Proposal and Evaluation of a Chinese Character Hash Function Based on Strokes for Fingerprinting*

Antoine Bossard†
Kanagawa University
Tsuchiya 2946, Hiratsuka, Kanagawa 259-1293, JAPAN

Abstract

Chinese character representation in computer systems has been a long-standing issue, which is directly related to the information representation and character encoding fields of computer science. For example, as of today some Chinese characters still cannot be easily input in a computer, let alone be universally represented (identified). In this research, we have been especially focusing on such Chinese characters that are not covered by the conventional character encodings, and in this paper, after having previously introduced a universal character encoding for Japanese, we propose a non-ambiguous hash function applicable to any Chinese character. Unlike previous and related works, the proposed function is solely based on the strokes of the character, thus leaving no room for ambiguity. Considering the sparsity and the low collision rate of the described hash function, fingerprinting is a meaningful application, which can then be used for information retrieval purposes, among others. Let us emphasize that simplicity and unambiguity are the two keys of this proposal. The described character hashing method is then evaluated both theoretically and in practice in order to quantitatively show its validity, applicability and contribution.

Key Words: Japanese; Chinese; *kanji*; character; symbol.

1 Introduction

Because Chinese characters are tens of thousands, their representation and processing in general has always been a difficult issue for computer systems. Several approaches were considered as hardware and software evolved over the years. In the early days of computing, the characters were hard-coded into read-only memory (ROM), a solution that is interestingly still in use today for example with some LCD panels [8]. Due to the lack of flexibility of this ROM approach, logical encodings such as those defined by the Japanese Industrial Standards Committee (JISC) rapidly replaced it.

There are two main approaches for such logical character encodings: the unifying approach which considers all the writing systems at once, followed for instance by Unicode [16], and the non-unifying approach which includes the encodings that are local to one writing system, such as Shift-JIS for Japanese. Because both of these two approaches fail at addressing the representation of *any* Chinese character, we recently proposed a universal character encoding for Japanese (UCEJ) [5], which is based on a three dimensional space used to assign distinct coordinates to characters. As per the definition of UCEJ, the first coordinate identifies the character radical, the second coordinate holds the number of strokes of the character and the third one distinguishes variants (forms) of a same character.

It is well known that some Chinese characters are morphologically structured according to composition patterns, such as vertical and horizontal combinations [2, 3, 7, 15]. Such (de)composition patterns can sometimes be ambiguous (e.g. the definition of the support set $\tilde{R}$ of [2]), and moreover they cannot be applied directly to character strokes since strokes, unlike characters, are not structured according to easily identifiable patterns. Hence, this decomposition approach is not a solution to unambiguous character representation.

Besides, a conventional character encoding like Unicode cannot be used as a hashing function given that only a limited number of characters are covered: many Chinese characters are left unsupported, that is unassigned to a code point, and this is typically the case of Chinese characters that are local to one culture, such as the *kokuji* characters of Japanese [2]. Because aimed at supporting any character, the conventional encoding UCEJ could be considered for hashing and fingerprinting, but since the hash value (coordinate) that corresponds to a character cannot be fully deterministically calculated, this is not a solution either.

In continuation of UCEJ, the objective of this research is the proposal of a non-ambiguous hash function that can be applied to *any* Chinese character. Identification of any Chinese character in a unique and unambiguous manner is a first application. Given in some cases the existence of numerous variants for a same character – regularly excluded from conventional encodings – this is a far from trivial issue. Hence, calculating a Chinese character fingerprint is meaningful to refer to a character that is absent from conventional encodings.

The rest of this paper is organized as follows: hashing,

*This paper is an extended version of [1].

†Graduate School of Science. Email: abossard@kanagawa-u.ac.jp.

fingerprinting and character properties are briefly recalled in Section 2. The proposed hash function is then presented in Section 3. It is next theoretically and practically evaluated in Sections 4 and 5, respectively. Section 6 concludes this paper.

2 Preliminaries

We first make a brief recall regarding hashing. Hash functions are frequently encountered in computer science: they are used to calculate an index from a datum so that this datum can be used to directly refer to, for example, the corresponding entry in a table in memory. In the case of table indices, the calculated values are expected to fall within a range so that the table entries tend to be consecutive in memory, and this without any assumption on the sizes of the original data. Besides, the indices calculated for distinct data are expected to be distinct too. If they are not, we say that collisions occur [12].

Such a function which realizes a mapping between data and identifiers, like indices, has other applications, for instance fingerprinting: rather than calculating consecutive or near consecutive table indices, a fingerprint is typically used to identify a datum of arbitrary size, and this with a more or less short value. This is comparable to the scientific applications of human fingerprints. The algorithm described by Rabin is a classic fingerprinting example [6].

