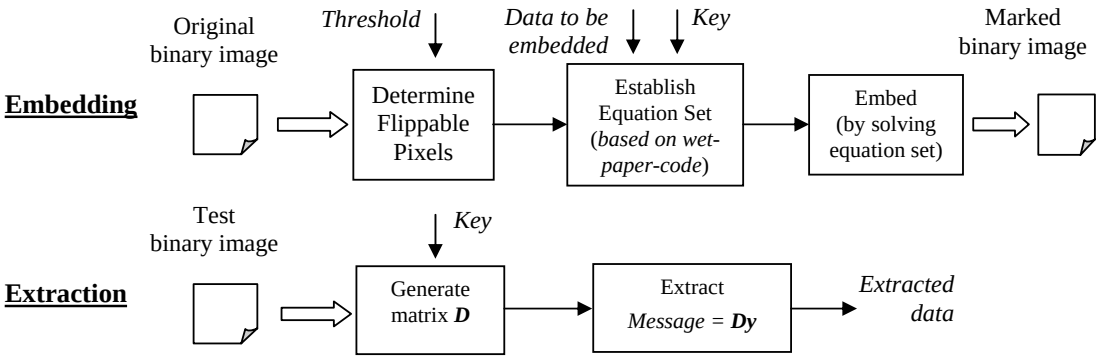


Figure 4.1: Block diagram of the proposed binary image data hiding system employing the Wet Paper Coding.

4.2.1 Basic Embedding and Detection

Consider a cover binary image $\mathbf{x}$ consisting of n elements $\{x_i\}, i = 1, ..., n, x_i \in \{0, 1\}$. Among them, there are k flippable pixels $\{x_j\}, j \in C \subset \{1, 2, ..., n\}$ and $|C| = k$. The embedded image $\mathbf{y}$ also consists of n pixels. The embedding may change a flippable pixel (i.e., $y_j = 1 - x_j$), or leave it unmodified (i.e., $y_j = x_j$). Consider the case of embedding q-bit data $\mathbf{m} = \{m_1, ..., m_q\}^\top$ in $\mathbf{x}$. A secret key is used to generate a pseudo-random binary matrix D of dimensions $q \times n$. The flippable pixels $\{x_j\}, j \in C$ are then modified if needed so that the watermarked binary image $\mathbf{y}$ satisfies

$$D\mathbf{y} = \mathbf{m}. \tag{4.1}$$

Thus, the sender needs to solve a system of linear equations in GF(2) (in binary arithmetic). In the detection, the hidden data $\mathbf{m}$ can be extracted by simply performing a matrix multiplication $\mathbf{m} = D\mathbf{y}$ using the shared matrix D.

In the WPC scheme, the embedding obviously involves bigger computational complexity because it requires solving a system of q linear equations in (4.1). One approach to solving (4.1) with a lower computational complexity is to impose a structure on the columns of matrix D to avoid having to solve the equations altogether. For example, the apparatus of LT codes [59] can be used to generate the columns of D so that their Hamming weight follows the "robust soliton distribution". A modified LT process [28,30] can then be used to solve the system in (4.1) efficiently. As discussed in [30], the modified LT process can also lead to a higher embedding efficiency, which indicates the number of embedded bits per embedding change. Improving on the embedding efficiency of the wet paper codes can also be achieved by employing random linear codes of small codimension [31].

4.2.2 Embedding Payload

The maximal length of a message that can be communicated using wet paper codes is related to the expected rank of the matrix D. Obviously, for small q, (4.1) will have a solution with a very high probability and this probability decreases with increasing q. Rewrite the equation set (4.1) as $D\mathbf{v} = \mathbf{m} - D\mathbf{x}$ using the vector $\mathbf{v} = \mathbf{y} - \mathbf{x}$. Corresponding to the $n-k$ non-flippable pixels that the embedder must keep unchanged, $n-k$ elements of the $\mathbf{v}$ vector will be fixed as zeros. Therefore, there are only k unknown $v_j, j \in C$ in (4.1), and we can remove from D the $n-k$ columns corresponding to the zero v_i's, $i \notin C$. Keeping the same symbol for $\mathbf{v}$, we

now have

$$H\mathbf{v} = \mathbf{m} - D\mathbf{x}, \tag{4.2}$$

where H is a binary $q \times k$ matrix consisting of those columns of D corresponding to indices C, and $\mathbf{v}$ is an unknown $k \times 1$ binary vector. This system has a solution for an arbitrary message $\mathbf{m}$ as long as $\text{rank}(H) = q$. Assume that we always try to embed as many bits as possible by adding rows to D while maintaining that (4.2) still has a solution. Let q_{max} be the expected maximum number of bits that can be embedded in this manner given k flippable pixels. It has been shown that [28]

$$q_{max} = k + O(2^{-k/4}). \tag{4.3}$$

This indicates that the embedding payload can approach the number of flippable pixels, suggesting high utilization of flippable pixels and a potentially significant gain in the embedding payload. It was also shown that the result in (4.3) still holds when 1's and 0's in the random binary matrix H do not occur with the same probability, and the density of 1's in H can be as small as $\log_2(k)/k$ [17, 28, 29].

4.2.3 Relation to Block-based Embedding with Shuffling

Through random projections brought by the random matrix D in (4.1), the WPC paradigm naturally handles the uneven embedding capacity in binary images. Moreover, by jointly considering the embedding of multiple bits, it allows for a high utilization of flippable pixels, thus can significantly improve the embedding payload when compared with the shuffling approach.

Taking a closer look, we find that the block-based embedding with the shuffling approach is actually a special case of the wet paper coding, where the random projection matrix $D_{shuffle}$ is highly structured with only one non-zero entry in

each column. This is because a block is the basic unit for hiding one message bit and the blocks are mutually disjoint. As a result, each pixel is involved in the embedding of only one message bit. Additionally, the randomness of the projection matrix $D_{shuffle}$ is brought in by the shuffling procedure.

In the WPC-based embedding, the random matrix D is defined in a more general sense, which is actually a double-edge sword. On one hand, it allows a flippable pixel to participate in the embedding of multiple message bits, thereby enabling high utilization of flippable pixels. On the other hand, with a non-structured random matrix D, more complex algorithms shall be involved in solving the equation set $D\mathbf{x} = \mathbf{m}$. As randomness appears in both methods and some secret information is required to correctly detect the hidden data, both the shuffling and the WPC schemes enhance the system security by making it difficult to perform tampering while preserving the embedded data.

4.3 Fidelity Considerations for Binary Image Watermarking

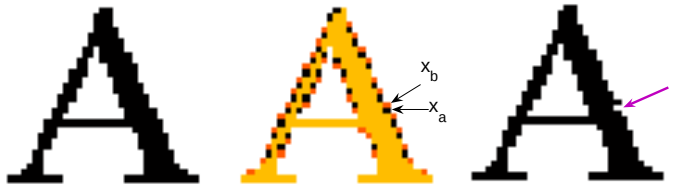
As the WPC based approach allows a much higher utilization of flippable pixels, the amount of pixels being flipped by the embedding process may also be larger than in the shuffling based approach. When more pixels are flipped, even though they are flippable ones, there may be a larger perceptual impact on the watermarked binary image, especially for binary images containing objects that a human observer expects to have a defined shape, such as fonts in text images. To ensure imperceptibility of hidden data while maintaining a high embedding payload, in this section, we propose a method of adaptive trimming to perform

flippability assignment on individual pixels in a binary image, and also develop a new concept of *super-pixels* to address the dependencies among the flippability of a group of pixels.

4.3.1 Adaptive Trimming for Flippability Assignment

In the shuffling approach, a trimming step [90] is employed to ensure any 3×3 window has no more than one pixel with a high flippability score; and such a trimming window can be enlarged to impose a more stringent constraint on the minimum distance between flippable pixels for text images. However, this window based trimming method does not take the image content into consideration, and may largely reduce the number of high-scoring flippable pixels and therefore the embedding rate. To achieve a better tradeoff between watermark imperceptibility and the embedding rate, we propose an adaptive trimming approach to identify high-scoring flippables that will be used in the embedding process in consideration of the content of binary images.

According to smoothness and connectivity measures in [90], a flippability score is assigned to each pixel. Among those with a high score $\{x_j\}, j \in C' \subset \{1, 2, ..., n\}$, there are black pixels $\mathbf{x}_{C'}^{(0)} = \{x_j : j \in C' \wedge x_j = 0\}$ that constitute the foreground, as well as white pixels $\mathbf{x}_{C'}^{(1)} = \{x_j : j \in C' \wedge x_j = 1\}$ that are somewhat further away from the foreground. As shown in Fig. 4.2, flipping two pixels $x_a \in \mathbf{x}_{C'}^{(0)}$ and $x_b \in \mathbf{x}_{C'}^{(1)}$ in the same neighborhood but with different colors generates artifacts with high possibility, since the foreground pixel x_a now becomes a white pixel while its neighbor x_b further away from the foreground is in black instead. To avoid such irregularity, the final high-scoring flippables in the same neighborhood should be in the same color, and we adaptively determine the final flippability score of pixels

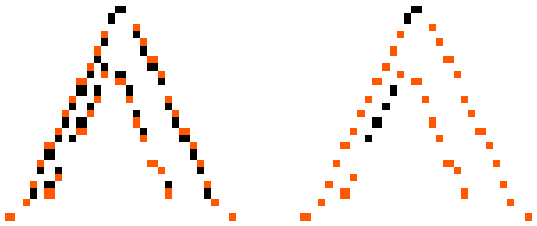


(a) (b) (c)

Figure 4.2: A closer look at flippables: (a) zoomed-in view of character "A" in the original image; (b) pixels with the highest flippability score, with those belonging to $\mathbf{x}_{C'}^{(0)}$ in black color and those belonging to $\mathbf{x}_{C'}^{(1)}$ in orange color; (c) flipping two highest-score pixels $x_a \in \mathbf{x}_{C'}^{(0)}$ and $x_b \in \mathbf{x}_{C'}^{(1)}$, which are in the same neighborhood but with different colors.

as follows.

Starting from pixels with the highest flippability score s_{max}, we identify connected components of them using an 8-connected neighborhood, and process each connected component adaptively. For the i^{th} connected component $T_{i,s_{max}}$, we first identify pixels that are originally black and white, $T_{i,s_{max}}^{(0)} = \{x_j : x_j = 0 \wedge x_j \in T_{i,s_{max}}\}$ and $T_{i,s_{max}}^{(1)} = \{x_j : x_j = 1 \wedge x_j \in T_{i,s_{max}}\}$, respectively. Then, we determine the final flippability score for pixels in the i^{th} connected component according to the size of the two sets $n_{i,s_{max}}^{(0)} = \left|T_{i,s_{max}}^{(0)}\right|$ and $n_{i,s_{max}}^{(1)} = \left|T_{i,s_{max}}^{(1)}\right|$. If $n_{i,s_{max}}^{(0)} > n_{i,s_{max}}^{(1)}$, we keep the flippability score of s_{max} unchanged for those pixels in $T_{i,s_{max}}^{(0)}$, but assign a small score of s_d (such as 0.1 in [90]) to those in $T_{i,s_{max}}^{(1)}$. Otherwise, a small score of s_d is assigned to pixels in $T_{i,s_{max}}^{(0)}$, while the score of s_{max} remains for those in $T_{i,s_{max}}^{(1)}$. In this way, we ensure that final flippables with a score of s_{max} in the same neighborhood are in the same color while the number of them is maximized. An example of such adaptive trimming for highest-score flippables is given in Fig. 4.3.



(a) (b)

Figure 4.3: An example of adaptive trimming for highest-score flippables: (a) highest-score flippables before adaptive trimming; (b) highest-score flippables after adaptive trimming.

The adaptive trimming can be further carried out for a smaller flippability score. Assume that we have identified pixels with a flippability score larger than this current score s. As shown in Fig. 4.4(a), we first combine these identified higher-scoring flippables with those of the current score s, and then find connected components from this pixel combination. For each of the connected components, we perform the following processing. Let $T_{i,\geq s}$ be the i^{th} connected component, in which $T_{i,s}$ and $T_{i,>s}$ contain its pixels of a score equal to s and larger than s, respectively. We then identify black and white pixels in $T_{i,s}$ and $T_{i,>s}$, respectively:

$$\begin{cases} T_{i,s}^{(0)} = \{x_j : x_j = 0 \wedge x_j \in T_{i,s}\} \\ T_{i,s}^{(1)} = \{x_j : x_j = 1 \wedge x_j \in T_{i,s}\} \end{cases}, \begin{cases} T_{i,>s}^{(0)} = \{x_j : x_j = 0 \wedge x_j \in T_{i,>s}\} \\ T_{i,>s}^{(1)} = \{x_j : x_j = 1 \wedge x_j \in T_{i,>s}\} \end{cases}. \quad (4.4)$$

After that, we adaptively assign flippability scores to pixels in $T_{i,s}$ according to the size of the two sets $T_{i,>s}^{(0)}$ and $T_{i,>s}^{(1)}$. Let $n_{i,>s}^{(0)} = \left|T_{i,>s}^{(0)}\right|$ and $n_{i,>s}^{(1)} = \left|T_{i,>s}^{(1)}\right|$. If $n_{i,>s}^{(0)} > 0$ and $n_{i,>s}^{(1)} = 0$, it indicates that we have assigned high flippability scores for pixels in $T_{i,>s}^{(0)}$, and therefore the score of t should remain for pixels in $T_{i,s}^{(0)}$ while a small score of s_d is assigned to pixels in $T_{i,s}^{(1)}$. Similarly, if $n_{i,>s}^{(1)} > 0$ and $n_{i,>s}^{(0)} = 0$, the score of s remains for pixels in $T_{i,s}^{(1)}$, and a small score of s_d is

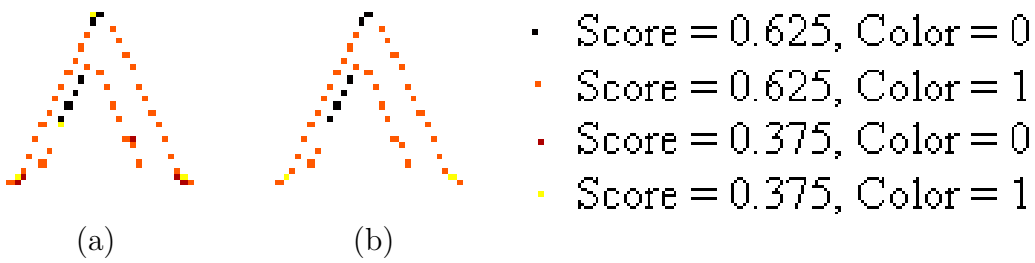


Figure 4.4: An example of adaptive trimming for non-highest-score flippables: (a) combine flippables with score 0.625 after adaptive trimming with those with score 0.375 before adaptive trimming; (b) flippables after adaptive trimming and with score larger than or equal to 0.375.

assigned to those in $T_{i,s}^{(0)}$. If $n_{i,>s}^{(1)} = 0$ and $n_{i,>s}^{(0)} = 0$, it indicates that all pixels in the i^{th} connected component have a score of t, and we assign flippability scores as above for the s_{max} case. Finally, if $n_{i,>s}^{(1)} > 0$ and $n_{i,>s}^{(0)} > 0$, it indicates that one or more pixels in $T_{i,s}$ have connected pixels with a flippability score higher than s but in different colors as a single connected component. To avoid the foreground-background irregularity as discussed before, we assign a small score of s_d to all pixels in $T_{i,s}$ in this case. With such adaptive trimming, we maximize the number of flippable pixels with a higher score under the perceptual constraint of requiring any two flippables in the same neighborhood be in the same color. An example of flippables after the above adaptive trimming and with a score larger than 0.375 is shown in Fig. 4.4(b). Additional trimming can be applied to the identified high-scoring flippable pixels, such as using the rule that any two of them cannot be in a 4-connected neighborhood.

4.3.2 "Super-Pixels" for Group Flippability

When evaluating pixel flippability in binary images, the prior art generally focuses on the flipping of individual pixels. However, in some types of binary images, such as text documents, a group of pixels that may not be individually flippable can be changed together without introducing visible artifacts, as shown in Fig. 4.5. After being grouped as a single unit, a set of individually non-flippable pixels may take a few number of patterns satisfying the perceptual quality requirement. This provides the embedder more resources to manipulate for carrying the hidden data. In our work, we refer to such a pixel set as a *super-pixel* [40], and propose to incorporate it in the framework of wet paper coding to achieve a better tradeoff between the embedding payload and the perceptual quality of watermarked binary images.

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

Figure 4.5: Super-pixel example: (a) original text image; (b) zoomed-in view for the lower part of the leftmost stroke in (a); (c) flipping an individually non-flippable pixel below the bottom horizontal line in (b); (d) flipping a set of individually non-flippables below the same horizontal line as in (c).

Problem Formulation: Given a cover binary image $\mathbf{x}$ consisting of n elements $\{x_i\}, i = 1, \cdots, n, x_i \in \{0, 1\}$, in addition to the k individually flippable pixels $\{x_j\}, j \in C$, we now have u super-pixels coming from those individually non-

flippable pixels $\{x_j\}, j \notin C$. In order to utilize the super-pixel resources to further increase the embedding payload, the embedder needs to determine values of the k individually flippable pixels as well as identify one of the allowed patterns for each of the u super-pixels so that the marked binary image $\mathbf{y}$ still satisfies $D\mathbf{y} = \mathbf{m}$. On the detector side, since blind detection is required, the hidden data $\mathbf{m}$ shall still be extracted via the simple matrix multiplication $\mathbf{m} = D\mathbf{y}$, without using the original image and any information about both individually flippable pixels and super-pixels.

Representing Super-Pixels as Regular Flippables: From the above problem formulation, we can see that the main issue remains at the embedder side, which needs to determine one of several patterns for each super-pixel. We solve this problem by representing super-pixels as regular individual flippables.

Assume that the i^{th} super-pixel consists of $p_i \geq 2$ black-white pixels and allows $s_i = 2^{t_i}$ patterns satisfying the fidelity requirement, where $t_i < p_i$. To represent the s_i patterns, we now can use t_i bits, each of which can take either 0 or 1, just like a regular flippable pixel. In this way, through the introduction of the i^{th} super-pixel, we transform p_i individually non-flippable pixels to t_i regular individual flippables. With all the u super-pixels, we can increase the total number of flippables from k to $k + \sum_{i=1}^{u} t_i$.

