HIDING INFORMATION INSIDE STRUCTURED SHAPES

Samarjit Das[*]
Iowa State University
Ames, IA

Shantanu Rane, Anthony Vetro
Mitubishi Electric Research Laboratories
Cambridge, MA

ABSTRACT

This paper describes a new technique for embedding a message within structured shapes. It is desired that any changes in the shape owing to the embedded message are invisible to a casual observer but detectable by a specialized decoder. The message embedding algorithm represents shape outlines as a set of cubic Bezier curves and straight line segments. By slightly perturbing the Bezier curves, a single shape can spawn a library of similar-looking shapes each corresponding to a unique message. This library is efficiently stored using Adaptively Sampled Distance Fields (ADFs) which also facilitate rendering of the modified shapes at the desired resolution and fidelity. Given any modified shape, a forensic detector applies Procrustes analysis to determine the embedded message. Results of an extensive subjective test confirm that the shape modifications are indeed unobtrusive. Further, to test the recovery of the message bits in noisy physical environments, a text document is put through a print-photocopy-scan process. Message recovery is found to be stable even after multiple rounds of photocopying.

***Index Terms*—** Data Hiding, Information Forensics, Bezier Curve, Procrustes Distance

1. INTRODUCTION

Our interest in the field of information embedding stems from the challenges involved in embedding data inside structured graphics, particularly in text characters. Unlike the case of natural images, where information may be invisibly hidden inside higher spatial frequencies, human eyes are finely conditioned to detect very minute changes in the shape of text characters. Hence there is little room to make changes in the text without the changes becoming visible to a human observer. Additionally, as the document folds, or shears, or is photocopied or scanned, the accompanying degradation may completely overwhelm a small modification, rendering it undetectable. A data hiding algorithm must be robust to such degradations if it has to be utilized in vital applications such as authentication, copyright protection, forensics, and so on.

There exist a number of schemes which embed information into the background of a printed document, for example [1]. To make the embedding unobtrusive, these changes are necessarily very light and thus may not survive photocopying. Methods to embed data by modulating the distance between successive lines of text [2] or words [3] are robust to noise but have very low embedding capacity because the modifications become very obtrusive at high embedding rates. Message bits can also be embedded into individual text characters by modulating their grayscale values or halftone patterns [4]. However, messages embedded in this way cannot survive photocopy operations. Recently, embedding schemes based on explicitly modifying the shapes of the text characters have been proposed. For example, [5] describes an extrinsic printer forensics technique which a printer inserts tiny sinusoidal patterns at the boundaries of text characters. At small font sizes, these patterns are invisible, but their robustness to photocopying has not been reported. Concurrently, a similar scheme [6] was proposed to modify the vertical edges of text characters. The modifications in this scheme are robust to two rounds of photocopying but may not always be unobtrusive.

This work takes a different approach to shape modifications that achieves both imperceptibility and robustness to noise. Although we were motivated by data hiding inside text documents, the method described herein generalizes to any multi-dimensional structured graphics such as stroke-based Japanese and Chinese fonts, cartoons, holographic logos and so on. In our work, the modifications are made to the shape outline as this enables tighter control over the type and extent of the modification. This is based on the fact that the outline is an inherent characteristic which retains information under many degradations encountered in nature. The embedding scheme is described in Section 2. At the decoder, Procrustes shape analysis has been used to discriminate between shapes and thereby determine the message, as described in Section 3. Results on the subjective invisibility of the modifications, and the robustness of the embedded message to noise are provided in Section 4.

2. EMBEDDING VIA PERTURBED BEZIER CURVES

The aim is to embed M symbols inside a given shape where each symbol is chosen from a finite alphabet set $\{0, 1, ..., K-1\}$. To accomplish this, it is necessary to have an automatic procedure for generating K^M modified versions of the given shape. We assume that M and K are known to the embedder. Determining M and K based on the inherent complexity of a given shape is a very interesting theoretical problem in itself, but it is outside our current scope. The attribute "structured" implies that the shape can be stored inside a data structure, such as one based on splines, wavelets or distance fields. The embedding and extraction algorithms in this paper are independent of the actual data structure. Nevertheless, we exploit a particular data structure in our implementation, namely an Adaptively Sampled Distance Field (ADF). A detailed discussion of ADFs may be obtained from [7]; at this stage, it is sufficient to know that an ADF allows a shape to be stored efficiently in a tree-based representation and enables rendering of the shape at various resolutions. ADF is a natural choice because it is a widely used technology for storing and rendering fonts on personal computers. The steps in the embedding a message are described below. Here, the message contains "raw" information bits and may optionally contain error correction symbols.