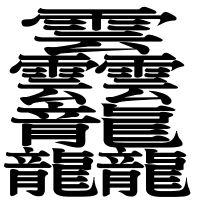
Figure 1: *taito*

Next, we recall essential properties of the Chinese characters. Each Chinese character has one radical, although there exist some characters for which the radical is not clearly identified, or at least is still debated (this is especially the case for characters that have undergone simplifications [2]). Each character has at least one reading, although there are usually several, especially when various languages whose writing system involves Chinese characters are considered.

A character is made of strokes (calligraphic brush strokes), and there is a consensus that the highest number of strokes in a Chinese character, at least in Japanese, is 84. This character, illustrated in Figure 1, is the *taito* character (a.k.a. *daito*, *otodo*) [9]. Furthermore, the strokes of a character are drawn in a precise order, although this order may depend on the writing system considered [2]. Besides, it should be noted that a character can have variants, which are in some cases numerous [13]. Additional details can be found for example in [14].

3 Methodology

We describe in this section the proposed hash function. This function is solely based on character strokes: it relies on the stroke number, the stroke types and the stroke writing order. Because it is essential, we emphasize here that this approach to the function definition induces no ambiguity at all. For comparison, we relied in previous researches for character processing on character radicals and character decomposition operations, two properties which are more (the latter) or less (the former) ambiguous. As recalled in Section 2, the number of strokes, the types of the strokes and the writing order of the strokes for a Chinese character is indeed unambiguously defined. Even if the writing order of the character strokes may differ for a few characters from one writing system to another, such as between Japanese and Chinese, it is clearly defined when considering one writing system. For example, the stroke order of the Chinese characters used in Japanese has been formally established by the Japanese government [10].

So as to lower the collision probability, the proposed function involves all the three aforementioned stroke properties: stroke number, stroke types and stroke order. Regarding stroke types, 36 strokes have been defined by the Unicode consortium for Chinese characters: this is the 31C0–31EF code block [16]. These 36 strokes are shown in Table 1; we have assigned to each of them (columns labeled "Str.") a unique identifier (columns labeled "Id.").

Table 1: The 36 strokes for Chinese characters (Unicode block 31C0–31EF). They are each assigned a unique identifier

Id.	Str.	Id.	Str.	Id.	Str.	Id.	Str.	Id.	Str.	Id.	Str.
0	㇀	6	㇆	12	㇌	18	㇒	24	㇘	30	㇞
1	㇁	7	㇇	13	㇍	19	㇓	25	㇙	31	㇟
2	㇂	8	㇈	14	㇎	20	㇔	26	㇚	32	㇠
3	㇃	9	㇉	15	㇏	21	㇕	27	㇛	33	㇡
4	㇄	10	㇊	16	㇐	22	㇖	28	㇜	34	㇢
5	㇅	11	㇋	17	㇑	23	㇗	29	㇝	35	㇣

Define S the set of these 36 character strokes, and $k : S \to \{0, 1, \ldots, 35\}$ the bijection between a stroke and its identifier. Let C be the set of Chinese characters; it is recalled that its cardinality is unknown. For a character $c \in C$ of $n \in \mathbb{N}^*$ strokes $s_i \in S$ ($0 \le i \le n-1$) and of stroke order that induced by the relation $i < j \Rightarrow s_i$ written before s_j ($0 \le i, j \le n-1$), we define the hash function h as follows:

$$\begin{array}{rcl} h & : & C \to \mathbb{N} \\ & & c \mapsto \sum_{i=0}^{n-1} 2^{6i} k(s_i) \end{array}$$

In other words, stroke identifiers are each represented with six bits, and the stroke number as well as the stroke order are directly induced by the concatenation of 6-bit sequences. The fingerprint can thus be conveniently represented with the octal notation: each stroke corresponds to two octal digits. Examples of fingerprint calculations are given in Table 2; in this table, the stroke order is indicated from left to right and fingerprints are given in the octal notation, with the most significant digit on the left.

Once a fingerprint has been obtained, it can then be adjusted for hashing purposes (e.g. hash table), that is, to reduce the

Table 2: Fingerprint calculation for sample Chinese characters

Character	Stroke number	Stroke types and stroke order	Fingerprint (octal notation)
大 "large"	3	一, ㇒, ㇏	17 22 20
水 "water"	4	亅, ㇇, ㇒, ㇏	17 22 07 32
凧 "kite"	5	㇒, ㇈, 丨, ㇆, 丨	21 06 21 10 22
迄 "until"	7	㇒, 一, 乙, ㇔, ㇔, ㇋, ㇏	17 13 24 24 40 20 22

sparsity of the obtained fingerprints. This would be at the cost of an increased collision rate though. For example, hashing with folding by summing each stroke value, or division hashing by applying a modulo function to the obtained fingerprints.