Under the framework of wet paper coding, the pseudo-random matrix D is the only information shared between the embedder and the detector, and the detector shall be able to extract the q-bit hidden data $\mathbf{m}$ from the test image $\mathbf{y}$ via a simple matrix multiplication $\mathbf{m} = D\mathbf{y}$. However, when incorporating super-pixels in the WPC-based embedding, the introduction of the i^{th} super-pixel shall reduce p_i individually non-flippable pixels to t_i flippables. Correspondingly, the p_i columns in the

pseudo-random matrix D for the p_i non-flippables shall be reduced to t_i columns for the t_i bits representing the i^{th} super-pixel. In order to support blind detection at the detector side, such pixel and matrix reduction should be appropriately executed. Denote the above p_i columns in D as a $q \times p_i$ sub-matrix D_{p_i}, and their reduced t_i columns as a $q \times t_i$ matrix D_{t_i}. Also organize the s_i patterns that the i^{th} super-pixel can take as a $p_i \times s_i$ matrix $P^{(i)} = [\mathbf{p}_1 \ \ \mathbf{p}_2 \ \ ... \ \ \mathbf{p}_{s_i}]$, and their corresponding t_i-bit super-pixel representation as a $t_i \times s_i$ matrix $T^{(i)} = [\mathbf{t}_1 \ \ \mathbf{t}_2 \ \ ... \ \ \mathbf{t}_{s_i}]$. It is required that

$$D_{p_i} P^{(i)} = D_{t_i} T^{(i)}, \tag{4.5}$$

and this should hold true for each of the u super-pixels. After applying the above reduction to all the u super-pixels, the original cover image $\mathbf{x}$ with n pixels becomes a vector $\mathbf{x}^{(r)}$ of $n^{(r)} = n - \sum_{i=1}^{u}(p_i - t_i)$ elements, and the matrix D of size $q \times n$ becomes $D^{(r)}$ of size $q \times n^{(r)}$.

In (4.5), the pseudo-random sub-matrix D_{p_i} and the pattern set $P^{(i)}$ are given, while the reduced matrix columns D_{t_i} and the t_i-bit pattern indices $T^{(i)}$ are to be determined. Since the pattern set $P^{(i)}$ is designed according to perceptual properties of binary images, there may not be a solution to (4.5) in some cases. One special situation is that $T^{(i)}$ can be chosen to satisfy a linear relationship with $P^{(i)}$, i.e., there exists a binary matrix $G^{(i)}$ such that $P^{(i)} = G^{(i)}T^{(i)}$. By plugging $P^{(i)} = G^{(i)}T^{(i)}$ into (4.5), we can easily identify $D_{t_i} = D_{p_i}G^{(i)}$. A simple example of this linear case is a super-pixel allowing two patterns of $\mathbf{p}_1 = [0, 0, \cdots, 0]$ and $\mathbf{p}_2 = [1, 1, \cdots, 1]$. The 1-bit indices of $\mathbf{p}_1$ and $\mathbf{p}_2$ can be determined as $\mathbf{t}_1 = 0$ and $\mathbf{t}_2 = 1$, respectively, while the $G^{(i)}$ matrix is an all 1 vector.

WPC-based Embedding with Super-Pixels: As shown in Fig. 4.6, there are three main steps when implementing WPC-based embedding with super-pixels. In

Step-1, by representing super-pixels as regular flippables, we reduce the original binary image $\mathbf{x}$ to $\mathbf{x}^{(r)}$, and the pseudo-random matrix D to $D^{(r)}$, as discussed above in Section 4.3.2. In Step-2, a set of equations are established over the reduced $\mathbf{x}^{(r)}$ and $D^{(r)}$ and solved to obtain a reduced marked vector $\mathbf{y}^{(r)}$ satisfying $D^{(r)}\mathbf{y}^{(r)} = \mathbf{m}$, using wet paper coding as discussed in Section 4.2. In Step-3, from the reduced marked vector $\mathbf{y}^{(r)}$ carrying the hidden data $\mathbf{m}$, we reconstruct the marked binary image $\mathbf{y}$ of the same size as the original image $\mathbf{x}$. In the following, we present some implementation details of Step-1 reduction and Step-3 reconstruction. To facilitate these two operations, we establish a locating vector to record locations of individual black-white pixels that are contained in each of the u super-pixels.

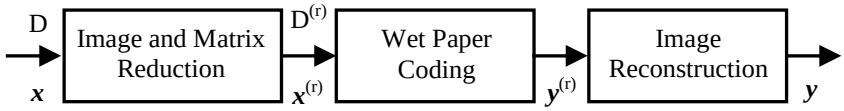


Figure 4.6: Diagram of WPC-based embedding with super-pixels.

Step-1 performs image and matrix reduction as follows. For the i^{th} super-pixel consisting of p_i individual pixels and allowing 2^{t_i} patterns, its t_i-bit index as in the original image $\mathbf{x}$ can be identified via looking up the index-pattern mapping $T^{(i)}$-$P^{(i)}$. Aided with the locating vector, we put this t_i-bit index at the first t_i locations of the super-pixel, and mark the remaining $p_i - t_i$ locations as "invalid". To perform matrix reduction, according to the locating vector, we first organize the p_i columns in D to obtain D_{p_i}. Then, as discussed in Section 4.3.2, we reduce D_{p_i} to D_{t_i} of t_i columns. After that, we replace the first t_i columns of D_{p_i} with D_{t_i}, and mark the remaining $p_i - t_i$ columns as "invalid". After the above processing is finished for all the u super-pixels, "invalid" pixels in $\mathbf{x}$ and "invalid" columns in D are repudiated to obtain $\mathbf{x}^{(r)}$ and $D^{(r)}$, respectively. The reduced image vector

$\mathbf{x}^{(r)}$ has $n - \sum_{i=1}^{u}(p_i - t_i)$ pixels, among which $k + \sum_{i=1}^{u} t_i$ are flippables, ready for the subsequent WPC-based embedding.

In Step-3, we have a reduced $\mathbf{y}^{(r)}$ of $n^{(r)} = n - \sum_{i=1}^{u}(p_i - t_i)$ pixels and satisfying $D^{(r)}\mathbf{y}^{(r)} = \mathbf{m}$, and we need to reconstruct a marked image $\mathbf{y}$ with n pixels and satisfying $D\mathbf{y} = \mathbf{m}$. Among the $n^{(r)}$ pixels in $\mathbf{y}^{(r)}$, WPC embedding may have flipped some of the $k + \sum_{i=1}^{u} t_i$ flippables. Comparing $\mathbf{y}^{(r)}$ and the reduced cover image $\mathbf{x}^{(r)}$, we can easily identify the actual flippings, each of which originally may be an individually flippable pixel or one element of a super-pixel index. For the former, we identify the pixel location j in the original image $\mathbf{x}$ and then flip it to obtain its value in the marked image $\mathbf{y}$, i.e., $\mathbf{y}(j) = 1 - \mathbf{x}(j)$. For the latter, aided with the locating vector, we first retrieve the whole index with t_i bits. Then, we find its corresponding p_i-pixel group via the established index-pattern mapping, and place the pixel group at its corresponding locations in $\mathbf{y}$. Such reconstruction is performed for all actual flippings, while the untouched pixels in $\mathbf{y}$ will be the same as their counterparts in $\mathbf{x}$.

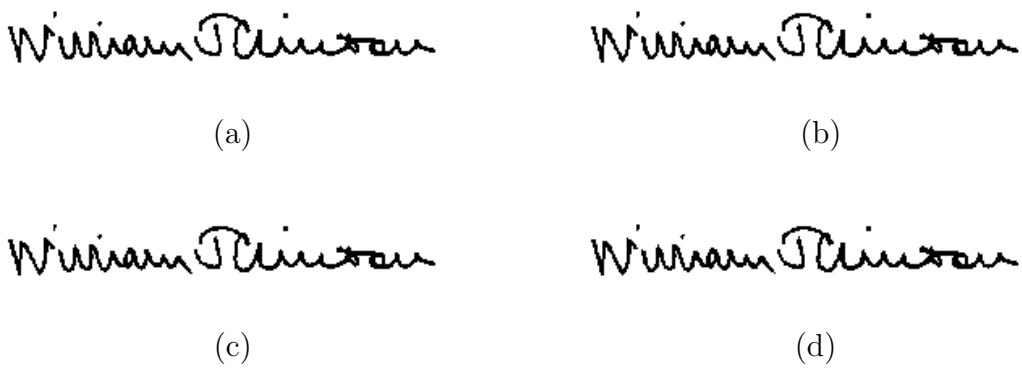
4.4 Experimental Results

In this section, we provide experimental results on employing wet paper codes to hide data in binary images. First, we demonstrate the embedding payload and perceptual quality of marked binary images, with flippable pixels identified using the proposed adaptive trimming method in Section 4.3.1. Then, we examine the performance of incorporating super-pixels in the framework of wet paper coding to further increase the embedding payload while preserving the perceptual quality of watermarked binary images.

4.4.1 Results with Adaptive Trimming

In the first test, the original binary image in Figure 4.7(a) consists of 48×288 pixels. Using adaptive trimming as discussed in Section 4.3.1, we identify 271 pixels having the highest flippability score 0.625, and 447 pixels having a score greater than 0.1. For comparison, the 3×3 window trimming in [90] returns 250 pixels of score 0.625 and 371 pixels with a score greater than 0.1. Figures 4.7(b)-(d) show the watermarked images after embedding $q = 54$, 150, and 260 bits, respectively, using wet paper coding. In the embedding, $k = 271$ flippable pixels are selected as all pixels with the highest flippability score 0.625. Visually comparing the watermarked images in Figures 4.7(b)-(d) with the original image in Figure 4.7(a), we can see that the embedding of 54 and 150 bits does not introduce detectable artifacts, while a few artifacts can be found through very careful observation in the case of embedding 260 bits. Compared with the shuffling approach that embeds one bit per 16×16 block for a total of 54 bits into this test image, the wet paper coding allows a five-fold improvement in the embedding payload. Using the wet paper coding, we can also easily accommodate different sizes of hidden data in different applications.

In the second test, we use a binary map image as shown in Figure 4.8(a), which consists of 321×321 pixels. After the adaptive trimming, this image returns $k = 2678$ flippable pixels with a flippability score strictly above 0.1, among which 1539 pixels have the highest flippability score 0.625. For comparison, the 3×3 window trimming provides 1514 pixels of score 0.625 and 2116 pixels of score above 0.1. With the flippables identified via adaptive trimming, the WPC approach was used to embed 1530 bits, and a marked map image is shown in Figure 4.8(b). Still the embedding selects all pixels with the highest flippability score 0.625 as



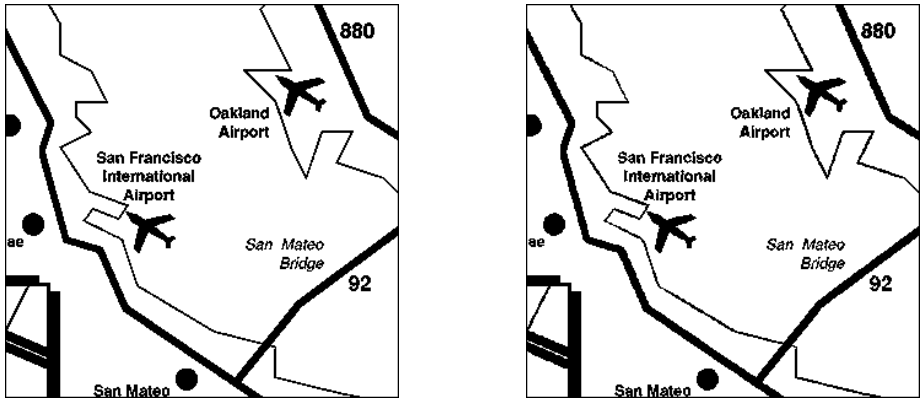
(a) (b)

(c) (d)

Figure 4.7: Results on applying WPC embedding to the Clinton's Signature image: (a) original Clinton's signature, (b) 54 bits embedded, (c) 150 bits embedded, (d) 260 bits embedded.

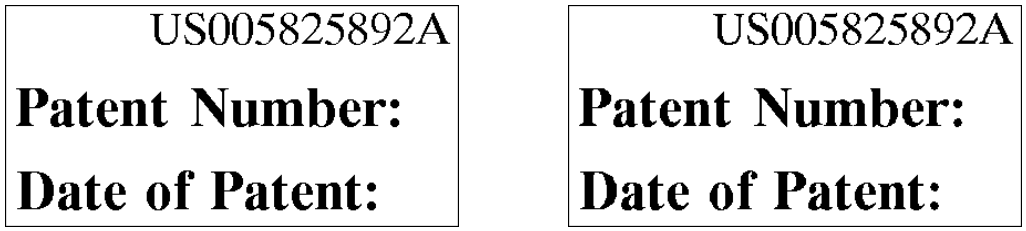
the $k = 1539$ flippables. As a comparison, the embedding payload for the shuffling approach is around 400 bits, which again suggests an almost five-fold improvement in the embedding payload by the WPC based embedding.

In the last test, we crop a 200×410 area from a binary scanned text document as the cover image, which is shown in Fig. 4.9(a). Using adaptive trimming, we arrive at 892 pixels with score 0.625, 309 pixels with score 0.375, 35 pixel with score 0.25, and 290 pixels with score 0.125. As a comparison, the 3×3 window trimming identifies 792, 174, 35, and 196 pixels with score 0.625, 0.375, 0.25, and 0.125 respectively. For this example, adaptive trimming increases the number of pixels with score 0.625 by 12.6%, and we select all of them as the $k = 892$ flippable pixels in the embedding. In Figure 4.9(b), 880 bits are embedded using flippables identified via adaptive trimming. Visually comparing Figures 4.9(a) and (b), we can see that the watermarked image has good perceptual quality while hosting a large embedding payload. Comparing with the shuffling approach which can embed 320 bits when 16×16 blocks are used, we achieve an almost three-fold



(a) (b)

Figure 4.8: The original and embedded binary map images: (a) original image; (b) embedded with 1530 bits.



(a) (b)

Figure 4.9: The original and embedded text document images: (a) original image; (b) embedded with 880 bits.

improvement on the embedding payload when using the WPC based approach.

4.4.2 Results with "Super-Pixels"

Using the binary text document as shown in Fig. 4.9(a) as an example, we evaluate the embedding payload and the perceptual quality of watermarked binary images with super-pixels identified and incorporated under the framework of wet paper coding. As presented in Section 4.4.1, the adaptive trimming provides us 892 pixels with the highest flippability score 0.625 for this cover image, and we select all of them as the $k = 892$ individually flippable pixels in the embedding. In Fig. 4.10(a) and (b), we re-draw the cover image and show the 892 individually flippable pixels, respectively. From Fig. 4.10(b), we can see that these individually flippable pixels are located on the non-straight boundaries of the characters.

In Fig. 4.5 of Section 4.3.2, we have shown that a set of individually non-flippable pixels along a straight boundary can be flipped together without affecting the perceptual quality of text documents. Hence, as shown in Fig. 4.10(c), we select 53 straight boundaries as our 53 super-pixels, after appropriate trimming to separate each super-pixel from individually flippable pixels as well as other super-pixels. Each of these 53 super-pixels can take two patterns, either all 1's or all 0's. As discussed in Section 4.3.2, with such a pattern design, we can successfully implement the image and matrix reduction before the wet paper coding. Incorporating these 53 super-pixels in the WPC-based embedding in addition to the 892 individually flippable pixels, we can further improve the embedding payload. As each of the $u = 53$ super-pixels takes two patterns that can be indexed by 1 bit, the total number of effective flippables is increased from $k = 892$ to $k + u \times 1 = 892 + 53 = 945$, so is the expected maximum embedding payload. This

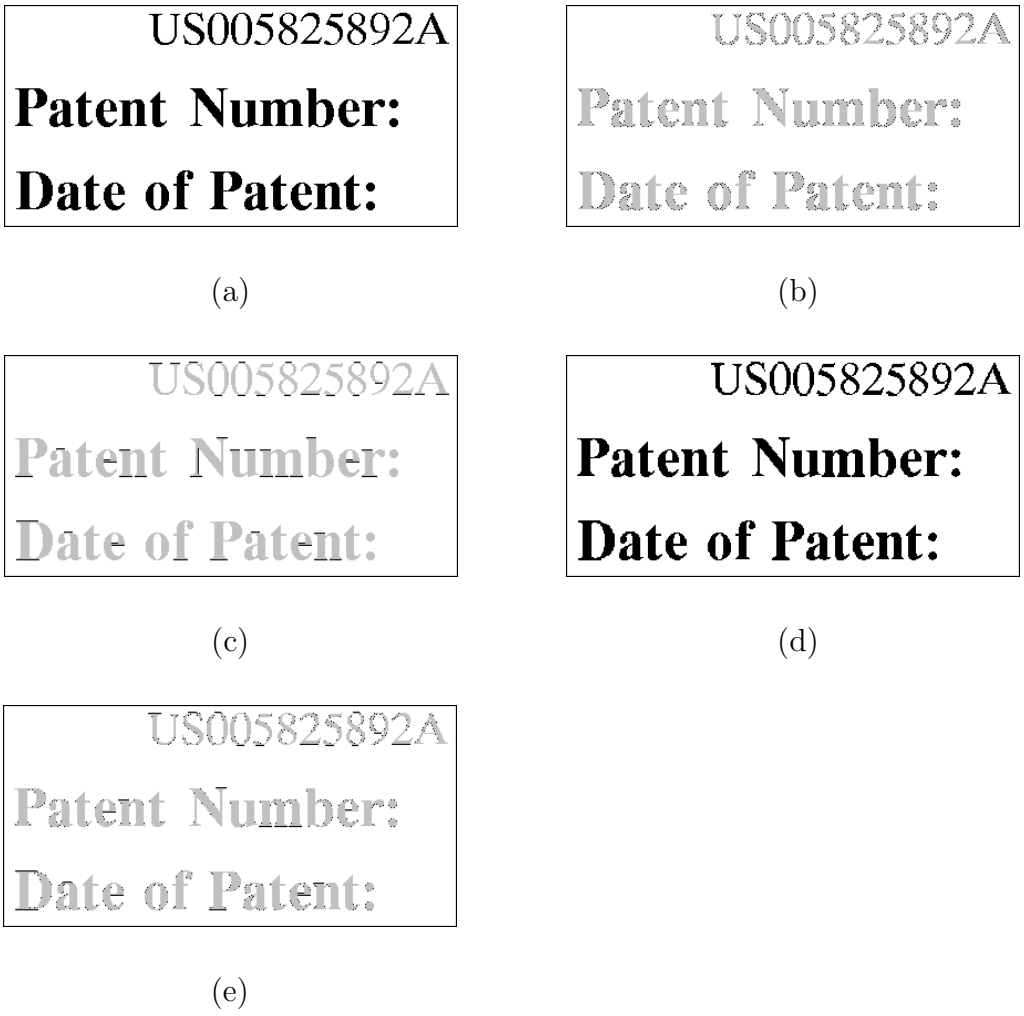
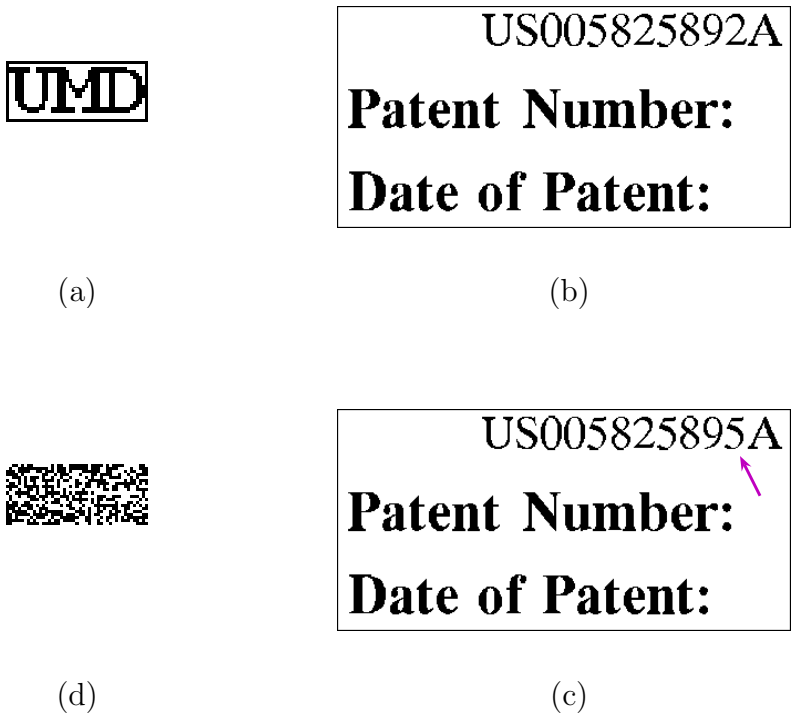


Figure 4.10: Experimental results with super-pixels: (a) original image; (b) individually flippable pixels shown as dark points; (c) straight boundaries as super-pixels shown as dark lines; (d) marked image; (e) pixel-wise difference between (a) and (d) shown as dark points/lines.

is an additional increase of 5.9%.

