Augmented Handwritings by Data-Embedding Pen

Akira YOSHIDA†, Marcus LIWICKI††, Seiichi UCHIDA†††, Masakazu IWAMURA††††,
Shinichiro OMACHI†††††, and Koichi KISE††††

† Graduate School of Systems Life Sciences, Kyushu University
††† Faculty of Information Science and Electrical Engineering, Kyushu University
Motooka 744, Nishi-ku, Fukuoka-shi, 819-0395 Japan
†† Knowledge Management, DFKI Trippstadter Str. 122, D-67669 Kaiserslautern, Germany
†††† Graduate School of Engineering, Osaka Prefecture University Sakaishi,Osaka, 599-8531 Japan
††††† Graduate School of Engineering, Tohoku University Sendai-shi, 980-8579 Japan
E-mail: †yoshida@human.ait.kyushu-u.ac.jp

Abstract In this paper, we propose a data-embedding pen, which adds an ink-dot sequence along a handwritten pattern during writing. The ink-dot sequence represents some meta-information, such as writer's name, date of writing, and URL. The meta-information can be extracted from the handwritten pattern by image processing techniques and a stroke recovery technique. Consequently, we can augment the handwritten pattern by the data-embedding pen to carry arbitrary information.

Key words data embedding, handwritings

1. Introduction

Handwriting is one of the most important methods for writing down information, making annotations, or just marking. Unfortunately, many information known during writing is already lost as soon as the ink is on the paper. In other words, we cannot access meta-information about the handwritten pattern from itself. For example, it is impossible to retrieve who wrote this pattern or when it was written. Consequently, a handwritten pattern on physical paper is just an ink pattern and it cannot represent any information but its shape.

We aim to realize a novel pen device which adds an information to handwriting on physical paper. The pen device, called *data – embeddingpen* [1], can embed meta-information by an additional ink-dot sequence along the ink stroke of the handwriting. Each ink-dot represents an information bit and thus an ink-dot sequence represents a bit-stream of the embedded information.

2. The Data-Embedding Pen

The data-embedding pen is a device which comprises a usual ballpoint pen and an ink-jet nozzle element. Figure 1 shows this device. The nozzle produces small ink-dots along handwritten stroke during writing. The number of ink-dots, size/length of ink-dots, and their timing represents the embedded information. In this paper, we used yellow ink for those dots, while it is also possible to use some invisible ink.

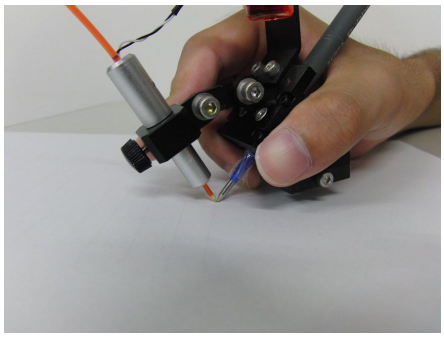

Figure 1 The data-embedding pen

3. Information Embedding

As stated above, the embedded information is represented by an ink-dot sequence. The nozzle used in this paper is able to generate up to 2,000 ink-dots per second. Using this high frequency, we can form a connected line by a continuation of several ink-dots. Hereafter, a line by n continuous ink-dots is called $n-pulseline$. Figure 2 shows an example of handwritten pattern with ink-dots.

Our cording scheme is based on the combination of three different n-pulse lines. Specifically, we use n =1 (a dot), 5 (a short line), and 20 (a long line). The ink-dots sequence of Fig.2 consists of those n-pulse lines. The information is con-

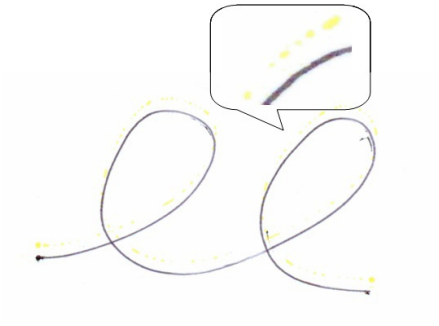

Figure 2 Example of handwriting pattern

verted into binary(0 and 1) sequence and embedded by using the 1-pulse line as 0 and the 5-pulse line as 1. A short pause is inserted between each bit information (1-pulse or 5-pulse line) like in the Morse code. The 20-pules line, hereafter called *synchronization blob*, is used as an marker to take synchronization of information recovery.

Our coding scheme is composed of three units, called *frame*, *block*, and *bit*. The bit is the smallest unit of information and defined by a 1-pulse line or 5-pulse line as noted before. Several consecutive bits comprise a block and several consecutive blocks comprise a frame. A pause which is longer than the pause between bits is inserted between two consecutive blocks. Each flame begins with a synchronization blob.

4. Information Recovery

4.1 Image Processing

In order to retrieve the embedded information, we extract ink-dots and black ink strokes from a scanned image. In the this section, four steps of image processing are explained.

The first step of ink-dot extraction is a simple thresholding operation to extract the black ink stroke and yellow ink-dots. The second step is noise removal because the black ink stroke image extracted includes many noisy pixels. Thus, we apply erosion and dilation to the black ink stroke image. Similar operations are also applied to the ink-dot image. The third step is a special treatment of ink-dots occluded by the black ink stroke. The fourth step is a thinning operation on the black ink stroke. Figure 3 shows the result of image processing.

4.2 Aligning Ink-Dots by Stroke Recovery

In order to decode the ink-dots, they should be aligned according to their original temporal order. Since this order is lost in the scanned image, we must estimate it by using the result of stroke recovery. Specifically speaking, after recovering the writing order of the black ink stroke based on the algorithm presented in [2] and establishing the correspondence between the ink-dots and the stroke, we align the ink-dots.