1. Automatic Landmark Extraction: If the shape is already stored as an ADF, its outline is available directly. If the shape is in the form of an image, the outline is obtained via edge extraction. Then, the discrete curvature profile of the entire outline contour is obtained and a curvature threshold is used to locate the points of high curvature. These points are assigned as the primary landmarks (Fig. 1)

[*]This work was performed while S. Das was an intern at MERL.

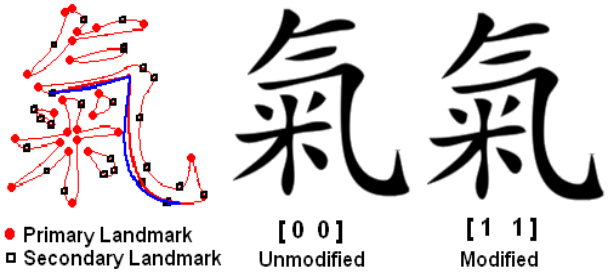


Fig. 1. Landmark points drive the selection of candidate slots for embedding bits inside an arbitrary shape. The outline segments in two chosen data slots are subtly deformed as shown by the dark lines. Each slot stores a single bit, resulting in four possible versions of the shape. Two of these versions, corresponding to message [0 0] and message [1 1] are shown. Observe that the rendered shapes are very subtly different from one another at the locations of the data slots.

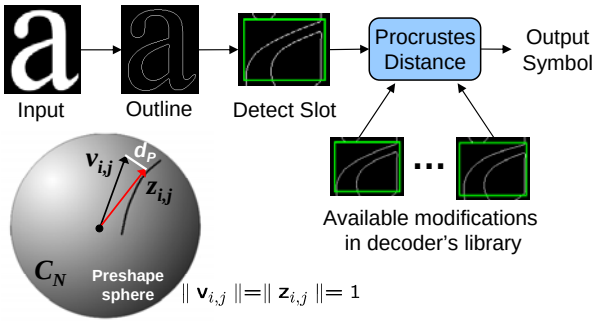


Fig. 2. Steps involved in locating the i^{th} data slot and determining the symbol embedded in it. $\mathbf{v}_{i,j}$ is the preshape corresponding to a database segment in the i^{th} data slot and $\mathbf{z}_{i,j}$ is the preshape obtained from input configuration, $\mathbf{s}_i$ after rotation normalization w.r.t $\mathbf{v}_{i,j}$. The Procrustes distance is $d_P = ||\mathbf{z}_{i,j} - \mathbf{v}_{i,j}||_2$.

and serve as primary visual cues for the structured shape. Then, via polygonal approximation of the outline segment between two primary landmark points, a set of secondary landmark points is obtained. The set of primary and secondary landmark points gives a coarse representation of the shape.

2. Determination of Candidate Data Slots: In this step, the algorithm traverses the outline to determine whether a symbol can be embedded in the segment between any two landmark points. A segment is chosen as a data slot if it satisfies the following heuristic imperceptibility constraints: (a) It should be a Bezier curve segment only. No straight line segment is used as a data slot because subjective tests have shown that deformations in straight line segment are easily perceptible; (b) A corner point cannot be in the interior of a data slot, only at the edge. Any significant modification at the corner point might lead to perceptibility; (c) The segment length is within a certain specified range $[l_{min}\ l_{max}]$ and the overall height and width of the shape is kept unchanged. Of all the segments that satisfy the above constraints, M are chosen at random for embedding.