In Fig. 4.10(d), we show a watermarked text document for the cover image in Fig. 4.10(a), with $q = 940$ bits embedded using both the 892 individually flippable pixels and the 53 super-pixels. Visually comparing these two images, we can see that the embedding does not introduce annoying artifacts. The pixel-wise difference between the original image and the marked image is shown in Fig. 4.10(e). Given the marked image in Fig. 4.10(d) and the shared pseudo-random matrix D, the 942-bit hidden data can be accurately extracted via matrix multiplication, without any information about either individual flippables or super-pixels.



(a) (b)

(d) (c)

Figure 4.11: Application to authenticating binary images: (a) a logo image as the watermark; (b) a watermarked binary image with the logo embedded in; (c) a tampered version of (b) with one digit changed as indicated by the arrow; (d) the watermark extracted from the tampered binary image in (c).

To demonstrate the application of WPC based embedding to authenticating binary images, we take a 47×20 binary logo image as shown in Fig. 4.11(a) as the watermark. We then embed it into the original image as shown in Fig. 4.10(a), and obtain a watermarked binary image as shown in Fig. 4.11(b). In the embedding, we use both the 892 individually flippable pixels and the 53 super-pixels as the $k = 945$ flippables. From this marked image in Fig. 4.11(b), the logo image can be accurately extracted when the correct pseudo-random matrix D is available. Acting as an attacker, we change one digit from "2" to "5" on the top row of the marked text image, as indicated in Fig. 4.11(c). From this tampered binary image and assuming the correct pseudo-random matrix D, we perform watermark extraction again, but the matrix multiplication no longer returns a meaningful logo image, but a meaningless random image as shown in Fig. 4.11(d). Because integrity of the logo image has been violated, we can claim with high confidence that the binary text image in Fig. 4.11(c) is a tampered image, although it is perceptually good to our eyes.

4.5 Discussions on Watermark's Robustness

In content-based authentication applications that distinguish malicious content tampering from content-preserving processing, some level of resilience against noise is desirable. In this section, we discuss the robustness of WPC-based data embedding in binary images against minor processing and distortions, and also compare it with the shuffling approach.

4.5.1 Analytic Results

In order to obtain analytic insights on the robustness of WPC-based watermarks on binary images, we consider a simplified distortion model, where some pixels of a marked image are randomly flipped due to noise. Reliable extraction of hidden data under such distortion is possible if we apply appropriate error correcting coding (ECC) to the data payload before embedding into the host image. In the following, we first examine the raw error probability before error correcting, as it provides a guideline on selecting what ECC to use. We then analyze the achievable payload that can be reliably conveyed using the WPC-based embedding.

Raw Error Probability: To model the pseudo-random binary matrix D in the WPC approach, we assume its elements to be i.i.d. realizations of a random variable that is 1 with probability δ and 0 with probability $1-\delta$, where $\delta = log_2(k)/k$ is the density of D matrix as discussed before. When m pixels in the marked image are flipped due to noise, we analyze how many message bits will be incorrectly extracted. We note that the i^{th} message bit is extracted as XOR of pixels whose indices are determined by 1's in the i^{th} row of D. Thus, it is extracted as incorrect if and only if there are an odd number of noise-flipped pixels among them. The probability that the i^{th} bit will be incorrect is thus

$$P_{WPC}(\delta) = \sum_{k=0}^{\lfloor (m-1)/2 \rfloor} \left(\begin{array}{c} m \\ 2k+1 \end{array} \right) \delta^{2k+1}(1-\delta)^{m-(2k+1)} = \frac{1-(1-2\delta)^m}{2}. \quad (4.6)$$

We can see that for the same number of noise-flipped pixels (as quantized by m), a smaller density δ will have a smaller probability of error. For small δ, we can also approximate $P(\delta)$ to be $m\delta$.

For comparison, we also analyze the error probability of the shuffling method. Using analysis similar to those in [90], we can calculate the mean value of the

percentage of blocks each having exactly r noise-flipped pixels

$$E\left[\frac{\mu_r}{q}\right] = \frac{\begin{pmatrix} B \\ r \end{pmatrix}\begin{pmatrix} n-B \\ m-r \end{pmatrix}}{\begin{pmatrix} n \\ m \end{pmatrix}}, \tag{4.7}$$

where μ_r is the number of blocks each having exactly r noise-flipped pixels, $q = n/B$ is the number of blocks (also the message size) with B being the block size and n being the total number of pixels, and m is the number of noise-flipped pixels. Since each odd-valued r will bring 1-bit message error, the message bit error rate for the shuffling method is

$$P_{shuffle} = \sum_{r_{odd}} E\left[\frac{\mu_r}{q}\right]. \tag{4.8}$$

As the block-based embedding with the shuffling approach is a special case of wet paper encoding as discussed earlier, we can also derive its bit error rate using (4.6) with the density $\delta = B/n$. Because B/n is generally smaller than $log_2(k)/k$, which is the density of the random matrix D in the WPC approach, it is expected that the shuffling approach has a smaller bit error rate than the general WPC approach.

Achievable Payload: Since WPC-based embedding allows many more bits to be embedded, we can use a stronger ECC code to correct bit errors. As such, it is possible that the actual payload to be reliably conveyed by WPC becomes higher than the shuffling based approach. An upper bound on the achievable payload can be established through modeling the erroneous channel as a binary symmetric channel (BSC) and examining its Shannon channel capacity. According to information theory [18], a BSC channel with cross-over probability p can reliably convey up to $1-h(p)$ bit per channel use, where $h(p) = -p\log_2(p)-(1-p)\log_2(1-$

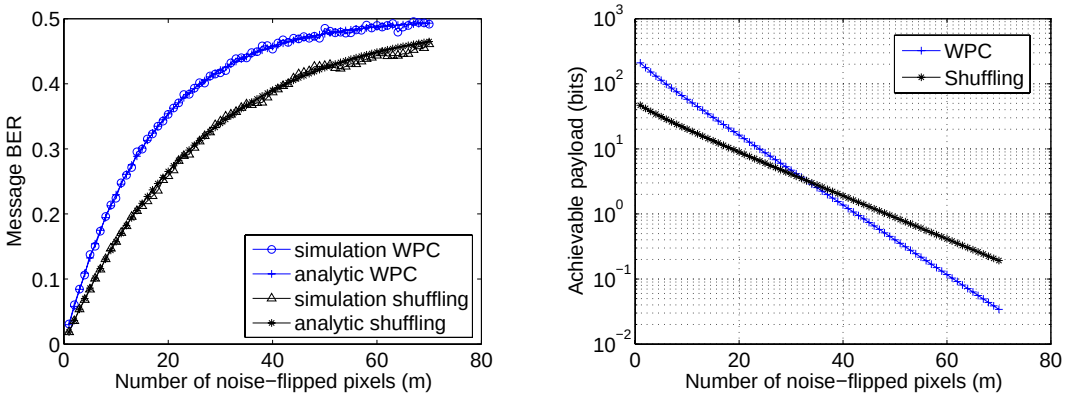
p). Then, the actual payload can be calculated through multiplying $1 - h(p)$ by the number of channels available. In our problem, the cross-over error probabilities for the WPC and shuffling approaches are given in (4.6) and (4.8), respectively.

4.5.2 Simulation Results

Using the Clinton's signature as an example, we first compare the raw error probability of the WPC and shuffling approaches. For the WPC approach, the matrix density is $\delta = \log_2(k)/k$, where $k = 271$, and a total of 260 bits are embedded; for the shuffling approach, the block size is $B = 256$, the number of blocks is $q = 54$, and the equivalent matrix density is $1/54$. The analytic results of the message bit error rate after randomly flipping m pixels are shown in Figure 4.12(a), along with the simulation results from 100 random realizations. We can see that the analysis and simulation agree very well. At the settings given here, the shuffling based approach appears to have a smaller bit error rate than the WPC. It is true for both approaches that the lower the density of D, the lower the error rate. However, the low density may adversely affects the solvability. For the shuffling approach, the low matrix density may violate the requirement of having at least one flippable per block. For the WPC approach, when the matrix density decreases below $\log_2(k)/k$, the probability of finding a solution to (4.2) quickly decreases.

After obtaining the raw error probabilities, we further calculate the achievable payload for the signature image using the shuffling and the WPC approaches. In our case, the number of channels is the number of bits embedded in the binary image, which is 54 for the shuffling approach and 260 for the WPC embedding, respectively. As we can see from Figure 4.12(b), the proposed data hiding approach for binary images based on WPC is capable of reliably conveying more bits than

the shuffling based approach when the amount of noise/distortion is moderate.



(a) (b)

Figure 4.12: Robustness comparison of the shuffling and WPC based approach: (a) raw error probability; (b) achievable payload.

4.6 Chapter Summary

In this chapter, we have investigated the incorporation of a recent steganography framework, wet paper coding, for high-payload and high-fidelity data embedding in binary images. The wet paper codes naturally handle the uneven embedding capacity through randomized projections. As opposed to the previous shuffling-based approach, where only a small portion of the flippable pixels are actually utilized in the embedding, the wet paper codes allow for a high utilization of pixels that have high flippability score for embedding, thus giving a significantly improved embedding payload than the previous approach. In order to preserve the perceptual quality of watermarked binary images under the high embedding payload, we have proposed an adaptive trimming method to perform flippability assignment on individual pixels in a binary image, and also developed a new con-

cept of super-pixels to address the dependencies among the flippability of a group of pixels. We have demonstrated on several representative images that the proposed technique provides a large enhancement on the embedding payload while achieving high perceptual quality for watermarked images. We also analyze and compare the watermark's robustness of the new WPC-based approach with the shuffling approach, and reveal the tradeoff between the embedding payload, problem solvability, and watermark robustness.

Chapter 5

Intrinsic Sensor Features for Scanner Forensics

While cameras provide digital reproduction of natural scenes, scanners are often used to capture hard-copy art in more controlled environment and have become a major kind of apparatus to transform graphic data into the digital domain. In this chapter, we propose a new technique of utilizing intrinsic sensor noise features for non-intrusive scanner forensics to verify the source and integrity of digital scanned images. Using only scanned image samples, we analyze scanning noise from several aspects, including through image de-noising, wavelet analysis, and neighborhood prediction, and then obtain statistical features from each characterization. Based on the proposed statistical features of scanning noise, we construct a robust scanner identifier to determine the model of the scanner used to capture a scanned image. Utilizing these noise features, we further broaden the scope of acquisition forensics to differentiating scanned images from camera taken images and computer generated images. The proposed noise features also enable integrity forensic analysis on scanned images, including detecting post-processing and stegonagraphy opera-

tions on scanned images. Through experimental results, we demonstrate that the proposed method can identify the correct scanner model with high accuracy and with strong robustness against a number of post-processing operations on scanned images. We also demonstrate the effectiveness of employing the proposed noise features for performing various forensic analysis on scanned images.

5.1 Introduction

Scanners are a major kind of apparatus to transfer graphic data on paper hard-copies to the digital domain. Such a transfer is of utmost importance in the modern information era, as it provides an essential means to facilitate maintenance and sharing of various graphic data. *Electronic filing cabinets* have been introduced in many companies and commercial establishments as a modern way to handle otherwise paper documents[1]. Banks and financial agencies are moving towards paperless solutions in their everyday operations. One recent advancement is the legislation of the Check Clearing for the 21^{st} Century Act (*Check 21*) [1], which was adopted as part of the *e-banking* initiative in 2004. This legislation allows banks to exchange scanned check images rather than physical checks, and use the scanned version as the record to authorize electronic fund transfer from one account to another[2]. As scanned check images can be electronically transmitted across the world in minutes, Check 21 can significantly reduce check processing costs as well as increase business efficiency. The benefits of fast transmission and convenient processing of scanned check images also give rise to new kinds of attacks. Analogous to forgeries of traditional paper checks, adversaries can now employ digital

[1]http://businessweek.buyerzone.com/computers/scanners/buyers_guide2.html

[2]http://www.eweek.com/category2/0,1738,1684884,00.asp

domain processing techniques to alter check images, such as changing the payee and the monetary amount. Therefore, it has become particularly important in these cases to ensure the images after scanning are not further tampered.

Traditional methods for image authentication include digital signature based on public-key encryption, robust image hashing [79], and fragile digital watermarking [26]. In digital signature based authentication, a key-based hash is computed from the digital image and then transmitted along with it. If tampering has happened to the digital image, it can be detected by verifying if the attached digital signature matches the one generated from the test image [94]. The change of even a single bit (e.g., due to transmission errors) may return a totally different digital signature. Content based robust image hashing schemes [79] generate image hashes resilient to content-preserving operations, which addresses the fragility issue of conventional hashing and can be used as robust digital signatures for image authentication. In watermarking based verification techniques as discussed in Chapter 4, pre-defined patterns or content features are seamlessly embedded into the digital image through slightly modifying its pixel values. The above encryption/watermarking module needs to be implemented inside the image capturing device. This imposes restrictions on its usage in authentication of scanned images, as many scanners in the market do not have such capabilities. Complementary to these existing techniques, there is a strong need to devise non-intrusive forensic methods for verifying the source and integrity of digital scanned images by using only scanned image samples.

While a growing amount of signal processing research is devoted to the security and protection of multimedia data, forensic research on image acquisition devices is still in its infancy. To our best knowledge, this is one of the first efforts to

perform comprehensive source and integrity forensic analysis on scanned images. The related prior works fall into two categories. In the first category, there are a few works on non-intrusive forensic analysis of digital cameras, and some of them have been employed to perform tampering detection on images captured by digital cameras. Component forensics was introduced as a methodology for forensic analysis [81], where the authors have shown that the parameters obtained from camera modelling, such as the color filter array (CFA) pattern and color interpolation coefficients, can be used to construct a camera identifier to determine the brand/model of the camera used to capture a photographic image. Based on such a camera identifier, the work in [80] detects multi-segment slicing tampering by assuming that different segments in a doctored image come from different cameras. In [73], specific correlations introduced by CFA interpolation are quantified, and an image region is identified as a tampered region if it lacks this specific correlation. Being an essential algorithm inside most digital cameras, the CFA interpolation is also utilized in [8] for classifying three different camera models. As we shall see in Section 5.2, scanners capture color information in a different way without using the CFA interpolation. Therefore, the above techniques for identifying digital cameras cannot be directly extended to identifying scanners. In [60–62], pattern noise associated with camera sensors is employed as a unique identifier of each camera, and further for detecting image forgeries, in which a tampered region may be copied from another region in the same image, or in a different image taken by either other cameras or the same camera at different times. Such pattern noise functions as a spread spectrum watermark unique to individual cameras, and its presence in a digital photograph is determined using a correlation detector. Because the correlation detector is highly sensitive to de-synchronization caused by geometric

distortions, which are inherent in the scanning process, extending this technique to scanner identification needs to address such de-synchronization issue carefully.

In the second category of related prior works, classifier based approaches were proposed to detect image tampering [5,23] and/or to perform blind steganalysis on images [6,24]. Finding suitable features is the key point of these classifier based approaches. In [5,6], features are calculated as a set of suitably chosen image quality metrics evaluated between an input image and its filtered version using a low-pass Gaussian filter. In [23, 24], multiple level wavelet decomposition is applied to the input image, and then one set of features is obtained as the $1^{st} \sim 4^{th}$ order moments of wavelet coefficients in individual subbands. Employing a linear predictor to capture correlations that exist across orientation, space, and scale, an additional set of features is computed from the $1^{st} \sim 4^{th}$ order moments of prediction errors. The above features in prior works may be employed to performing forensic analysis on scanned images. However, our testing with those features shows a very limited performance in identifying sources of scanned images. This is because that the features used in these prior works are extracted without specifically taking the scanning process and properties into consideration.

Concurrently with our work, the pattern noise introduced in [60, 62] for identifying individual cameras has been utilized in [48] to identify individual scanner sets. Certain statistical features of the pattern noise are extracted and then feed into a support vector machine to classify individual scanners. An overall classification accuracy of 96% is reported in [48] for identifying four scanners, among which two are of the same model. The pattern noise features in [48] have also been extended in [49] to differentiate scanned images from camera taken images. However, scanned images samples in [48,49] are generated using the native resolutions

of the scanners, which are barely used in practice due to the long time of image capturing and the huge size of the resulted scanned images.