4 Theoretical Evaluation: Size and Sparsity

4.1 Memory Size Requirements

First, let us compare the size of fingerprints versus the size of a character coordinate in UCEJ. To this end, we first recall that each character stroke is represented on 6 bits, and that the highest number of strokes in a Chinese character, at least in Japanese, is 84 is a consensus. So, a character of n strokes requires at most $6n$ bits ("at most" because the last stroke may not require all the six bits, thus resulting in a few zeros at the MSB, in other words digits that can be discarded). So, an n-stroke character is expressed on at most $6n/8 = 0.75n$ bytes. On the other hand, the coordinate of any character in UCEJ takes 10 bytes [5]: the required memory size does not depend on the character. And in the case of the refinement of UCEJ which takes into account stroke types and the stroke order, each character coordinate takes 38 bytes, again no matter the character [4]. This memory size requirement comparison is illustrated in Figure 2; because a conventional encoding such as Shift-JIS or Unicode only supports a fraction of the Chinese characters, it is not included in this comparison as it would be obviously largely unfair. Given that the vast majority of Chinese characters have at most 30 strokes (this is further detailed in Section 4.2 below), the memory size taken by a fingerprint remains reasonable compared to a UCEJ coordinate.

It is however critical to note that a UCEJ coordinate cannot be completely calculated from a character: as recalled in the introduction, the UCEJ lookup function calculates from a character its X and Y coordinates only, thus not involving Z. This is a major drawback compared to the fingerprint calculation method proposed herein, and one reason for that lookup function not being a suitable hashing function.

4.2 Hash Function Sparsity

Next, we analyze the projected sparsity of the calculated fingerprints. Directly from above, we have that the fingerprint of a 1-stroke character is in the interval $[0, 2^6 - 1]$ (since six bits per stroke), that of a 2-stroke character in the interval $[2^6, 2^{12} - 1]$ (since twelve bits for the two strokes) and so on. Because a

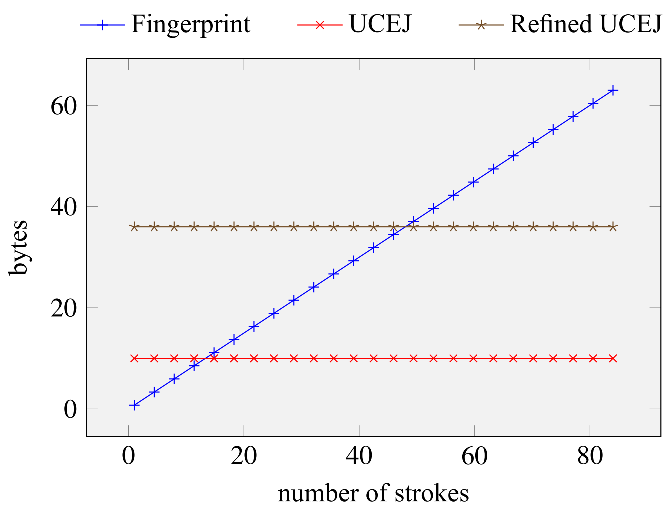

Figure 2: Memory size requirement of a fingerprint versus a UCEJ coordinate

character includes at most 84 strokes as recalled, a fingerprint consists in at most $84 \times 6 = 504$ bits. Therefore, there are a total of 2^{504} distinct fingerprints, which is of course significantly larger than the number of Chinese characters (even if only an estimation, several tens of thousands, of this character grand total is known). So, the character density in the range of the possible fingerprint values is globally low.

The distribution of the stroke number of the Chinese characters used in Japanese is illustrated in Figure 3. For reference, we have represented in the same plot the maximum number of bits required to represent the fingerprint of a character depending on the stroke number. These data have been extracted from the List of MJ Characters provided by the Japanese Character Information Technology Promotion Council [11]. This database includes in total 58 862 characters. Note that 84 has been considered as the highest stroke number as explained, but since the *otodo* character does not appear in the database, the number of occurrences therein is 0. Hence, although this database is rather exhaustive, the zero number of occurrences as soon as stroke number 65 is yet another indicator of the lacking support of the Chinese characters by computer systems.