3. Data Slot Modification: The candidate segments from the above step are now modified in any one of K ways. For imperceptibility, the modifications should be smooth and gradual along the curved segment. Additionally, the modification should not introduce any C0 or C1 discontinuities[1]. The total number of control points in the modified shape must be the same as those in the unmodified shape. For example, consider the following modification technique for inserting a bit in a data slot: The curved segment is protruded along its normal with maximum protrusion halfway through the segment. To prevent the introduction of discontinuities, the modifications are minimal near the edges of the segment (Fig. 1). The deformation along the curve follows a Gaussian profile with its mode located halfway through the curve and edges located at -3σ and $+3\sigma$. This *Gaussian deformation* is used for embedding bits inside structured shapes. In this case, a modification to a data slot represents a 1-bit while no modification represents a 0-bit. A wide spectrum of modifications are possible depending on the imperceptibility desired and the sophistication of the detection algorithm.

4. Perturbed Outline: Bezier curves are fitted to the modified curved segments in the candidate slots. If there are M data slots in a shape, then each unique combination of M symbols is thus associated with a unique modified outline. A single shape has thus spawned K^M modified versions.

5. Storage and Rendering via ADFs: The outline of each modified shape is now represented as a connected set of cubic Bezier curves and line segments. Each line segment is trivially represented by the locations of its landmark endpoints. Each Bezier curve segment is represented by the locations of four control points along the segment. The coordinates of all control points collectively constitute the *outline path information* of that particular shape. This path information is stored in an ADF. Note that the path information is given with respect to a normalized coordinate frame with $[X_{min}\ X_{max}\ Y_{min}\ Y_{max}] = [0\ 1\ 0\ 1]$ and the ADF module maps this coordinate frame to the image grid when the shape is to be rendered.

[1]A C0 discontinuity is a break in the outline. A C1 discontinuity is a discontinuity in the tangent, i.e., a kink in the curve.

3. DECODING VIA PROCRUSTES ANALYSIS

The goal of the decoder is to recover all M symbols embedded in a modified shape. For example, this shape would be that of a text character extracted from an electronic document, or a printed-photocopied-scanned page. Non-blind detection is assumed, i.e., the decoder also possesses a helper library of candidate data slots in a given noiseless and unmodified shape, modified versions of those data slots, and message symbols associated with each modified shape. The decoding steps (See Fig. 2) are described below:

1. Segmentation and Registration: The first step in the decoding process is to obtain a segmentation containing the input shape. It is assumed that the decoder knows the identity of the underlying unmodified shape via, for example, a rudimentary Optical Character Recognition (OCR) algorithm available in most contemporary scanners. Since the modifications described in Section 2 are very small, OCR is found to work equally well for modified and unmodified characters. The next step is to register the cropped shape image to the original unmodified shape image in the decoder's database. In this way, the input shape is mapped onto a pixel grid corresponding to a normalized coordinate frame.

2. Extraction of Potential Data Slots: Edge detection or outline contour extraction is performed on the registered shape image. Since the identity of the underlying shape is known, the decoder also knows which portions of the outline are candidate data slots containing message symbols. The remainder of the outline is of no use in message decoding. Therefore, it suppresses all edge elements except those in the approximate neighborhood of each candidate data slot. Using the helper database, a mask is created around the approximate location of each candidate data slot and all outline contours outside this mask are discarded. A standard contour-tracing algorithm, based on 8-connected neighborhoods, is used to extract the possible outline segment within the masked area. The tracing is initiated at the edge

pixel which is closest to the predicted location of the start of the data slot. After a contour segment is traced out, its length is compared to that of the segment in the unmodified data slot. If the lengths are outside the allowable $[l_{min}\ \ l_{max}]$ interval from Section 2, the decoder keeps looking for other contour segments of the correct length within the masked region. If a valid curve segment is traced out for a given data slot, it proceeds to decode the symbol embedded therein.