In this work, we introduce a novel technique to establish the trust-worthiness of scanning devices and scanned images. In our test, we focus on flat-bed scanners which are one of the most popularly used classes of scanners. Based on a detailed study of the imaging process in digital scanners, we extract informative sensor noise features from scanned image samples for verifying their source and integrity. Specifically, we analyze scanning noise from several angles including through image de-noising, wavelet analysis, and neighborhood prediction, and then obtain statistical features from each characterization. Building upon these statistical noise features, we first construct a robust scanner identifier to determine the model of the scanner used to capture individual scanned images, by using only scanned image samples. Such a non-intrusive system would provide *acquisition forensic* information useful for law enforcement and intelligence agencies, and facilitate secure *e-banking* solutions in financial systems. Utilizing the proposed statistical noise features, we further broaden the scope of *acquisition forensics* to different types of image acquisition apparatus by differentiating scanned images from camera taken images and computer generated images. Finally we perform *integrity forensic* analysis on scanned images using the noise features, including detecting post-processing operations after scanning, and conducting steganalysis on scanned images.

The rest of this chapter is organized as follows. In Section 5.2, we introduce the basic scanner model and discuss scanning noise that forms a unique characteristic of individual scanner models. Section 5.3 elaborates the extraction of statistical noise features from scanned images. In Section 5.4, we present experimental results on

scanner model identification as well as the further acquisition and integrity forensic analysis on scanned images, using the features extracted in Section 5.3. Finally, we summarize this chapter in Section 5.5.

5.2 Scanner Model and Scanning Noise

Fig. 5.1 shows the scanning system of a flat-bed scanner [33]. The system consists of hardware devices that include a scanner head and a motion system, and software algorithms that process and combine sensed data into an output image file. The scanner head contains hardware components for generating a signal corresponding to a thin horizontal strip of the hard-copy original, including a lamp, mirrors, optical lens, color filters, color sensors, and electronic devices. The thin strip is illuminated by the lamp, and its reflected light goes through the lens and further through a trilinear color filter array consisting of red, green, and blue (RGB) filters, respectively. The red, green, and blue light exposures are then captured by the corresponding red, green, and blue CCD (charge-coupled device) sensors. As the three exposures captured by the CCD sensors at a given time are for three parallel raster lines with slightly different offsets in the thin horizontal strip, two of them are buffered until all three colors of individual raster lines are captured. Further, the exposure voltages recorded by CCDs for the three colors are amplified, and then converted to digital values through an analog/digital converter. Driven by a motion system, the scanner head moves from the top of the hard-copy document to its bottom, scans it line-by-line, and captures the three color components (RGB) for all horizontal rows into the digital domain. Corresponding to this line-by-line scanning pattern, the resolution of a scanner contains two parameters, namely, a horizontal optical resolution describing the number of CCD sensors in each row

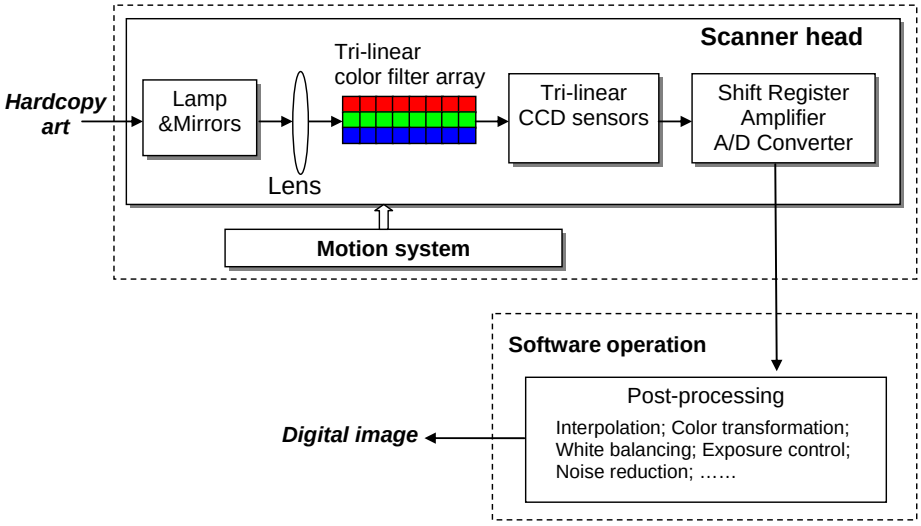


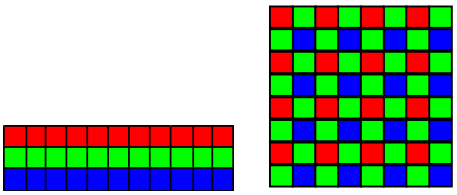
Figure 5.1: The basic model of a scanning system.

of the trilinear structure, and a vertical resolution indicating the step-size of the motion system. Finally, the captured digital RGB values undergo post-processing operations carried out by software algorithms, such as color transformation and white balancing, to create the final scanned image.

As two major devices for acquiring digital images, scanners and digital cameras share a considerable amount of similarities, and also differ in several aspects in imaging process. Both scanners and digital cameras use color filters and CCD sensors as basic elements of capturing color values of the objects to be imaged; they also have similar A/D converters and post-processing operations such as color transformation and white balancing. Several main differences between scanners and digital cameras are highlighted in Table 5.1. The most notable one is the different geometries in which the CCD sensors are organized. Scanners use trilinear CCDs corresponding to red, green, and blue components (as shown in Fig. 5.2(a)), while digital cameras use a 2-D periodic CCD array, such as the well-known Bayer CFA pattern shown in Fig. 5.2(b). Using the trilinear CCD array along with the

Table 5.1: Comparison between scanners and digital cameras.

	Scanner	Camera
Data source	Hard-copy art	Physical scene/object
Light source	High-intensity artificial light	Natural or scene light augmented with flash
Acquisition geometry	One raster line at a time	Capture entire 2-D scene in one shot
Color acquisition	Trilinear CCD to capture numerous full-colored pixels per line, with hundreds or thousands of lines per image	Square CCD array to capture numerous single-colored pixels per image according to CFA pattern (then interpolate to obtain full-colored pixels)



(a) (b)

Figure 5.2: CCD sensor patterns in scanners and digital cameras: (a) Trilinear CCD pattern in scanners, (b) Bayer color filter array pattern used in digital cameras.

line-by-line acquisition mode enabled by the motion system, scanners can directly capture all the *three* color components of each raster line. On the contrary, digital cameras use a square CCD array to capture the entire 2-D scene in one shot. Because each CCD sensor can measure only the value of one color component, color interpolation is required to obtain the other two color components. Serving as a critical step in the imaging process of digital cameras, color interpolation coefficients have been used in the literature [8, 81] to classify different camera models. However, as described above, color interpolation is not used in the scanning and thus would not be a good feature in classifying different scanner models.

To identify features suitable for this new task, we resort to one inherent phe-

nomenon of scanning: the noise effect. During the scanning process, light reflected from the hard-copy original is recorded by CCD sensors, and then converted to digital values. Quantitatively, the amount of reflected light is proportional to the number of photoelectrons captured by a CCD sensor per time period, which is then converted to an analog CCD voltage. Due to the discrete and random nature of photoelectrons, several kinds of noise occur when photoelectrons are created, including random shot noise, dark signal non-uniformity (DSNU), and photo response non-uniformity (PRNU) [43]. The DSNU represents pixel-to-pixel variations in the CCD voltage when no light is incident on the CCD sensors, while the PRNU models the variations under a fixed-intensity light. When the voltage values stored in CCD sensors are read out, additional reset noise occurs. Voltage amplification further introduces white noise and $1/f$ noise whose power spectrum is a function of the inverse of the frequency f. Possible imperfections in the optical system, such as deficiency or spectral peaks in the light source and light intensity fluctuation, may also bring in noise.

Some types of noise in the scanning process can be mitigated to some degree [33]. For instance, to compensate for DSNU, a short scan is done with the scanner's light off or blocked, and then the dark voltages are recorded. These estimated dark voltages represent DC offsets, and therefor can be added to or subtracted from voltages recoded in future scanning operations. PRNU can be measured using a test scan of a built-in, calibrated target strip, with gain factors obtained through comparing measured voltages with their expected values for the target strip. These gain factors can then be used to compensate for the PRNU to some extent for future scans. Effects of random noise in the scanning process can be reduced by using better mirrors/lens, adopting a brighter light source, and/or

by increasing exposure time. However, due to the inherent random nature of noise, some amount of noise always persists even after noise compensation and reduction, leaving intrinsic information for forensic analysis on scanners and scanned images.

In our work, we characterize the scanning noise from a statistical perspective, and further utilize these statistical features to verify the source and integrity of scanned images. Although there are several sources of scanning noise and different kinds of noise have different properties, eventually they all appear as variations in pixel values of scanned images, reflecting intrinsic information of scanner models. Therefore, we characterize these pixel variations without separating the PRNU and DSNU from other kinds of scanning noise. For a single scanner, when specific values of these pixel variations may be changing from scan to scan due to randomness of the scanning noise and/or geometric distortions in the scanning process, their statistical characteristics are expected to be stable for different scans. For the same scanner model, as the same hardware devices and software solutions are employed for its individual sets, the statistical characteristics of the scanning noise are also expected to be stable for different scanner sets of the same model. On the other hand, considering the tradeoff between production cost and scanner quality, different scanner models may use hardware devices of different quality, and/or employ different methods to mitigate scanning noise, and therefore the statistical characteristics of the scanning noise are expected to be different for different scanner models. Based on the above analysis, we extract statistical features of scanning noise to identify scanner models. Considering that images produced by cameras or computer programs contain noise of different amounts and properties from scanned images, we also expect the statistical noise features would help on differentiating scanned images from camera taken images and computer generated images. If

scanned images directly output from scanners are modified to carry stego information or go through certain post-processing operations, such as low-pass filtering, scaling, and rotation as possibly involved in tampering a scanned image, with a high probability the intrinsic statistical noise features of the original scanned images shall be altered. Therefore, it is also promising to employ these statistical noise features for such integrity forensic analysis as detecting post-processing after scanning and performing steganalysis on scanned images.

5.3 Proposed Statistical Feature Extraction

In this section, we discuss how to extract statistical features of scanning noise that can be used for identifying the scanner model source and verifying the integrity of scanned images. We statistically characterize the scanning noise from the following three aspects [34, 35]. First, we apply different denoising algorithms to scanned images, and then obtain their noise part for feature extraction. Second, we focus on histograms of wavelet coefficients at the finest scale, where the histogram shape can be changed by scanning noise. Finally, we study the relationship between each pixel and its eight neighbors, which also reflects the effect of scanning noise to some extent.

5.3.1 Features from Denoising Algorithms

To extract statistical features of scanning noise, we first utilize image denoising algorithms. As shown in Fig. 5.3, given a scanned image I, denoising operation is applied to obtain its denoised version I_D. The scanning noise n_S at the pixel location (i, j) is then found by pixel-wise subtraction $n_S(i, j) = I(i, j) - I_D(i, j)$.

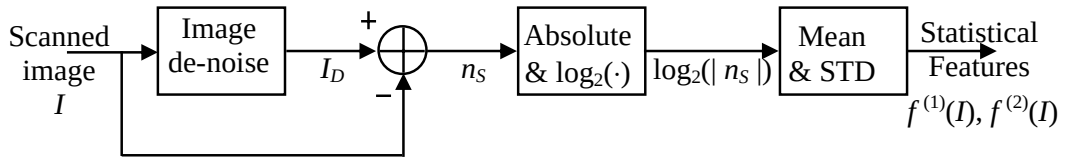


Figure 5.3: Statistical noise feature extraction by using denoising algorithms.

Statistical properties of the estimated noise $\{n_S(i,j)\}$ can then be used as features. In our test, we use the absolute value of scanning noise, take $\log_2$ transformation, and then calculate the mean and the standard deviation of the $\log_2$-transformed absolute values:

$$\begin{aligned} f^{(1)}(I) &= \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\log_2\left(|n_S(i,j)|\right), \\ f^{(2)}(I) &= \left(\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(\log_2\left(|n_S(i,j)|\right)-f^{(1)}(I)\right)^2\right)^{\frac{1}{2}}, \end{aligned} \tag{5.1}$$

where M and N indicate the size of the scanned image.

To capture various kinds of noise in the scanning process, we apply multiple image denoising algorithms to the scanned image. In our work, we employ four denoising algorithms: spatial linear filtering with an averaging filter, spatial linear filtering with a Gaussian filter, spatial median filtering, and Wiener adaptive image denoising algorithm. A low-pass linear filtering using averaging or Gaussian filters helps model the high-frequency noise, a non-linear median filtering addresses the "salt and pepper" noise, and adaptive methods such as Wiener filtering and those in [14, 67] can tailor noise removal to the local pixel variance. Below we present some details of the donoising algorithms used in our work.

(i) Linear filtering with averaging and Gaussian filters: Given a scanned image I and a linear average/Gaussian filter h, the denoised image I_D can be obtained by correlating the filter h with the scanned image I appropriately. The

value of a pixel in the denoised image I_D is computed as a weighted sum of its neighboring pixels and itself in the original scanned image I, with the neighborhood and the weights determined by the filter h

$$I_D(i,j) = \sum_{k=-\lfloor L/2 \rfloor}^{\lfloor L/2 \rfloor} \sum_{l=-\lfloor L/2 \rfloor}^{\lfloor L/2 \rfloor} h(k,l) I(i+k, j+l). \tag{5.2}$$

where the filter size is $L \times L$ with L an odd number. In our test, we use an average filter $h_{average}$ of size 3×3, and a Gaussian low-pass filter $h_{Gaussian}$ of the same size and with a standard deviation $\sigma = 0.5$.

$$h_{average} = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}, \qquad h_{Gaussian} = \begin{bmatrix} 0.0113 & 0.0838 & 0.0113 \\ 0.0838 & 0.6193 & 0.0838 \\ 0.0113 & 0.0838 & 0.0113 \end{bmatrix}.$$

In average/Gaussian linear filtering, because of the low-pass filtering effect, local variations caused by scanning noise can be reduced to some extent.

(ii) Median filtering based denoising: Median filtering is similar to using a linear filter, as the value of a pixel in the denoised image I_D is also determined based on the pixel values of its neighborhood in the original scanned image I. However, in median filtering, the pixel value is set to the median of its neighborhood pixels, rather than a weighted sum. Let the median filter be with size $L \times L$, we have

$$I_D(i,j) = \text{median}\left\{ I(k,l), k \in [i - \lfloor L/2 \rfloor, i + \lfloor L/2 \rfloor], l \in [j - \lfloor L/2 \rfloor, j + \lfloor L/2 \rfloor] \right\}. \tag{5.3}$$

Using order statistics, median filtering is robust to extreme values, and therefore able to extract out the outliers in scanned images. The median filtering method helps model the "salt and pepper" noise. The value of $L = 3$ in our experiments.

(iii) Wiener adaptive image denoising: Wiener denoising performs pixel-wise adaptive filtering based on statistics estimated from a local neighborhood of

each pixel. For a pixel $I(i,j)$, its local mean and variance are first estimated from an $L \times L$ neighborhood:

$$\begin{aligned}\hat{\mu}(i,j) &= \frac{1}{L \times L} \sum_{k=-\lfloor L/2 \rfloor}^{\lfloor L/2 \rfloor} \sum_{l=-\lfloor L/2 \rfloor}^{\lfloor L/2 \rfloor} I(i+k, j+l) \\ \hat{\sigma}^2(i,j) &= \frac{1}{L \times L} \sum_{k=-\lfloor L/2 \rfloor}^{\lfloor L/2 \rfloor} \sum_{l=-\lfloor L/2 \rfloor}^{\lfloor L/2 \rfloor} \left(I(i+k, j+l) - \hat{\mu}(i,j)\right)^2 .\end{aligned}$$

These estimated local mean and variance are then used to obtain a denoised pixel value $I_D(i,j)$ through the following adaptive filtering:

$$I_D(i,j) = \hat{\mu}(i,j) + \frac{\hat{\sigma}^2(i,j) - \hat{\sigma}_n^2}{\hat{\sigma}^2(i,j)} (I(i,j) - \hat{\mu}(i,j)), \tag{5.4}$$

where $\hat{\sigma}_n^2$ is an estimate of the noise variance, and can be computed as the average of all the local estimated variances $\hat{\sigma}_n^2 = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \hat{\sigma}^2(i,j)$. In our test, we choose two neighborhood sizes $L = 3, 5$ for Wiener denoising.

In summary, we perform average filtering, Gaussian filtering, median filtering, and Wiener filtering (with two different neighborhood sizes) to a scanned image to obtain five of its denoised versions. For each of the five denoised version, we extract the two features in equation (5.1) from each of the three color components (RGB), and therefore arrive at a total of $5 \times 3 \times 2 = 30$ features from the image denoising aspect.

5.3.2 Features from Wavelet Analysis

In this sub-section, we characterize scanning noise through wavelet analysis. After one stage of 2-D wavelet decomposition [63], an input image is decomposed to be four subbands, namely, low-low (LL), low-high (LH), high-low (HL), and high-high (HH) subbands. Among these four subbands, the LL subband contains

low-frequency components, while the other three are for high-frequency components. In literature, it has been observed that for a large class of images, the wavelet coefficients in LH, HL, and HH subbands do not follow a Gaussian distribution [14, 15, 75]. This is because the spatial structure of these images consists of smooth areas interspersed with occasional edges, and therefore coefficients in the high-frequency subbands are sharply peaked at zero with broad tails. However, because of the presence of scanning noise, we observe that high-frequency wavelet coefficients of scanned images may approach Gaussian distributions. In Fig. 5.4(b), we plot the histogram of the wavelet coefficients in the HH subband of the blue color component for the digital photograph in Fig. 5.4(a). For comparison, the histograms for its corresponding scanned images from an *Epson Perfection 2450 photo* scanner, and a *Canon CanoScan D1250U2F* scanner are shown in Fig. 5.4(c) and (d), respectively. When generating each of these histograms, we normalize the corresponding image to have an unit energy before the wavelet decomposition. For each histogram, we also plot its Gaussian fitting shown as red curves in the figures, whose mean and variance are set as the sample mean and the sample variance of the subband coefficients. We observe that the histogram for the digital photograph cannot be well approximated by Gaussian distributions, while those for scanned images indeed approach Gaussian distributions. Further, we notice that different scanner models produce scanning noise of different variances and affect goodness of the Gaussian fitting to different extents, which are then expected to be beneficial in identifying scanner models.