4.3 Data Decoding

For decoding, the bit information (i.e., 1-pulse and 5-pulse

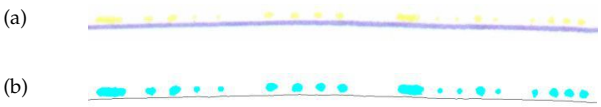

Figure 3 (a)Ink-dots (yellow) nearby a handwriting stroke (black). (b)After image processing.

lines and synchronization blob) is first recovered at every ink-dot, just by checking its size/length. The sequence is separated into frames using the synchronization blobs. Larger gaps which detected within each frame are assumed as the gaps between block.

Next, a plausibility control is performed on the extracted data. For each block, the number of bits is confirmed. Sometimes a block has spurious bits, resulting from a wrong mapping or just from noise. In this case, those adjacent bits whose distance deviates too much from the mean distance are deleted. If the number of bits (blocks) does not correspond to the values the number of bits per block (blocks per frame), the frame is rejected.

5. Error Correction

The process of embedding the ink next to the handwritten stroke is often accompanied with several errors. Especially, it is a big issue that an ink-dot sometimes overlaps with the next ink-dot because of slow pen-movement. Moreover, the black ink sometimes overlaps with the ink-dots.

In order to recover from the errors, in this paper we use Reed-Solomon error correction [3] for reliably recovering from the errors. For the Reed-Solomon error correction, it is not needed to recover all frames correctly as long enough frames are present. Because, it contains data of frame position, and it can recover the entire information from only several frame. In this paper, we design each frame to be comprised of two blocks, the first block for the position of the frame and the second block for its value. Each block contains 4 bit. Naturally, each frame consists of 8 bit; 4 bit for the frame position and 4 bit for its value. Hereafter, k represents the number of frames. More details of Reed-Solomon codes can be found in [3].

6. Experiments and Results

6.1 Data

We collected data-embedded handwritings using the current pen prototype. The dataset contains 50 horizontal straight lines with a length of 5 cm or 10 cm. All lines have been drawn with approximately the same velocity.

For the Reed-Solomon encoding, we used the Shifra Open Source error correcting code library. The code length was fixed to 15 frames.

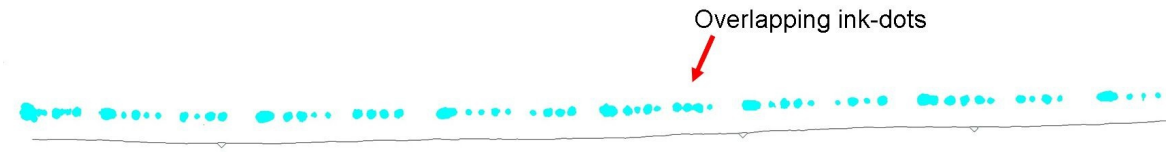

Figure 4 Extraction result (after thinning) of a 5 cm long line (enlarged).

Table 1 Percentage (%) of correctly recovered information

# frames (k)	# bits	5 cm	10 cm
1	4	100	100
4	16	100	100
5	20	64	100
6	24	36	100
7	28	16	100
8	32	4	100
9	36	0	100
10	40	0	86
11	44	0	50
12	48	0	26
13	52	0	6
14	56	0	0

6.2 Result

We focus on how much information can be embedded in straight lines. In this task only rarely some decoding errors occurred on the frame level, since there are no crossing. (Figure 4 provides an example for the extraction result of a 5 cm straight line where no error occurred.) The only problem was some overlapping ink-dots due to a slow pen-movement. Note that these frames were rejected during the frame decoding step presented in Section 4. 3, because it resulted in missing points for the Reed-Solomon error correction.

Table 1 shows the results of the experiments. This table shows the percentage of samples where the entire information could be correctly recovered by using the Reed-Solomon error correction. For $k \leqq 4$, the entire information was always correctly recovered even for straight lines as short as 5 cm. For larger k value, the performance decreases, because only a limited number of frames appear in a 5 cm line. (In fig. 4, for example, 6 frames appear.)

Some errors were corrected by the Reed-Solomon code. In Fig. 4, some ink-dots are overlapping. (For example, the third block from the right is overlapping.) In this case, 5 blocks were correctly recovered. If $k \leqq 5$, the entire information can be correctly recovered by using the Reed-Solomon error correction.

7. Conclusions

In this paper, we have presented results of the experiment on a data-embedding pen, which can add meta-information to handwritten patterns. The meta-information represent the date of writing, the writer ID, and URL. The main idea is to encode the desired information in an ink-dot sequence plotted nearby handwriting strokes.

We proposed the use of the Reed-Solomon error correction scheme for reliable encoding and recovering the meta-information. We can have correct recovery results by the error correction results, even if we have several erroneous ink-dots due to noises on image processing.

In our experiments, we have shown how much information can be embedded in straight line with the Reed-Solomon error correction. From straight lines of just 5 cm, 16 bits of information could be successfully encoded and recovered. Note that 16 bit is enough to distinguish 2^{16} people. This implies that if a company uses this short line for showing that a certain employee has checked a document, it is possible to identify which employee has checked the document.

References

[1] M. Liwicki, S. Uchida, M. Iwamura, S. Omachi, and K. Kise, "Data-Embedding Pen - Augmenting Ink Strokes with Meta-Information," Proc. DAS, pp.43–51, 2010.

[2] Y. Kato and M. Yasuhara, "Recovery of Drawing Order from Single-Stroke Handwriting Images," IEEE Trans. Pat. Anal. Mach. Intell., 22(9):938-949, 2000.

[3] I. S. Reed and G. Solomon. "Polynomial codes over certain finite fields," Journal of the Society for Industrial and Applied Mathematics, 8(2):300-304, 1960.