3. Mapping into N-dimensional unit hypersphere: A Bezier curve is fit to the extracted contour segment. The shape of this curve segment must now be compared with the shapes of corresponding curve segments extracted from the database of all modified data slots. To do this, the curve is uniformly sampled at N discrete points along its length to give a N-dimensional vector $\mathbf{s}_i$, $i = 1, 2, .., M$. The vector $\mathbf{s}_i$, is mapped to a unit hypersphere, called a preshape sphere, via translation and scale normalization as follows:

$$\mathbf{y}_i = C_N\,\mathbf{s}_i \quad \text{and} \quad \mathbf{w}_i = \frac{\mathbf{y}_i}{||\mathbf{y}_i||}$$

where $C_N = I_N - 1_N 1_N^T/N$, I_N is an $N \times N$ identity matrix and 1_N is a column vector of ones. $\mathbf{w}_i$ is called the preshape of the i^{th} data slot in the input shape, and resides on a unit hypersphere. Similarly, the decoder obtains the preshapes $\mathbf{v}_{i,j}$, $j = 0, 1, ..., K-1$ of the i^{th} data slot in the database of modified shapes.

4. Procrustes Distance Calculation: The preshape $\mathbf{w}_i$ corresponding to data slot i is rotation-normalized with respect to each of the possible database segments $\mathbf{v}_{i,j}$, $j = 0, 1, ..., K-1$ in data slot i to obtain $\mathbf{z}_{i,j} = \mathbf{w}_i e^{j\theta_{i,j}}$ where $\theta_{i,j} = angle(\mathbf{w}_i^* \mathbf{v}_{i,j})$. After rotation normalization in the preshape space, the Procrustes distance [8], as shown in Fig. 2 between a database segment in the i^{th} data slot and the input configuration $\mathbf{s}_i$ is given by

$$d_P(\mathbf{v}_{i,j}, \mathbf{s}_i) = ||\mathbf{z}_{i,j} - \mathbf{v}_{i,j}||_2$$

5. Determination of Embedded Message: The decoded message symbol $b_i \in \kappa = \{0, 1, ..., K-1\}$ in the i^{th} data slot is then given by:

$$b_i = \arg\min_{\forall j \in \kappa} d_P(\mathbf{v}_{i,j}, \mathbf{s}_i)$$

In the most general case of message decoding, the sequence $\{b_i\}$, $i = \{1, 2, ..., M\}$ gives the message embedded in the entire shape. In an important special case, only one bit is embedded inside each shape. In this case, the same bit (0 or 1) is embedded in all M data slots. This bit is recovered by majority voting over the M data slots, simulating a rudimentary ECC.

4. EXPERIMENTAL RESULTS

4.1. Examples

Unlike previous work in text data hiding, the automated embedder was designed to work with any structured shape including Latin-based and stroke-based fonts and any other arbitrary shapes. Depending on the number of candidate data slots, many subtly modified versions were rendered, a small sampling of which appears in Fig. 3. In another example shown in Fig. 4, each English or Japanese text character was restricted to have a maximum of two candidate data slots and one bit was embedded per lowercase character. It is difficult to perceive a difference between the original and modified documents. We verify this imperceptibility in our subjective tests to be discussed in next subsection. Since the modified shapes are stored using ADFs, our algorithm can be incorporated into standard word processing software to generate documents with imperceptibly hidden messages. For a given font type, each glyph may have up to K^M slightly different versions stored as ADFs [7] in the font library. The appropriate version is then rendered onto the page depending on the desired bit combination. It is straightforward to adapt the decoder to cases in which the number of embedded symbols, M, is different for different shapes.

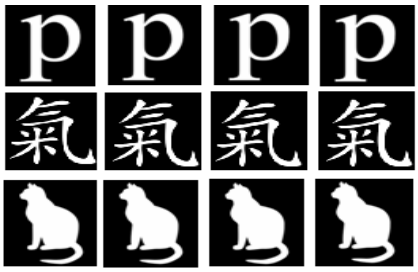
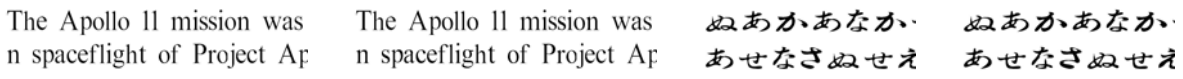


Fig. 3. Examples of embedding message inside English text (2 bits/shape), traditional Chinese Kanji (3 bits/shape) and a simple cartoon (4 bits/shape). Each rendered version represents a unique hidden bit combination. Upon close observation, the back and chest of the cat will be found to protrude outwards in some versions. Very close inspection of the 'p' and the Kanji symbol will indicate similar subtle modifications.