Based on above observations, we extract statistical noise features in the wavelet domain as follows. Given a scanned image I, we first normalize it to be $\tilde{I}$ with

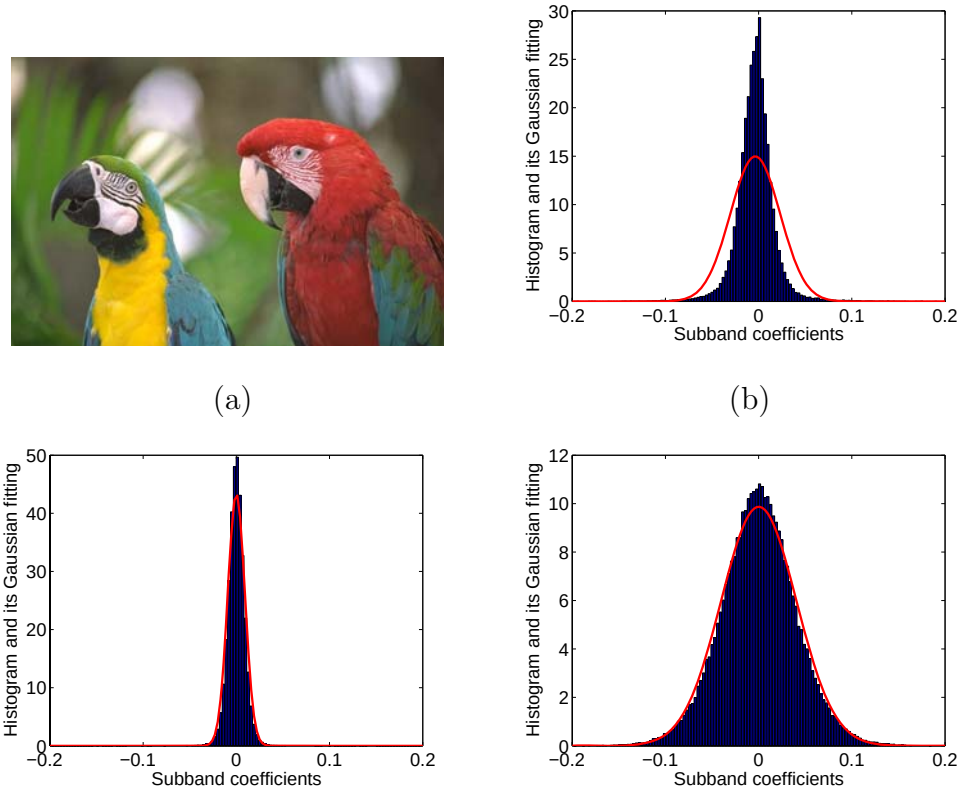


Figure 5.4: Histograms of wavelet coefficients in the HH_1 subband for the blue component: (a) a sample photograph; (b) histogram for the sample photograph; (c) histogram for the scanned image output from an Espon Perfection 2450 photo scanner; (d) histogram for the scanned image output from a Canon CanoScan D1250U2F scanner.

unit energy, i.e.,

$$\tilde{I}(i,j) = \frac{I(i,j)}{\left(\frac{1}{MN}\sum_{k=1}^{M}\sum_{l=1}^{N} I(k,l)^2\right)^{\frac{1}{2}}}. \tag{5.5}$$

Then, we perform one stage 2-D wavelet decomposition to $\tilde{I}$ and obtain its three high-frequency subbands HH, HL, LH. After that, for each of these three subbands, we calculate the mean and the standard deviation of its coefficients $\{Y(u,v)\}$

$$\begin{aligned} \mu_Y &= \frac{1}{u,v}\sum_{u=1}^{U}\sum_{v=1}^{V} Y(u,v), \\ \sigma_Y &= \left(\frac{1}{UV}\sum_{u=1}^{U}\sum_{v=1}^{V}\left(Y(u,v)-\mu_Y\right)^2\right)^{\frac{1}{2}}, \end{aligned}$$

where U and V are the size of the wavelet subband, and we take σ_Y as our third statistical noise feature of image I,

$$f^{(3)}(I) = \sigma_Y. \tag{5.6}$$

With the calculated sample mean μ_Y and sample variance σ_Y^2, we arrive at a Gaussian distribution $\mathbf{N}(\mu_Y, \sigma_Y^2)$, and further quantify the goodness of fitting this Gaussian distribution to the distribution of subband wavelet coefficients $\{Y(u,v)\}$. Let $p(y)$ and $q(y)$ denote the probability density functions (PDFs) of the Gaussian distribution $\mathbf{N}(\mu_Y, \sigma_Y^2)$ and the distribution of subband wavelet coefficients $\{Y(u,v)\}$, respectively. We quantify the goodness of Gaussian fitting by measuring the distance between $p(y)$ and $q(y)$ as

$$\delta_1 = \int |p(y) - q(y)| dy.$$

In our implementation, we approximate this integration by discrete summation to obtain the fourth statistical noise feature

$$f^{(4)}(I) = \sum_i |p(y_i) - q(y_i)| \Delta y, \tag{5.7}$$

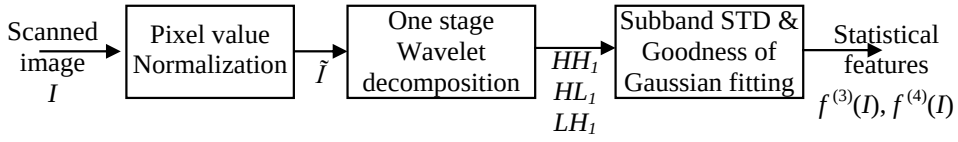


Figure 5.5: Statistical noise feature extraction via wavelet analysis.

where i is the index of histogram bins, Δy is the length of each bin, $p(y_i)$ is the value of Gaussian PDF $p(y)$ at the center of the i^{th} bin, and $q(y_i)$ is the count of the i^{th} bin normalized toward a valid PDF such that $\sum_i q(y_i)\Delta y = 1$.

In Fig. 5.5, we summarize the basic modules of extracting statistical noise features via wavelet analysis. After performing pixel value normalization as in Eqn. (5.5), we apply wavelet decomposition to each color component of a scanned image, and then extract two features (standard deviation of wavelet coefficients and goodness of Guassian fitting) from each of the three high-frequency subbands. Therefore, a total of $3 \times 2 \times 3 = 18$ features are obtained in the wavelet domain.

5.3.3 Features from Neighborhood Prediction

Most images consist of some smooth regions, where pixel values can be predicted from certain neighboring pixels with high accuracy. However, because of the effect of scanning noise, the smooth regions in scanned images may be contaminated, resulting in a non-trivial error in the neighborhood prediction. As scanning noise may vary from one scanner model to another scanner model, such noise variation may be reflected by the difference in the neighborhood prediction error as discussed above. In this subsection, we characterize the scanning noise in terms of the neighborhood prediction error in the smooth regions, and then extract statistical noise features from it.

Given a scanned image I, we identify its smooth regions based on local image

gradient values. As in Section 5.3.2 for wavelet analysis, we first normalize I to be $\tilde{I}$ with unit energy in (5.5). Then, for pixels in the normalized $\tilde{I}$, we calculate their horizontal/vertical gradient values g_h and g_v using linear filtering with Prewitt filters on horizontal/vertical directions, respectively

$$g_h(i,j) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} h_{hor}(k,l)\tilde{I}(i+k, j+l)$$
$$g_v(i,j) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} h_{ver}(k,l)\tilde{I}(i+k, j+l),$$

where

$$h_{hor} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \text{ and } h_{ver} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

Comparing the gradient values with a threshold t_g, we identify pixels in smooth regions as those with both a small horizontal gradient and a small vertical gradient, i.e.,

$$(i,j) \in \texttt{smooth region, if } \; (g_h(i,j) \leq t_g) \wedge (g_v(i,j) \leq t_g).$$

Observing that scanning noise varies in dark and bright areas, we further partition the identified pixels into two regions: a smooth dark region and a smooth bright region, by setting a threshold t_i on the pixel intensity values. In our experiments, we set the gradient threshold $t_g = 0.2$ and the intensity threshold t_i as the median of all the pixel intensity values in the normalized $\tilde{I}$.

For each of the two regions, we now perform neighborhood prediction as follows. For each pixel b_i in a given smooth region, we predict its value using a linear model on its eight neighbors:

$$\hat{b}_i = \sum_{k=1}^{8} x_k a_{i,k}, \tag{5.8}$$

where $a_{i,k}$ is the pixel value of the eight neighbors as indicated in Fig. 5.6, and $x_k \geq 0$ is the weight associated with each of the eight neighbors. We constrain the weights to be non-negative numbers, with the aim of modelling positive correlation between individual pixels with their neighbor pixels. To facilitate discussions, we denote non-negative weight coefficients for each of the two regions as a column vector $\mathbf{x} = [x_1, x_2, \ldots, x_8]^T$, and represent its Q pixel values as a column vector $\mathbf{b} = [b_1, b_2, \ldots, b_Q]^T$. Each of these Q pixels has eight neighbors that can be represented as a row vector $\mathbf{a}_i = [a_{i,1}, a_{i,2}, \ldots, a_{i,8}]$, and all of these row vectors can be organized as a matrix A of size $Q \times 8$. Now we can formulate the estimation of weight coefficients $\mathbf{x}$ as a non-negative least-squares (NNLS) problem

$$\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|^2, \ \textit{subject to} \ x_k \geq 0, k = 1, 2, \ldots, 8, \tag{5.9}$$

The non-negative weight coefficients $\mathbf{x}$ can be found using the *NNLS* algorithm described in [54], whose major points are summarized in Appendix 5.6. After we obtain the weight coefficients $\mathbf{x}$, we calculate the prediction vector $\hat{\mathbf{b}} = \mathbf{A}\mathbf{x}$ with respect to the original pixel value vector $\mathbf{b}$.

$a_{i,1}$	$a_{i,4}$	$a_{i,6}$
$a_{i,2}$	b_i	$a_{i,7}$
$a_{i,3}$	$a_{i,5}$	$a_{i,8}$

Figure 5.6: The neighborhood of b_i in the linear prediction model.

After obtaining the neighborhood predictions, we calculate the errors between the predicted values and the original values, and finally extract certain statistics of the errors as our noise features. Specifically, we calculate the absolute prediction error

$$\Delta\mathbf{b} = |\hat{\mathbf{b}} - \mathbf{b}| = |\mathbf{A}\mathbf{x} - \mathbf{b}|, \tag{5.10}$$

and take its mean and standard deviation as our last two statistical noise features

$$\begin{aligned} f^{(5)}(I) &= \frac{1}{N}\sum_{j=1}^{N}\Delta b_j, \\ f^{(6)}(I) &= \left(\frac{1}{N}\sum_{j=1}^{N}\left(\Delta b_j - f^{(5)}(I)\right)^2\right)^{\frac{1}{2}}. \end{aligned} \tag{5.11}$$

In Fig. 5.7, we summarize the basic modules of extracting statistical noise features via neighborhood prediction. First, we identify pixels in a bright smooth region and a dark smooth region, respectively, according to their intensity and gradient values. Then, for pixels in each of the two regions, we perform neighborhood prediction, and calculate the absolute prediction error. Finally, we extract the mean and the standard deviation of the absolute prediction error as statistical noise features. We apply the same procedure to all the three color components, and therefore a total of $2 \times 2 \times 3 = 12$ features are obtained from the neighborhood prediction.

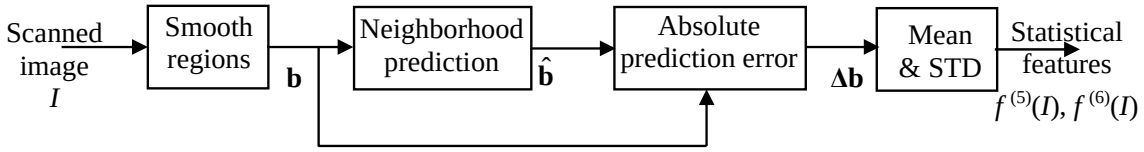


Figure 5.7: Statistical noise feature extraction via neighborhood prediction.

5.4 Experimental Results and Forensic Applications

In this section, we present experimental results to demonstrate the performance of utilizing the proposed statistical noise features for identifying the source and integrity of scanned images. From each of the images in the test, we extract a total

of $30+18+12 = 60$ statistical noise features as discussed in Section 5.3, then apply principle component analysis (PCA) [22] to these extracted features, and finally feed them as inputs to a support vector machine (SVM) called LIBSVM [13] for different identification/verification purposes. In the following, we will first briefly review the SVM classifier with some implementation details. Then, we present results on identifying scanner model sources using the statistical noise features. After that, we broaden the scope of image source identification to differentiating scanned images from camera taken and computer generated images. After that, we demonstrate the effectiveness of using the statistical noise features for identifying post-processing operations and performing blind steganalysis on scanned images. Finally, we compare the proposed scheme with existing works in performing various forensic analysis on scanned images.

5.4.1 Constructing Classifier via Support Vector Machines

Support vector machines provide a powerful tool for data classification. Given a set of training samples, each with l features and a class label, the SVM constructs and solves an optimization problem that maximizes the separate margin among different classes formed by the given training samples. Those training samples that the margin pushes up against are called support vectors. Such an optimization problem can be tackled via solving a dual quadratic problem that involves certain kernel functions. Some examples of kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid kernels. Utilizing non-linear kernels aims at making linearly non-separable samples separable in the transformed non-linear space. In our experiments, we use ν-support vector classification [13] with the non-linear RBF kernel. The parameter $\nu \in (0, 1]$ in the ν-support vector classification pro-

vides an upper bound on the fraction of training errors, and needs appropriate tuning in practical use. The RBF kernel is defined as $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$, where γ is another parameter to be determined in our SVM classification, and x_i, x_j are corresponding feature values of the i^{th} and the j^{th} samples, respectively. The RBF kernel is a reasonably good choice for the kernel function, in that the linear kernel is a special case of the RBF, and the sigmoid kernel behaves like the RBF for certain parameters. Comparing with the polynomial and sigmoid kernels, which involves two or three parameters to be determined, the RBF kernel also eases the parameter tuning.

In practical use of SVM classification, it is also found that certain pre-processing on feature values before applying the SVM classifier can increase classification accuracy [44]. One pre-processing example is linearly scaling values associated with individual features [44] to the range of $[-1, 1]$ or $[0, 1]$. When using the proposed noise features to classify scanner models, we found that taking a $\log_2(\cdot)$ operation on the feature values for $f^{(3)}(I), f^{(4)}(I)$ from wavelet analysis and $f^{(5)}(I), f^{(6)}(I)$ from neighborhood prediction can help increase the accuracy of scanner classification.

To determine the two parameters ν and γ in the above SVM classification, we perform a grid search on the (γ, ν) pair using v-fold cross validation, as recommended in [44]. In the v-fold cross validation, all the training samples are divided into v subsets in a random way. Each of these v subsets is tested using the classifier build upon the remaining $v - 1$ subsets. In this way, every training sample is tested once and only once, and the cross-validation accuracy is the percentage of samples that are correctly classified. In our experiments, for each pair of (γ, ν) with γ in the range of $\{2^{-15}, 2^{-11}, \cdots, 2^4, 2^5\}$ and ν in the range

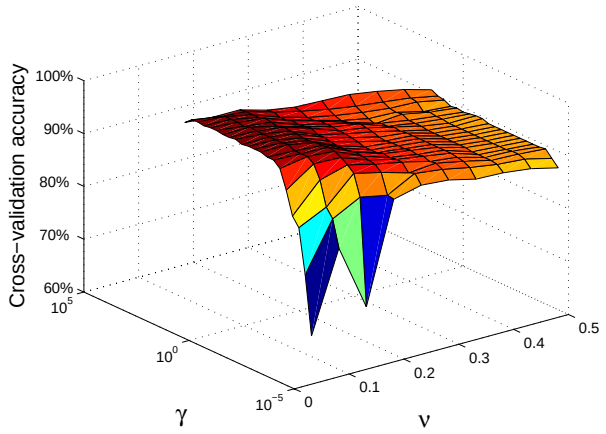


Figure 5.8: Cross-validation results for determining the parameters of SVM classification.

of $\{0.05, 0.1, 0.15, \cdots, 0.35, 0.4\}$, we perform 5-fold cross validation, and finally choose the (γ, ν) pair with the highest cross-validation accuracy. In Fig. 5.8, we show an example of the cross-validation results in determining the two parameters of γ and ν.

5.4.2 Results on Scanner Model Identification

In our test with scanner model identification, we collect six scanner sets from five scanner models, as shown in Table 5.2, among which the last two scanners are of the same model. To generate scanned images, we print out 96 digital photographs of different content on 5" $\times$ 7" glossy photo paper, and then scan each of them back to be a digital image using each of the seven scanners listed in Table 5.2. The same scanning resolution of 150dpi is used for all the scanners. In this way, we build a scanned image database consisting of $96 \times 6 = 576$ images, upon which we examine the performance of scanner model identification using the proposed statistical noise features.

Table 5.2: A list of scanner models in the experiments.

ID	Scanner and its model	Resolution
1	Canon CanoScan D1250U2F	1200 × 2400 dpi
2	Epson Perfection 2450 photo	2400 × 4800 dpi
3	Microtek ScanMaker 3600	600 × 1200 dpi
4	Visioneer OneTouch 5800USB	600 × 1200 dpi
5	CanonScan LiDE70 Set 1	2400 × 4800 dpi
6	CanonScan LiDE70 Set 2	2400 × 4800 dpi

Taking the 60 noise features with parameters (γ, ν) determined as above in Section 5.4.1, we train the SVM with q samples randomly chosen out of the total of n images for each class, and test it with the remaining $n - q$ images. The identification performance is then measured in terms of the *confusion matrix* C, where the element $C_{i,j}$ represents the percentage of images from class i identified as from class j. The average of the diagonal elements $C_{i,i}$ represent the overall classification accuracy. In our experiments, we repeat the above SVM training-testing process 200 times, each time with a different selection of the q training images. Confusion matrices obtained from all the 200 iterations are averaged to arrive at the confusion matrix reported below.

Statistical Noise Features Associated with Scanner Model: In this test, we examine if the statistical noise features are indeed associated with the scanner model and how they perform in terms of identifying the scanner model. First, we take the six scanner sets as the six classes to be identified. Using $q = 48$ images for training and the remaining $n - q = 96 - 48 = 48$ images for testing, we obtain a confusion matrix as shown in Table 5.3. We can see that a non-

trivial amount of confusion exists only between the last two scanners coming from the same model, and all the other scanners of different models can be accurately identified. To further demonstrate that the proposed statistical noise features are indeed associated with the scanner model but not with individual scanner sets, we take the five scanner models as the five classes, and perform the following two training-testing tasks. The first one uses the same scanner set "CanonScan LiDE 70 Set 1" for both the training and the testing, while the second one uses "CanonScan LiDE 70 Set 2" for the training but "CanonScan LiDE 70 Set 1" for the testing. In Table 5.4(a) and (b), we show the confusion matrices for the above two tests, respectively, with half of the images for the training and the remaining half for the testing as before. We can see that both cases are able to identify the correct scanner models with very high accuracy of above 98%. Similar results are obtained when images from "CanonScan LiDE 70 Set 1" are used for the training, but those from "CanonScan LiDE 70 Set 2" for the testing, as shown in Table 5.4(c) for the confusion matrix. The above tests demonstrate the effectiveness of our proposed scheme in scanner model identification. In Table 5.4(d), we show the confusion matrix of the 5-scanner model identification with each of the 96 scanned image samples for the "CanonScan LiDE 70" model randomly chosen from its 2 scanner sets with an equal probability.

Scanner Model Identification with Robustness against Post-Processing: In the following, we examine robustness of the scanner model identification against various post-processing operations on scanned images. We take the five scanner models as the five classes to be identified, and use half of the images for training and the remaining half for testing. When multiple ($m > 1$) scanner sets are available for a single scanner model, each of the n scanned image samples for this model

Table 5.3: Confusion matrix with 48 training images in a total of 96 images for the proposed scheme, with the six scanners listed in TABLE 5.2 as the six classes.

ID	1	2	3	4	5	6
1	98%	0	1%	1%	0	0
2	0	99%	0	0	1%	0
3	1%	0	99%	0	0	0
4	1%	0	0	99%	0	0
5	0	1%	0	0	85%	14%
6	0	0	0	0	17%	83%

will be randomly chosen from the m scanner sets with an equal probability of $\frac{1}{m}$.

First, we examine how the size of scanned images affects the accuracy of scanner model identification. In order to do so, we divide each of the scanned images of size $M \times N$ into image blocks of size $M \times \frac{N}{2}$ and $\frac{M}{2} \times \frac{N}{2}$, respectively, and then use these image blocks as scanned image samples for identifying the scanner models. For the above two image partition cases, we now have $n = 96 \times 2 = 192$ ($n = 96 \times 4 = 384$) scanned image samples for each of the five scanner models, and we obtain confusion matrices as shown in Table 5.5(a)((b)) with an overall classification accuracy of 99% (98%). Further we collect image blocks of different sizes to form the n samples for each of the five scanner models. Specifically, for the 96 scanned images directly output from each scanner, we randomly divide them into three groups, with 32 images for each group. Images in the second and third group are divided into image blocks of size $M \times \frac{N}{2}$ and $\frac{M}{2} \times \frac{N}{2}$, respectively, which are then combined with the images in the first group to form a total of $n = 32 + 32 \times 2 + 32 \times 4 = 224$ image samples for each scanner. Using these 224 image samples for scanner model

Table 5.4: Confusion matrix with 48 training images in a total of 96 images for the proposed scheme, with the five scanner models listed in TABLE 5.2 as the five classes: (a) the same scanner "CanonScan LiDE70 Set 1" used for both training and testing, (b) "CanonScan LiDE70 Set 2" used for training but "CanonScan LiDE70 Set 1" used for testing, (c) "CanonScan LiDE70 Set 1" used for training but "CanonScan LiDE70 Set 2" used for testing, (d) Each scanned image sample for the "CanonScan LiDE70" model is randomly chosen from its 2 sets with an equal probability.

(a)

ID	1	2	3	4	5
1	99%	0	1%	0	0
2	0	99%	0	0	1%
3	1%	1%	98%	0	0
4	1%	1%	0	98%	0
5	0	1%	0	0	99%

(b)

ID	1	2	3	4	6
1	98%	0	1%	1%	0
2	0	99%	0	0	1%
3	1%	1%	98%	0	0
4	1%	0	0	98%	1%
5	0	2%	0	0	98%

(c)

ID	1	2	3	4	5
1	98%	0	1%	1%	0
2	0	99%	0	0	1%
3	1%	1%	98%	0	0
4	1%	1%	0	98%	0
6	0	1%	0	0	99%

(d)

ID	1	2	3	4	5/6
1	98%	0	1%	1%	0
2	0	99%	0	0	1%
3	1%	0	99%	0	0
4	0	1%	0	98%	1%
5/6	0	1%	0	0	99%

identification, we obtain a confusion matrix as shown in Table 5.5(c) with an overall classification accuracy of 98%. From the above results, we can see that scanner models can be accurately identified even though the scanned images are in different sizes. To evaluate the impact of image content in identifying scanner models, we carry out the following test. We randomly partition the $96 \times 4 = 384$ scanned image blocks of size $\frac{M}{2} \times \frac{N}{2}$ into six sets, with 64 blocks for each set. We then use the 64 image blocks in set $1, 2, \cdots, 5$ as the training samples for scanner model $1, 2, \cdots, 5$, respectively. Performing testing with image blocks in the last set for all of the five scanner models, we obtain a confusion matrix as shown in Table 5.5(d) with an overall classification accuracy of 97%. This demonstrates that our scheme does not require different scanner models capture the same images for building the scanner model identifier.

Second, we examine the performance of scanner model identification with scanned images having gone through various post-processing operations. In Table 5.6, we list the post-processing operations as well as the overall classification accuracy corresponding to each of them. The results here are obtained under an informed experiment, with half of the scanned images after a certain post-processing operation for the training, and the remaining half after the same post-processing operation for the testing. Under many post-processing operations, such as additive noise, rotation, gamma correction, sharpening, contrast enhancement, scaling and JPEG compression with a factor large enough, the proposed statistical noise features are still capable of identifying the scanner models with high accuracy. Under strong average and median filtering, scaling with a very small factor, and JPEG compression with a low quality factor, the scanning noise is severely smoothed out, resulting in the performance drop on scanner model identification. Among

Table 5.5: Confusion matrix with half of the available image blocks of certain sizes for training and the remaining half for testing using the proposed scheme, with the five scanner models listed in TABLE 5.2 as the five classes, and using (a) image blocks of size $M \times \frac{N}{2}$, (b)image blocks of size $\frac{M}{2} \times \frac{N}{2}$, (c)image blocks of the three different sizes $M \times N$, $M \times \frac{N}{2}$, and $\frac{M}{2} \times \frac{N}{2}$, and (d) image blocks of different content for training different scanner models.

(a)

ID	1	2	3	4	5
1	98%	0	2%	0	0
2	1%	99%	0	0	0
3	1%	0	99%	0	0
4	1%	0	1%	98%	0
5	0	1%	0	0	99%

(b)

ID	1	2	3	4	6
1	98%	0	2%	0	0
2	1%	98%	0	0	1%
3	0	1%	99%	0	0
4	1%	0	0	99%	0
5	0	2%	0	0	98%

(c)

ID	1	2	3	4	5
1	98%	0	1%	1%	0
2	0	99%	0	0	1%
3	1%	1%	98%	0	0
4	1%	1%	0	98%	0
6	0	1%	0	0	99%

(d)

ID	1	2	3	4	5/6
1	98%	0	2%	0	0
2	0	97%	1%	0	2%
3	1%	1%	98%	0	0
4	1%	0	0	98%	1%
5/6	0	3%	0	1%	96%

the six post-processing categories with an identification accuracy higher than 95%, {additive noise with a Signal to Noise Ration of 5dB, rotation of 5 degrees, resizing with a scaling factor of 1.25, gamma correction with $\gamma = 1.25$, sharpening with the filter parameter equaling to 0.2, and contrast enhancement}, we further perform the following non-informed experiment. Each of the $n = 96$ image samples for each scanner model is now randomly chosen from its six post-processed versions as above with an equal probability of $\frac{1}{6}$. Still using half of the 96 selected image samples for the training and the remaining half for the testing, we obtain a confusion matrix as shown in Table 5.7 for the scanner model identification. For this challenging task of non-informed scanner identification, the proposed scheme still achieves a high overall classification accuracy of 90%, demonstrating the robustness of our scanner model identifier.

5.4.3 Identifying Image Acquisition Apparatus

Besides identifying the model of the scanner used to capture individual scanned images, some forensic scenario may require scanned images be differentiated from camera taken images and computer generated images. Considering that images produced by digital cameras or computer programs contain noise of different amounts and properties from scanned images, here we evaluate how the proposed statistical noise features would help on broadening the scope of *acquisition forensics* to multiple kinds of image acquisition apparatus: scanner, camera, or computer program.

In carrying out the evaluation, we randomly select each of the 96 scanned images from the five scanner models we have with an equal probability. Similarly, we select 96 digital images taken by five different camera models, Canon G6, OlympusC3100Z, Canon Powershot S400, Minolta DiMAGE F100, and Nikon E4300, as

Table 5.6: Overall classification accuracy of scanner model identification after post-processing operations on scanned images.

Post-processing with parameters	*Average filtering*			*Median filtering*			*Additive noise*	
	Order size			*Order size*			*SNR (dB)*	
	3	5	7	3	5	7	5	10
Classification accuracy	89%	85%	79%	88%	81%	76%	98%	99%
Post-processing with parameters	*Rotation*				*Resizing*			
	Angle				*Scaling factor*			
	1	5	10	20	0.5	0.75	1.25	1.5
Classification accuracy	96%	96%	96%	94%	87%	94%	97%	97%
Post-processing with parameters	*JPEG compression*				*Gamma correction*			
	Quality factor				γ			
	100	95	90	75	0.5	0.75	1.25	1.5
Classification accuracy	93%	86%	77%	59%	98%	98%	97%	97%
Post-processing with parameters	*Sharpening*				*Contrast enhancement*			
	Unsharp filter parameter							
	0	0.2	0.4	0.6				
Classification accuracy	98%	98%	98%	98%	96%			

Table 5.7: Confusion matrix with 48 training images in a total of 96 images for the proposed scheme, with the five scanner models listed in TABLE 5.2 as the five classes, and each of the 96 image samples is randomly chosen from six of its post-processed versions.

ID	1	2	3	4	5/6
1	93%	2%	2%	2%	1%
2	4%	87%	3%	0	6%
3	2%	3%	91%	3%	1%
4	5%	3%	2%	89%	1%
5/6	2%	8%	0	0	90%

well as 96 computer generated images provided by [68]. We then calculate the 60 statistical noise features for each of these images, and use the SVM to differentiate scanned images from 1) camera taken images and 2) computer generated images. For each of the above two identification tasks, we randomly choose half of the images from each category for training, and test on the remaining images. Computing the fraction of correctly identified scanned images P_d, and the percentage of non-scanned images wrongly classified as scanned images P_f, we obtain the receiver operating characteristics (ROC). In Fig 5.9, we show the ROCs averaged over 200 iterations for identifying scanned images from the other two kinds of image acquisition apparatus, each time with a different selection of the 48 training images. We observe from the figure that the identification performance is very good, with the P_d around 90% (94%) under a very low P_f of 1% for differentiating scanned images from camera taken images (computer generated images). This suggests that the proposed statistical noise features are capable of identifying scanned images from other kinds of image acquisition apparatus.

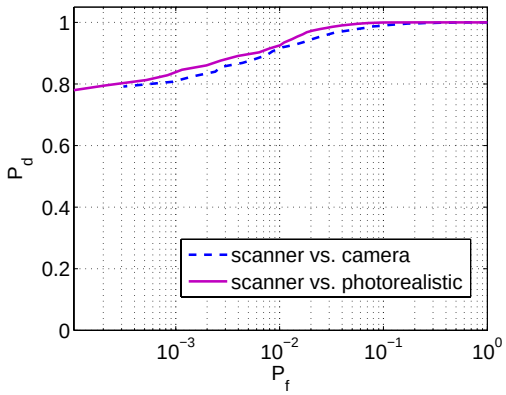


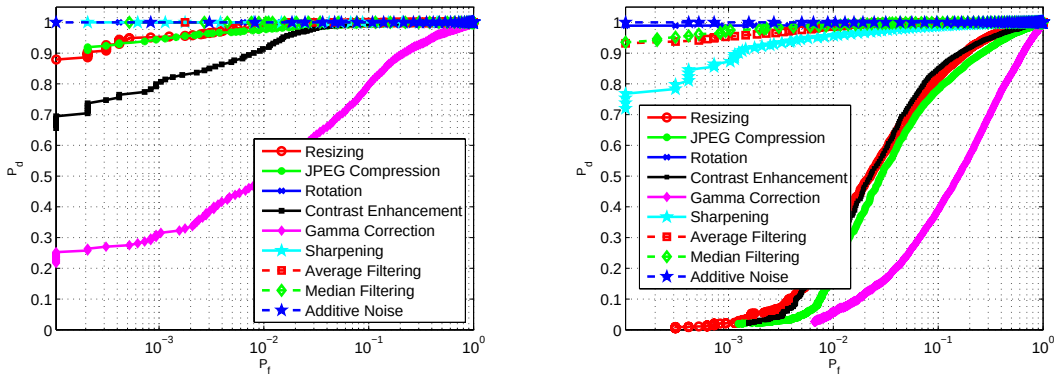
Figure 5.9: Receiver Operating Characteristics for identifying scanned images from camera taken images and computer generated images.

5.4.4 Integrity Forensic Analysis on Scanned Images

When scanned images directly output from scanners are modified to carry stego information or go through certain post-processing operations, such as low-pass filtering, scaling, and rotation as possibly involved in tampering a scanned image, with a high probability the intrinsic statistical noise features of the original scanned images shall be altered. In the following, we examine how the proposed scheme performs in such integrity forensic analysis on scanned images as detecting post-processing after scanning and performing steganalysis on scanned images.

Results for Identifying Post-processing after Scanning: In the test of post-processing identification, each scanned image directly output from a scanner is processed to generate 29 manipulated versions by using the 29 post-processing operations listed in Table 5.6. First, we take the 96 scanned images from a single scanner model, CanonScan D1250U2F, as the original unprocessed scanned images, and calculate the 60 noise features for each of them. For each of the nine types of post-processing operations listed in Table 5.6, we randomly choose the 96 post-processed images from the operations listed in Table 5.6 for each type, and then feed them into the SVM to classify the original scanned images from the post-processed scanned images. Similar to the above image acquisition analysis, we randomly choose 48 original images along with their corresponding post-processed versions for training, and test on the remaining images to obtain the ROC which indicates the relationship between the probability of detecting post-processing P_d and the probability of false alarm P_f. Fig. 5.10(a) shows the ROC averaged over 200 iterations for each of the nine types of post-processing, each time with a different selection of the 48 training images. Similarly, Fig. 5.10(b) reports the ROC curves when the 96 original scanned images and their corresponding post-

processed versions are randomly chosen from all the five scanner models. We observe from both figures that the identification performance is very good for most post-processing manipulations, especially when scanned images are coming from a single scanner model. This indicates that the proposed noise features can capture the changes from a direct scanner output to its further manipulated versions, and therefore can effectively detect post-processing on scanned images.



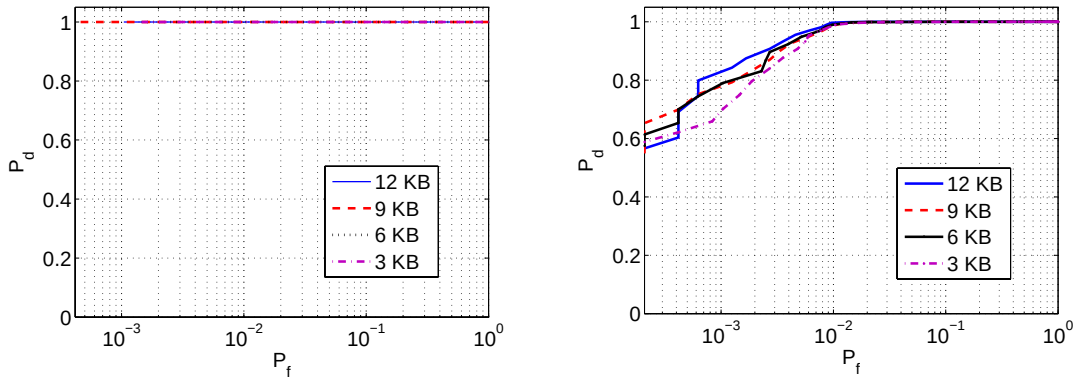
(a) (b)

Figure 5.10: Receiver Operating Characteristics for identifying nine types of post-processing operations: (a) scanned images are from a single scanner model; (b) scanned images are from multiple scanner models.

Results for Steganalysis on Scanned Images: In the steganalysis test, we use 96 scanned images directly output from a single scanner model or from multiple scanner models as the cover images, similar to the way of choosing the unprocessed images in the test of post-processing detection. From each cover image, a number of stego images are generated by embedding random messages of different sizes. In a general scenario, the maximum embedding payload depends on the nature of the cover data and the steganographic embedding method. In our test, we first find the average of the maximum embedding payload across the 96 cover images and then

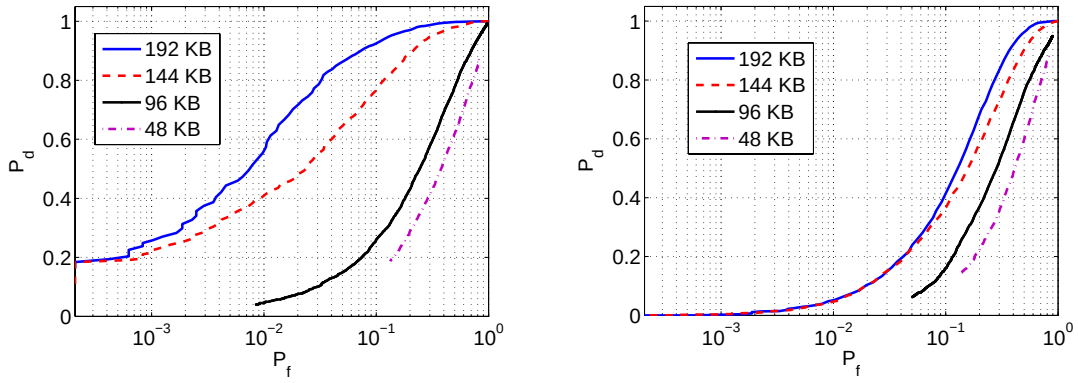
embed messages at 100%, 75%, 50%, and 25% of this value. In our current studies, we consider two popular types of least significant bit (LSB) based steganographic algorithms, namely, F5 [86], and hide4pgp [4].

In the case of F5, the maximum embedding payload averaged over our 96 cover images is around 12 KB. Corresponding to the percentages of 100%, 75%, 50%, and 25%, messages of sizes 12 KB, 9 KB, 6 KB, and 3 KB are embedded into each of the 96 cover images to generate $4 \times 96 = 384$ stego images in total. For each of the four message sizes, we perform SVM training and testing as in the post-processing detection test, and obtain ROC curves as shown in Fig. 5.11(a) when the 96 cover images come from a single scanner model and in Fig. 5.11(b) when they are from multiple scanner models. We can see that the performance in discriminating the cover and stego images is extremely good, no matter the cover images are from a single scanner model or from multiple models. Further, we notice that the discrimination performance is independent of the embedding rate, suggesting that the proposed noise features can perform accurate steganalysis on F5 even under low embedding payloads. Similarly, we show the results for the hide4pgp algorithm in Fig. 5.12 (a) and (b) for the single scanner model case and for the multiple scanner model case, respectively. The maximal embedding payload of the hide4pgp algorithm averaged over the 96 cover images is 192 KB, and we show the ROC curves under 100%, 75%, 50%, and 25% of this maximal average embedding payload. In performing steganalysis for the hide4pgp algorithm, we observe that the steganalysis performance improves as the embedded message size increases, and better discrimination is obtained when the cover images are from a single scanner model.