4.2. Subjective Tests for Imperceptibility

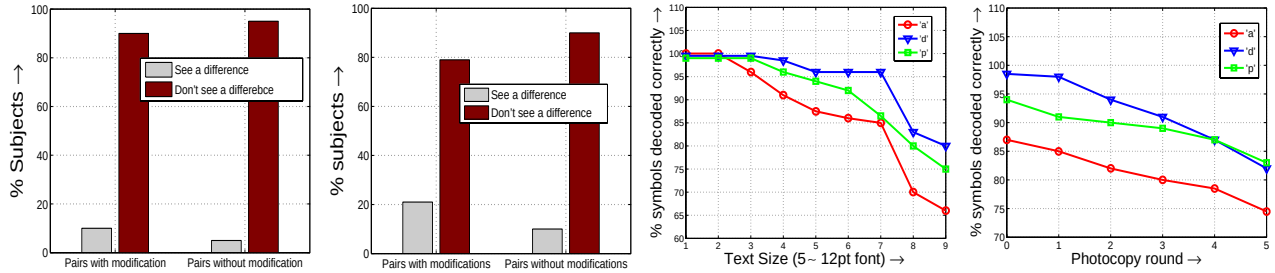
The data set for the subjective test was divided in four parts (a) Paragraphs of English texts with different contents at 4 font sizes ranging from 11 pt to 24 pt Times New Roman. Each lower-case character contained one embedded message bit that was equally likely to be 0 or 1. (b) Pages of Japanese Kana characters with different contents at 4 font sizes similar to in Fig. 4. Each Kana contained one embedded message bit which is 0 or 1 with equal probability. (c) 4 different traditional Chinese Kanji characters with 4 embedded message bits per character and (d) 4 different arbitrary structured shapes with 4 embedded message bits per character. All these 16 shapes were printed out on white paper at 800 dpi. The tests were performed over a total of 60 subjects. Each subject was provided with 32 sample pairs. Of these 16 pairs consisted of identical content but with the shapes slightly modified. The remaining 16 pairs consisted of shapes which were in fact identical, to serve as a control experiment for user biases. The subjects were asked to answer a simple question for each of the 32 pairs: Do you see a difference between the shapes in a pair?

Owing to space limitations, it is difficult to describing the full results of the subjective tests. Therefore, we will concentrate on the results for English and Japanese text, which are shown in Fig. 5(a) and Fig. 5(b) respectively. For English text of all sizes, 10% of the testers reported seeing a difference when a difference actually existed. However, 5% of the testers reported a difference when, in fact, there was none. Thus, on an average, the modifications were uniquely imperceptible to about 95% of the testers. Interestingly, for the Japanese text, this number was about 90%. Since the testers were predominantly English speakers, they approached the Japanese Kana as shapes rather than meaningful text. It appears that the act of associating some meaning to shapes seems to prevent a tester from seeing the differences. For traditional Chinese Kanji characters with only one very large character per page, the modifications were imperceptible to 78% of the testers. For the category of arbitrary shapes with one very large shape per page, the modifications were imperceptible to 70% of the testers. These numbers provide information not only on the kinds of modifications that are visible to casual observers, but also enable us to determine the number and extent of the modifica-



(a) Original English (b) Modified English (c) Original Kana (d) Modified Kana

Fig. 4. One bit is embedded per character in an English paragraph (lowercase letters only) and in a collection of Japanese Kana. The modifications are along smooth curved outlines making them difficult to discern on casual reading.