(a) (b)

Figure 5.11: Receiver Operating Characteristics for steganalysis on the F5 embedding algorithm: (a) scanned images are from a single scanner model; (b) scanned images are from multiple scanner models.



(a) (b)

Figure 5.12: Receiver Operating Characteristics for steganalysis on the hide4pgp embedding algorithm: (a) scanned images are from a single scanner model; (b) scanned images are from multiple scanner models.

5.4.5 Comparisons with Related Prior Art

As pointed out in Section 5.1, classifier based approaches are also used in some prior work for tampering detection [5, 23] and blind steganalysis [6, 24] on digital images. In this subsection, we examine how the features in these two works perform in terms of scanner model identification. For the purpose of comparison, we will also report the performance of using the features in [23, 24] for identifying image acquisition apparatus, for detecting post-processing operations after scanning, and for performing steganalysis on scanned images.

In our experiments, for each of the $96 \times 5 = 480$ scanned images in our database from the five scanner models, we calculate 216 high-order wavelet statistic features in [23] and 45 image quality metric features in [6], respectively. Then, we borrow each of these two groups of features for scanner model classification, via sequentially applying the PCA and SVM as used before. In Table 5.8(a), for using these wavelet statistic features, we show the confusion matrix averaged over 200 iterations with 50% of the scanned images in the training set and the remaining 50% in the testing set, and we notice that the overall classification accuracy is around 81%. Similarly, for using image quality metric features in [6], we obtain a confusion matrix under the *half-half* scenario as shown in Table 5.8(b), which gives us an overall classification accuracy of 79%. From the results here, we can see that both values for the overall classification accuracy, 81% and 79%, are much smaller than the accuracy of 98% achieved by using our statistical noise features in the same *half-half* scenario.

In the following, we present the results of using the features in [23] for differentiating scanned images from camera taken images and computer generated images, as well as for detecting post-processing operations after scanning and perform-

Table 5.8: Confusion matrix with 48 training images in a total of 96 images for borrowing features from prior works to identify the five scanner models listed in TABLE 5.2, and each scanned image sample for the "CanonScan LiDE70" model is randomly chosen from its 2 sets with an equal probability: (a) Farid-Lyu's scheme [23], (b)Avcibas *et al.*'s scheme [6].

(a)

ID	1	2	3	4	5/6
1	86%	1%	9%	4%	0
2	0	81%	2%	0	17%
3	13%	2%	78%	5%	2%
4	6%	2%	4%	88%	0
5/6	3%	21%	2%	0	74%

(b)

ID	1	2	3	4	5/6
1	84%	10%	4%	2%	0
2	9%	76%	4%	2%	9%
3	5%	4%	76%	14%	1%
4	6%	4%	14%	75%	1%
5/6	2%	13%	2%	1%	82%

ing steganalysis on scanned images. Using the 216 high-order wavelet statistic features in [23], we obtain ROC curves as shown in Fig. 5.13 for differentiating scanned images from camera taken image and computer generated images. In Fig. 5.14 and Fig. 5.15, we show the ROC curves for post-processing identification and steganalysis on scanned images, respectively. Comparing results here with the results obtained in Section 5.4.3∼5.4.4 by using the proposed statistical noise features, we can see that the proposed scheme has comparable performance with or outperforms the prior art for most identification tasks involved in differentiating scanned images from camera taken images and computer generated images, as well as in post-processing detection and steganalysis on scanned images. Throughout the comparisons in this subsection, we demonstrate the advantages of the proposed scheme in addressing forensic analysis problems related to scanned images. This is because our statistic noise features represent unique characteristics of scanners

and scanned images, and therefore can accurately reflect the intrinsic traces of scanning devices.

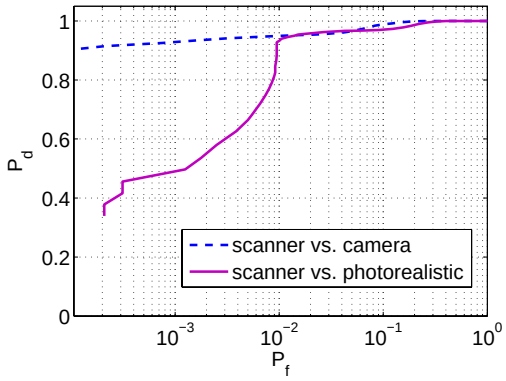
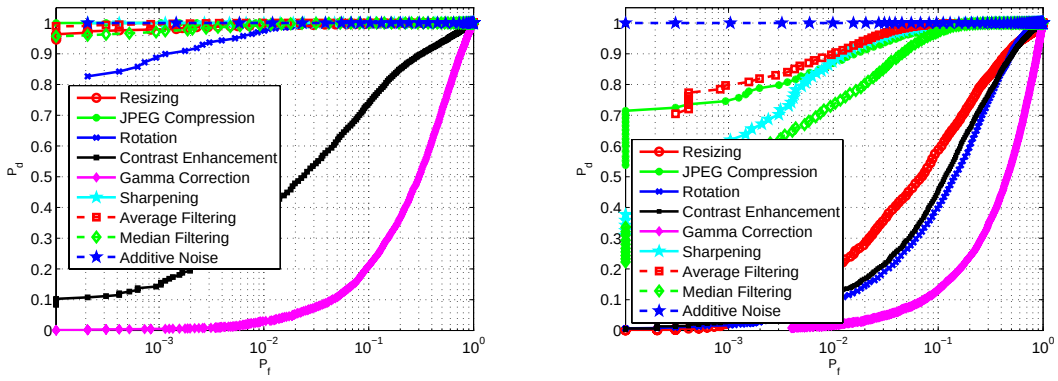


Figure 5.13: Receiver Operating Characteristics of Farid-Lyu's scheme [23] for differentiating scanned images from camera taken images and computer generate images.

5.5 Chapter Summary

In this chapter, we have proposed a novel approach to identifying the source and integrity of scanned images, based on extracting statistical features of scanning noise from scanned image samples. We characterize scanning noise from multiple perspectives, and then obtain statistical noise features from each characterization. Applying image denoising algorithms, we obtain estimates of the scanning noise, and then extract the first set of features from them. We observe that scanning noise affects the non-Gaussian property of wavelet subband coefficients, and therefore leverage it to extract the second set of features. We also perform neighborhood prediction and utilize the prediction error for extracting the third set of scanning noise features. Using these three sets of features, we build a robust scanner identifier



(a) (b)

Figure 5.14: Receiver Operating Characteristics of Farid-Lyu's scheme [23] for post-processing identification: (a) scanned images are from a single scanner model; (b) scanned images are from multiple scanner models.

that can determine the scanner model used to capture individual scanned images with high accuracy and strong resilience to a number of post-processing operations on scanned images. We also examine the the performance of using the proposed statistical noise features to differentiate scanned images from camera taken images and computer generated images, as well as to detect post-processing operations and to perform steganalysis on scanned images. The proposed technique provides a new way to establishing the trust-worthiness of scanners and scanned images, and foster secure exchange of scanned documents with widespread applications in the modern information era.

5.6 Appendix: Solving Nonnegative Least Squares

The non-negative least-squares (NNLS) problem is defined by

$$\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|^2 \; \textit{subject to} \; \mathbf{x} \geq \mathbf{0}, \tag{5.12}$$

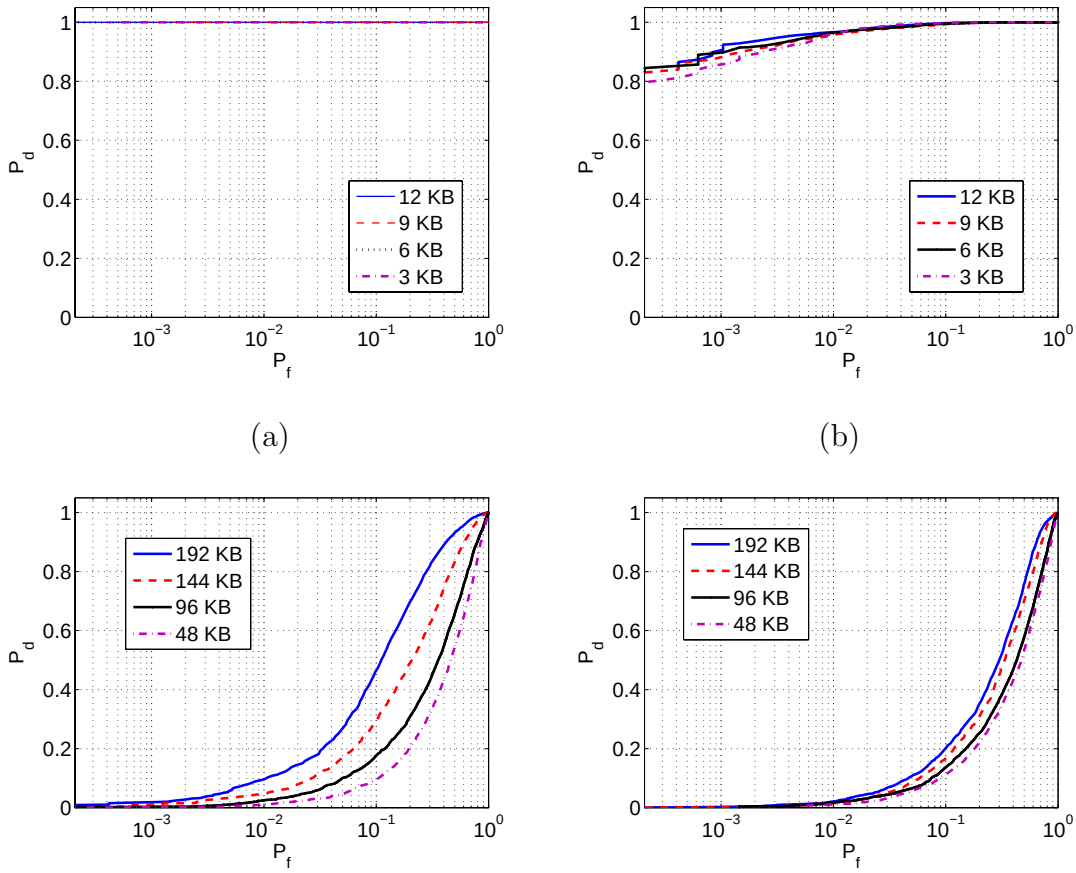


Figure 5.15: Receiver Operating Characteristics of Farid-Lyu's scheme [23] for Steganalysis on scanned images: (a) F5 embedding algorithm with scanned images from a single scanner model; (b) F5 embedding algorithm with scanned images from multiple scanner models; (c) hide4pgp embedding algorithm with scanned images from a single scanner model; (d) hide4pgp embedding algorithm with scanned images from multiple scanner models.

where A is of size $N \times n$, $\mathbf{x}$ of size $n \times 1$, and $\mathbf{b}$ of size $N \times 1$. To find the non-negative weight coefficients $\mathbf{x}$, we define an $n \times 1$ dual vector $\mathbf{w} = A^T(\mathbf{b} - A\mathbf{x})$, and further obtain a partition of the set of $\{1, 2, \ldots, n\}$ into two index subsets $\mathcal{P}$ and $\mathcal{Z}$ so that

$$\begin{aligned} x_j > 0 \quad \texttt{for} \quad j \in \mathcal{P}, \quad x_j = 0 \quad \texttt{for} \quad j \in \mathcal{Z} \\ w_j = 0 \quad \texttt{for} \quad j \in \mathcal{P}, \quad w_j \leq 0 \quad \texttt{for} \quad j \in \mathcal{Z} \end{aligned} \tag{5.13}$$

Equations in (5.13) constitute the Karush-Kuhn-Tucker(KKT) conditions characterizing a solution vector $\mathbf{x}$ for the NNLS problem, and can be established using the *NNLS* algorithm [54]. This algorithm starts with $\mathcal{P} = \Phi$, $\mathcal{Z} = \{1, 2, \ldots, n\}$, and $\mathbf{x} = \mathbf{0}$. Then, it iteratively modifies the set partition as well as the value of $\mathbf{x}$ in the course of executing the algorithm. Each time an index t will be moved from the set $\mathcal{Z}$ to the set $\mathcal{P}$, according to the maximal value in the dual vector $\mathbf{w}$: $w_t = \max\{w_j, j \in \mathcal{Z}\} \wedge w_t > 0$. Variables with indices in the set $\mathcal{Z}$ will be held at the value 0, while variables with indices in the set $\mathcal{P}$ will take values determined by solving a least squares problem. If a variable indexed in the set $\mathcal{P}$ takes a non-positive value, the algorithm will either make the variable take a positive value, or set the variable to 0 and move its index from the set $\mathcal{P}$ to the set $\mathcal{Z}$. The algorithm terminates when either the set $\mathcal{Z}$ becomes empty or there is no positive number in the dual vector, i.e.,$w_j \leq 0, \ \forall j \in \mathcal{Z}$.

Chapter 6

Conclusions and Perspectives

In this dissertation, we have investigated two complementary mechanisms for performing forensic analysis on various graphic data to facilitate their proper distribution and usage. In the area of constructing forensic systems via extrinsic data embedding, we have developed robust digital fingerprinting for curve-based graphics as well as 3-D digital elevation maps(DEMs) for tracing traitors. We have also investigated high-payload watermark embedding for binary images to facilitate authentication of such graphic data as critical document and signature images. In the area of constructing forensic systems via intrinsic sensor features, we have studied sensor noise features of scanners for non-intrusive scanner forensics on verifying the source and integrity of digital scanned images.

In Chapter 2, we have proposed a robust data embedding method for curves and investigates its feasibility for digital fingerprinting of such curve-based graphic data as topographic maps and writhings/drawings for traitor tracing. In our method, we have parameterized a curve using the B-spline model and then added spread spectrum sequences to B-spline parameters. In conjunction with the basic embedding and detection techniques, we have proposed an iterative alignment-minimization

algorithm to allow for robust fingerprint detection under unknown geometric transformations and in the absence of explicit point correspondence. We have demonstrated the proposed method can achieve high-fidelity curve embedding as well as effective fingerprint detection with strong robustness against a number of challenging attacks.

In Chapter 3, we have extended the curve-based data embedding in Chapter 2 to fingerprinting digital elevation maps for the same purpose of tracing traitors. We have developed the framework and algorithms for cross-dimensional DEM fingerprinting via hiding data in a DEM's 2-D contour curves. Using the proposed method, digital fingerprints can be reliably detected from both a 3-D DEM and its 2-D rendering, whichever format available to a detector. In this chapter, we have also examined the tradeoff between fidelity of fingerprinted DEMs and robustness of fingerprints, with several fine-tuning techniques to adjust the tradeoff.

In Chapter 4, we have studied high-payload and high-fidelity data embedding in binary images, and shown its feasibility for binary image authentication. Our work has incorporated a recent steganography framework known as wet paper coding, which can naturally handle the uneven embedding capacity in a binary image through randomized projections. By jointly considering the embedding of multiple bits, the wet paper coding also allows for a high utilization of flippable pixels available, thus providing a high embedding payload for binary images. With this potential in embedding a large payload, it is important to preserve the perceptual quality of watermarked binary images. To achieve this purpose, we have developed an adaptive trimming method to perform flippability assignment on individual pixels as well as a new concept of super-pixels for addressing flippability dependencies of a group of pixels.

In Chapter 5, we have focused on non-intrusive forensic analysis on scanners and scanned images via detecting intrinsic sensor noise features, and demonstrated various forensic applications in identifying the source and integrity of scanned images. By using only scanned image samples, we have characterized scanning noise from multiple perspectives, including through image de-noising, wavelet analysis, and neighborhood prediction, and then obtained statistical noise features from each characterization. Building upon these intrinsic noise features, we have constructed a robust scanner identifier that can determine the scanner model used to capture individual scanned images with high accuracy and strong resilience to a number of post-processing operations on scanned images. Using these noise features, we have further broadened the scope of acquisition forensics to differentiating scanned images from camera taken images and computer generated images, as well as enabled such integrity forensic analysis as detecting post-processing and stegonagraphy operations on scanned images.

Throughout this dissertation, we have shown that when developing forensic analysis systems for graphic data, it is important to identify characteristics unique to each type of graphic data, and then to develop extrinsic data embedding algorithms and/or intrinsic sensor features in a way of tailoring to these unique characteristics. In the following, we point out possible extensions of this dissertation that can be pursued further.

In the extrinsic data embedding area, first, our B-spline based curve fingerprinting method may be extended to fingerprinting 3-D computer graphics, where B-spline curves and surfaces are commonly employed in their vectorized surface representations. In a discrete digital setting, 3-D computer graphics are created through digitally synthesizing and manipulating various geometric shapes and ob-

jects. Considering design complexity and efforts in creating 3-D computer graphics as well as their commercial values, it is desirable to embed digital fingerprints into 3-D computer graphics at the time of their creation. Based on the fact that 3-D computer graphics are directly generated in the digital domain and possibly with B-spline parameterization, it is both efficient and feasible to apply B-spline based digital fingerprinting to 3-D computer graphics. Second, wet paper codes can be employed and modified correspondingly to enhance the performance of hiding data in some specific binary images such as halftone images. In the wet paper codes, the pseudo-random matrix D plays an important role, with its elements being able to have certain structure or follow some probability distribution. Meanwhile, for a specific kind of binary image, there may exist a clever way to choose flippable pixels. Jointly considering the above two aspects, the design of the D matrix and the selection of flippables, is expected to arrive at an improved data embedding system for those specific binary images.

In the area of detecting intrinsic sensor features, performing non-intrusive forensic analysis on various computer generated graphics can be an interesting perspective to explore. In the entertainment industry, a device known as 3-D scanners[1] has been extensively used to create 3-D graphics for both movies and video games. Instead of collecting color information as in cameras and scanners, 3-D scanners collect distance information about surfaces within its field of view. Such distance information from multiple scans and different directions can then be used to construct 3-D models for creating computer graphics. Several types of 3-D scanners have been developed with a variety of technologies. When there does not exist a real-world equivalent of the desired model, certain 3-D modelling software is

[1]http://en.wikipedia.org/wiki/3D_scanner

needed to construct 3-D models. Additionally, various algorithms are employed in generating computer graphics after obtaining the models. There are also many software programs for manipulating computer graphics. Given computer generated graphics in movies, video games, or as digital art[2], it would be interesting to know which hardware devices and software algorithms have been employed to acquire/manipulate them, and non-intrusive forensic analysis using intrinsic hardware/software features may help on answering this question.

[2]http://en.wikipedia.org/wiki/Digital_art

BIBLIOGRAPHY

[1] `http://www.federalreserve.gov/paymentsystems/truncation/default.htm`: "Check Clearing for the 21st Century Act".

[2] `http://www.adobe.com/svg/main.html`: "Scalable Vector Graphics".

[3] `http://www.ngdc.noaa.gov`: National Geophysical Data Center (NGDC).

[4] *Hide4pgp*, Steganography software available online at `http://www.heinz-repp.onlinehome.de/Hide4PGP.htm`.

[5] I. Avcibas, S. Bayram, N. Memon, M. Ramkumar, and B. Sankur, "A Classifier Design for Detecting Image Manipulations," *Proceedings of IEEE International Conference on Image Processing (ICIP)*, Vol. 4, pp. 24–27, Singapore, Oct. 2004.

[6] I. Avcibas, N. Memon, and B. Sankur, "Steganalysis Using Image Quality Metrics," *IEEE Transaction on Image Processing*, Vol. 12, No. 2, Feb. 2003.

[7] M. Barni, F. Bartolini, A. Piva, and F. Salucco, "Robust Watermarking of Cartographic Images," *EURASIP Journal on Applied Signal Processing*, vol. 2, pp. 197–208, 2002.

[8] S. Bayram, H. T. Sencar, N. Memon, and I. Avcibas, "Source Camera Identification based on CFA Interpolation," *Proceedings of IEEE International Conference on Image Processing (ICIP)*, 2005.

[9] E. Belogay, C. Cabrelliay, U. Molter, and R. Shonkwiler, "Calculating the Hausdorff Distance between Curves," *Information Processing Letters*, vol. 64, pp. 17–22, 1997.

[10] H. D. Brunk, *An Introduction to Mathematical Statistics*. Boston: Ginn and Company, 1960.

[11] C. Cabrelli and U. Molter, "Automatic Representation of Binary Images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, no. 12, pp. 1190–1196, 1990.

[12] H. Chang, T. Chen, and K. Kan, "Watermarking 2D/3D Graphics for Copyright Protection," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 720–723, 2003.

[13] C-C. Chang and C-J. Lin, "LIBSVM: a Library for Support Vector Machines," 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[14] S. G. Chang, B. Yu, and M. Vetterli, "Spatially Adaptive Wavelet Thresholding with Context Modeling for Image Denoising," *IEEE Trans. on Image Processing*, Vol. 9, no. 9, pp. 1522–1531, Sept. 2000.

[15] S. G. Chang, B. Yu, and M. Vetterli, "Adaptive Wavelet Thresholding for Image Denoising and Compression," *IEEE Trans. on Image Processing*, Vol. 9, no. 9, pp.1532–1546, Sept. 2000.

[16] F. S. Cohen and J. Y. Wang, "Modeling Image Curves using Invariant 3-D Object Curve Models - a Path to 3-D Recognition and Shape Estimation from Image Contours," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pp. 1–12, Jan. 1994.

[17] C. Cooper, "On the Rank of Random Matrices," *Random Structures and Algorithms* vol.16, no.2, pp. 209-232, 2000.

[18] T. Cover and J. Thomas,"Elements of Information Theory," Wiley Series in Telecommunications, John Wiley and Sons, New York, 1991.

[19] I. Cox, J. Bloom, and M. Miller, *Digital Watermarking*, Morgan Kaufmann, 2001.

[20] I. Cox, J. Killian, F. Leighton, and T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia," *IEEE Transcations on Image Processing*, vol. 6, no. 12, pp. 1673–1687, Dec. 1997.

[21] H. S. M. Coxeter, *Introduction to Geometry*, 2nd ed. New York: Wiley, 1969.

[22] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed., Wiley Interscience, 2000.

[23] H. Farid and S. Lyu, "Higher-Order Wavelet Statistics and their Application to Digital Forensics," *IEEE Workshop on Statistical Analysis in Computer Vision*, Madison, WI, June 2003.

[24] H. Farid and S. Lyu, "Steganalysis using Higher-order Image Statistics," *IEEE Trans. on Information Forensics and Security*, Vol. 1, No. 1, pp. 111–119, March 2006.

[25] G. E. Farin, *Curves and Surfaces for Computer-aided Geometric Design: a Practical Guide*, 4th ed. Academic Press, 1997.

[26] J. Fridrich, "Image Watermarking for Tamper Detection," *Proceedings of IEEE International Conference on Image Processing (ICIP)*, Vol. 2, pp. 404–408, Chicago, IL, Oct 1998.

[27] J. Fridrich, M. Goljan, P. Lisonek, and D. Soukal, "Writing on Wet Paper,", SPIE Electronic Imaging, Security, Steganography, and Watermarking of Multimedia VII, Jan. 2005.

[28] J. Fridrich, M. Goljan, P. Lisonek, D. Soukal, "Writing on Wet Paper," *IEEE Trans. on Signal Processing, Supplement on Secure Media*, vol. 53, no. 10, Part 2, pp. 3923–3935, Oct. 2005.

[29] J. Fridrich, M. Goljan, and D. Soukal, "Perturbed Quantization Steganography with Wet Paper Codes," Proc. ACM Multimedia and Security Workshop, pp. 4-15, Sept. 2004.

[30] J. Fridrich, M. Goljan, and D. Soukal, "Efficient Wet Paper Codes," *7th International Workshop on Information Hiding*, pp.204–218, 2005.

[31] J. Fridrich, M. Goljan, and D. Soukal, "Wet Paper Codes with Improved Embedding Efficiency," *IEEE Trans. on Information Forensics and Security*, vol. 1, no. 1, pp.102–110, March 2006.

[32] M. S. Fu and O. C. Au, "Data Hiding Watermarking for Halftone Images", *IEEE Trans. on Image Processing*, April 2002

[33] R. G. Gann, *Desktop Scanners: Image Quality Evaluation*, Prentice-Hall, 1999.

[34] H. Gou, A. Swaminathan, and M. Wu, "Robust Scanner Identification based on Noise Features", *SPIE Conference on Security, Watermarking and Steganography*, Jan. 2007.

[35] H. Gou, A. Swaminathan, and M. Wu, "Noise Features for Image Tampering Detection and Steganalysis", *IEEE International Conference on Image Processing (ICIP)*, to appear, Sept. 2007.

[36] H. Gou and M. Wu, "Data Hiding in Curves for Collusion-resistant Digital Fingerprinting," *Proc. IEEE International Conference on Image Processing (ICIP)*, pp. 51–54, 2004.

[37] H. Gou and M. Wu, "Robust Digital Fingerprinting for Curves," *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Volume II, pp. 529–532, 2005.

[38] H. Gou and M. Wu, "Data Hiding in Curves with Application to Fingerprinting Maps," *IEEE Trans. on Signal Processing, Supplement on Secure Media*, vol. 53, no. 10, Part 2, pp. 3988–4005, Oct. 2005.

[39] H. Gou and M. Wu, "Fingerprinting Digital Elevation Maps", *SPIE Conference on Security, Watermarking and Steganography*, Jan. 2006.

[40] H. Gou and M. Wu, "Improving Embedding Payload in Binary Images with "Super-Pixels"", *IEEE International Conference on Image Processing (ICIP)*, to appear, Sept. 2007.

[41] C. Heegard and A. El-Gamal, "On the Capacity of Computer Memory with Defects," *IEEE Trans. Information Theory*, vol. 29, pp. 731-739, 1983.

[42] H. Z. Hel-Or, "Copyright Labelling of Printed Images," *Proc. of International Conference on Image Processing (ICIP)*, 1999.

[43] G. C. Holst, *CCD Arrays, Cameras, and Displays*, JCD Publishing and SPIE Optical Engineering Press, 1996.

[44] C-W. Hsu, C-C. Chang, and C-J. Lin, "A Practical Guide to Support Vector Classification," Available online at `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

[45] Z. Huang and F. Cohen, "Affine-invariant B-spline moments for curve matching," *IEEE Trans. on Image Processing*, vol. 5, no. 10, pp. 1473–1480, October 1996.

[46] A. K. Jain, *Fundamentals of Digital Image Processing.* Prentice Hall, 1989.

[47] D. Kacker and J. P. Allebach, "Joint Halftoning and Watermarking", *IEEE Trans. on Signal Processing*, April 2003.

[48] N. Khanna, A. K. Mikkilineni, G. T. C. Chiu, J. P. Allebach, and E.J. Delp, "Scanner Identification Using Sensor Pattern Noise," *SPIE Conference on Security, Watermarking and Steganography*, Jan. 2007.

[49] N. Khanna, A. K. Mikkilineni, G. T. C. Chiu, J. P. Allebach, and E.J. Delp, "Forensic Classification of Image Sensor Types," *SPIE Conference on Security, Watermarking and Steganography*, Jan. 2007.

[50] D. Kirovski, H.S. Malvar, and Y. Yacobi, "Multimedia Content Screening using a Dual Watermarking and Fingerprinting System,", *Proc. of ACM Multimedia*, 2002.

[51] K. H. Ko, T. Maekawa, N. M. Patrikalakis, H. Masuda and F.-E. Wolter, "Shape Intrinsic Fingerprints for Free-form Object Matching," *Proc. 8th ACM symposium on Solid modeling and applications*, pp. 196–207, 2003.

[52] E. Koch and J. Zhao, "Embedding Robust Labels into Images for Copyright Protection," *Proc. Int. Congr. Intellectural Property Rights for Specicalized Information, Knowledge and New Technologies*, 1995.

[53] I. Kweon, and T. Kanade, "Extracting Topographic Terrain Features from Elevation Maps," *Journal of Computer Vision, Graphics and Image Processing: Image Understanding*, Vol. 59, No. 2, pp. 171–182, March, 1994.

[54] C. L. Lawson and R. J. Hanson, *Solving Least-Squares Problems*, Prentice-Hall, 1974.

[55] J. J. Lee, N. I. Cho, and J. W. Kim, "Watermarking for 3D NURBS Graphic Data," *Proc. of IEEE International Workshop on Multimedia Signal Processing (MMSP)*, pp. 304–307, 2002.

[56] J. J. Lee, N. I. Cho, and S. U. Lee, "Watermarking Algorithms for 3D NURBS Graphic Data," *EURASIP Journal on Applied Signal Processing*, vol. 2004, no. 14, pp. 2142–2152, October 2004.

[57] Y. Liu, J. Mant, E. Wong, and S. H. Low, "Marking and Detection of Text Documents Using Transform-domain Techniques," *Proc. of SPIE, vol. 3657, Electronic Imaging (EI'99) Conference on Security and Watermarking of Multimedia Contents*, San Jose, CA, 1999.

[58] J. Lubin, J. A. Bloom, and H. Cheng, "Robust, Content-dependent, High-fidelity Watermark for Tracking in Digital Cinema," *SPIE Conference on Security and Watermarking of Multimedia Contents*, vol. 5020, pp. 536–545, 2003.

[59] M. Luby, "LT Codes," *Proc. The 43rd Annual IEEE Symposium on Foundations of Computer Science*, pp. 271-282, Nov. 2002.

[60] J. Lukas, J. Fridrich, and M. Goljan,"Determining Digital Image Origin Using Sensor Imperfections," *Proc. of SPIE Electronic Imaging*, Jan. 2005.

[61] J. Lukas, J. Fridrich, and M. Goljan, "Detecting Digital Image Forgeries Using Sensor Pattern Noise," *Proc. of SPIE Electronic Imaging*, Jan. 2006.

[62] J. Lukas, J. Fridrich, and M. Goljan,"Digital Camera Identification from Sensor Noise,", *IEEE Trans. on Information Security and Forensics*, Vol. 1, no. 2, pp. 205–214, June 2006.

[63] S. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 11, no. 7, pp. 674–693, July 1989.

[64] K. Matsui and K. Tanaka, "Video-steganography: How to Secretly Embed a Signature in a Picture," *Proc. IMA Intellectural Property Project*, vol. 1, no. 1, 1994.

[65] N. F. Maxemchuk and S. Low, "Making Text Documents," *Proc. IEEE International Conf. on Image Processing (ICIP)*, 1997.

[66] Q. Mei, E.K. Wong, and N. Memon, "Data Hiding in Binary Text Documents," Proc. of SPIE, Jan 2001.

[67] M. K. Mihcak, I. Kozintsev, K. Ramchandran, and P. Moulin, "Low-Complexity Image Denoising Based on Statistical Modeling of Wavelt Coefficients," *IEEE Signal Processing Letters*, Vol. 6, no. 12, pp. 300–303, Dec. 1999.

[68] T.-T Ng, S.-F. Chang, J. Hsu, and M. Pepeljugoski, "Columbia Photographic Images and Photorealistic Computer Graphics Dataset," Technical Report 205-2004-5, ADVENT, Columbia University, 2004.

[69] R. Ohbuchi, H. Masuda, and M. Aono, "A Shape-preserving Data Embedding Algorithm for NURBS Curves and Surfaces," *Proc. of CGI*, pp. 180–187, 1999.

[70] R. Ohbuchi, H. Ueda, and S. Endoh, "Watermarking 2D Vector Maps in the Mesh-spectral Domain," *Proc. of the Shape Modeling International (SMI)*, 2003.

[71] H. Pan, Y. Chen, and Y. Tseng, "A Secure Data Hiding Scheme for Two-Color Images", *IEEE Sympt. on Computers and Communications*, 2000.

[72] S. C. Pei and J. M. Guo, "Hybrid Pixel-based Data Hiding and Block-based Watermarking for Error-diffused Halftone Images," *IEEE Trans. on Circuits and Systems for Video Technology*, Aug. 2003.

[73] A.C. Popescu and H. Farid, "Exposing Digital Forgeries in Color Filter Array Interpolated Images," *IEEE Trans. on Signal Processing*, Vol. 53, no. 10, pp. 3948–3959, 2005.

[74] E. Praun, H. Hoppe, and A. Finkelstein, "Robust Mesh Watermarking," *SIGGRAPH 1999, Computer Graphics Proceedings*, pp. 49–56, 1999.

[75] E. Simoncelli, and E. Adelson, "Noise Removal via Bayesian Wavelet Coring," *Proc. IEEE Internation Conference Image Processing (ICIP)*, Vol. 1, pp. 379–382, 1996.

[76] V. Solachidis and I. Pitas, "Watermarking Polygonal Lines Using Fourier Descriptors," *IEEE Computer Graphics and Applications*, vol. 24, no. 3, pp. 44–51, May/June 2004.

[77] A. Sole, V. Caselles, G. Sapiro, and F. Arandiga, "Morse Description and Geometric Encoding of Digital Elevation Maps," *IEEE Trans. on Image Processing*, vol. 13, no. 9, pp. 1245–1262, Sept. 2004.

[78] H. Stone, "Analysis of Attacks on Image Watermarks with Randomized Coefficients," *Technical Report 96-045, NEC Research Institute*, 1996.

[79] A. Swaminathan, Y. Mao, and M. Wu, "Robust and Secure Image Hashing," *IEEE Trans. on Information Forensics and Security*, June 2006.

[80] A. Swaminathan, M. Wu, and K. J. Ray Liu, "Component Forensics of Digital Cameras: A Non-Intrusive Approach," *Proceedings of Conference on Information Sciences and Systems (CISS)*, March 2006.

[81] A. Swaminathan, M. Wu, and K. J. Ray Liu, "Non-Intrusive Forensic Analysis of Visual Sensors Using Output Images," *IEEE Conference on Acoustic, Speech and Signal Processing (ICASSP)*, May 2006.

[82] A. Swaminathan, M. Wu, and K. J. Ray Liu, "Image Tampering Identification using Blind Deconvolution," *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, October 2006.

[83] W. Trappe, M. Wu, Z.J. Wang, and K.J.R. Liu, "Anti-collusion Fingerprinting for Multimedia," *IEEE Trans. on Signal Processing*, vol. 51, no. 4, pp. 1069–1087, April 2003.

[84] B.S. Tsybakov and A.V. Kuznetsov, "Coding in a Memory with Defective Cells," *Probl. Inform. Transmission*, vol. 10, pp. 132-138, 1974.

[85] Z.J. Wang, M. Wu, H. Zhao, W. Trappe, and K.J.R. Liu, "Collusion Resistance of Multimedia Fingerprinting using Orthogonal Modulation," *IEEE Trans. on Image Processing*, vol. 14, no. 6, pp. 804–821, June 2005.

[86] A. Westfeld, "F5–A Steganographic Algorithm: High Capacity Despite Better Steganalysis," *Proc. of Intl. Workshop on Info. Hiding*, Pittsburgh, PA, April 2001.

[87] M. Wu, J. Fridrich, M. Goljan and H. Gou"Handling Uneven Embedding Capacity in Binary Images: A Revisit", *Proc. of SPIE Conference on Security, Watermarking and Steganography*, Jan. 2005.

[88] M. Wu and B. Liu: "Watermarking for Image Authentication", *Proc. IEEE International Conf. on Image Processing (ICIP)*, vol.2, pp. 437–441, Oct. 1998.

[89] M. Wu and B. Liu, "Data Hiding in Image and Video: Part-I Fundamental Issues and Solutions," *IEEE Trans. on Image Processing*, vol. 12, pp. 685–695, June 2003.

[90] M. Wu and B. Liu, "Data Hiding in Binary Image for Authentication and Annotation," *IEEE Trans. on Multimedia*, Aug. 2004.

[91] M. Wu, E. Tang, and B. Liu, "Data Hiding in Digital Binary Image," *Proc. of IEEE International Conference on Multimedia and Expo (ICME)*, pp. 393–396, 2000.

[92] M. Wu, W. Trappe, Z.J. Wang, and K.J.R. Liu, "Collusion Resistant Fingerprinting for Multimedia," *IEEE Signal Processing Magazine, Special Issue on Digital Rights Management*, vol. 21, no. 2, pp. 15–27, March 2004.

[93] M. Xia and B. Liu, "Image Registration by 'Super-curves'," *IEEE Trans. on Image Processing*, vol. 13, no. 5, pp. 720–732, May 2004.

[94] L. Xie, G. R. Arce and R. F. Graveman, "Approximate Image Message Authentication Codes," *IEEE Trans. on Multimedia*, Vol. 3, No. 2, pp. 242-252, June 2001.

[95] M.M. Yeung and F. Wintzer, "An Invisible Watermarking Technique for Image Verification," *Proc. IEEE International Conference on Image Processing (ICIP)*, Oct. 1997.

[96] R. Zamir, S. Shamai, and U. Erez, "Nested Linear/Lattice Codes for Structured Multiterminal Binning," *IEEE Trans. Inform. Theory* 48(6), pp. 1250-1276, 2002.

[97] H. Zhao, M. Wu, Z.J. Wang, and K.J.R. Liu, "Forensic Analysis of Nonlinear Collusion Attacks for Multimedia Fingerprinting," *IEEE Trans. on Image Processing*, vol. 14, no. 5, pp. 646–661, May 2005.