(a) Subjective test : English (b) Subjective test : Hiragana (c) Decoding performance (d) Decoding after Photocopy

Fig. 5. (a) and (b) When user biases are accounted for by a control experiment consisting of identical shapes, it is observed that changes in English text are perceived by only about 5 % of the testers, while those in Japanese Hiragana are perceived by about 10 %. (c) The decoding performance deteriorates when the font size is made smaller. Size 1, size 5 and size 9 correspond to 36 pt, 12 pt and 8 pt respectively in Times New Roman. (d) Message recovery degrades gradually with multiple rounds of photocopying, but even after 5 rounds of photocopying, about 80 % of the message bits are still recoverable.

tions. Note that this test measures only whether a user can perceive a difference in a side-by-side comparison of shapes. Indeed, in the absence of a reference, it is highly unlikely that an observer can detect any changes in shape. Morever, perceiving a difference does *not* imply that the user is able to decode the message bits. Even if the embedding/decoding algorithm is available to a potential attacker, the actual message is protected by a secret key shared only by the embedder and decoder.

4.3. Message Recovery in the Presence of Noise

Message recovery was tested for both electronic and print-scanned documents. We report only the results for English text here. For electronic documents, the decoder was given a JPEG image (quality : 90%) of the document with 1 message bit embedded per letter. The message bits were generated by a Bernoulli-0.5 distribution.

For electronic documents with a maximum of one bit per character, the decoder could recover 100% of the message bits at font sizes of 12-point and above in the Times New Roman (TNR) font. For 11-point TNR font, the performance dropped slightly to about 96%. Even for sizes as small as 16×16 pixel images of the characters, 90% of the message bits are recovered. These numbers guide the selection of the code rate for an error correcting code that can be applied to the embedded message prior to embedding.

When the decoder was provided with print-scanned versions of the above documents with 11 point TNR font, the message recovery was about 87%. To examine the effect of font size on decoding performance, a page was generated with the characters 'a' repeated across the page and a Bernuolli-0.5 message was embedded on the page with 1 bit per 'a'. Similar experiments were conducted on pages with characters 'd' and 'p'. Of the font sizes tested, size 1, 5, 9 correspond to TNR font sizes 36 pt, 12 pt and 8 pt respectively.

Next, we tested the system's robustness to photocopying. Up to 5 rounds of photocopying were performed for text with 12-point TNR font before the document was scanned and the message was decoded. As shown in Fig. 5(d), message recovery degrades quite slowly w.r.t the number of rounds of photocopying. Even after 5 rounds, about 80% of the bits were decoded correctly.

5. REFERENCES

[1] T. Sugai, U. Shimmyo, H. Ito, and M. Suzuki, "Development of a watermarking method for printed documents," in *70th Convention of the Information Processing Society of Japan, Tsukuba, Japan*, March 2008.

[2] J. T. Brassil, S. Low, and N. F. Maxemchuk, "Copyright protection for the electronic distribution of text documents," *Proc. of the IEEE*, , no. 7, pp. 1181–1196, July 1999.

[3] D. Huang and H. Yan, "Interword distance changes represented by sine waves for watermarking text images," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, no. 12, pp. 1237–1245, December 2001.

[4] R. Villan et al., "Text data-hiding for digital and printed documents: Theoretical and practical considerations," *SPIE-IST Electronic Imaging*, pp. 15–19, January 2006.

[5] P.-J. Chiang, A. Mikkilineni, E. Delp, J. Allebach, and G. Chiu, "Extrinsic signatures embedding and detection in electrophotographic halftone images through laser intensity modulations," in *Proc. IS and T NIP22: Intl. Conf. Digital Printing Technologies*, Denver, CO, September 2006, pp. 432–435.

[6] A. Varna, S. Rane, and A. Vetro, "Data hiding in hard-copy text documents robust to print, scan and photocopy operations," in *IEEE Intl. Conf. Acoustics, Speech, Sig. Proc. (ICASSP)*, April 2009.

[7] S.F. Frisken, R.N. Perry, A.P. Rockwood, and T.R. Jones, "Adaptively sampled distance fields: A general representation of shape for computer graphics," in *ACM SIGGRAPH*, July 2000, pp. 249–254.

[8] I.L. Dryden and K.V. Mardia, *Statistical Shape Analysis*, John Wiley and Sons, 1998.