It should be noted that the proposed fingerprinting algorithm is not perfect in the sense that it is possible – although rather rare – to find two distinct characters that induce the same fingerprint, for example 引 *hiku* and 弔 *tomurau*, both of fingerprint 21

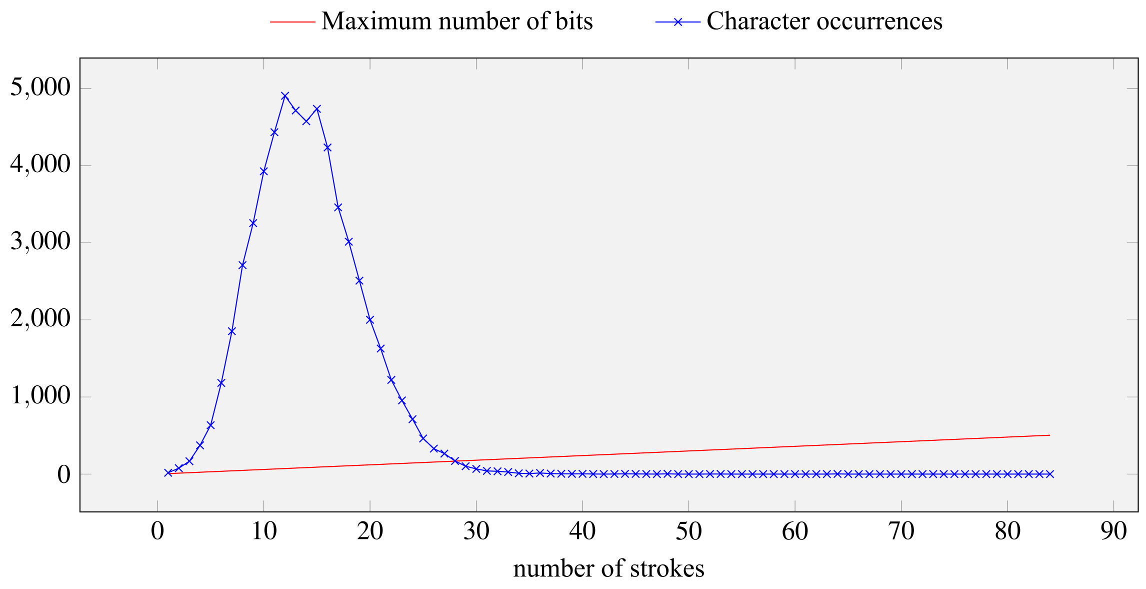

Figure 3: Distribution of the stroke number of the Chinese characters used in Japanese

`11 20 25` (octal notation). In other words, the described hashing function is not injective. In an attempt to further reduce the collision rate, additional character properties could be considered. Nonetheless, this would be at the cost of increased ambiguity in the function definition. It is recalled that we have completely eliminated such ambiguity with the approach proposed in this paper. Besides, in this search for a perfect hashing function, it will become clear that the successively established functions, defined at the beginning in a discrete manner, will inevitably evolve towards a continuous (i.e. non-discrete) function, which is problematic considering the hashing applications.

Finally, it is interesting to remark the following paradox regarding character density: the characters that have the greatest stroke number are those whose fingerprint occupies the greatest number of bits but which are the least "dense" characters. That is, when considering characters of at most n strokes, the number of representable such characters is 2^{6n}, but at the same time as n increases, the number of n-stroke characters (i.e. character occurrences) decreases. This is clearly visible in Figure 3.

5 Practical Evaluation: Collision Analysis

5.1 Methodology

In this section, we conduct another quantitative analysis by measuring in practice the collision rate of the proposed hash function. To this end, we have generated a database that associates to a Chinese character the ordered sequence of its strokes, such including the stroke types and stroke order information.

This database is essential to this work; it has been created with the following recursive algorithm.

Step 1. We have manually defined character (topmost) decompositions: only the vertical and horizontal composition operations have been considered, which is not an issue since these two composition operations cover the vast majority of Chinese characters (more than 80% for a representative character subset [2]).

For instance, the topmost decomposition operation of the character 加 is defined as the algebraic expression 力 + 口, with "+" the horizontal composition operation.

Step 2. We have manually defined the ordered stroke sequence for a few characters that are "prime", that is which cannot be further decomposed [2]. This is typically the case for character radicals. This step induces the base case of the recursion.

For instance, the ordered stroke sequence for the character radical 禾 is manually defined as: ㇒, ㇐, ㇑, ㇒, ㇏.

Step 3. For each character c of the database obtained at Step 1, we recursively calculate its ordered stroke sequence as follows: if a stroke sequence for c has been already defined (i.e. during Step 2 or Step 3), it is returned. Otherwise, let $c_1 \bullet c_2$ be the decomposition of c obtained from the database of Step 1, with "$\bullet$" a composition operation such as "+". We apply this process recursively on c_1 and c_2 to obtain the ordered stroke sequences $\bar{c}_1$ and $\bar{c}_2$, respectively. Let $\bar{c}$ be the concatenation of the two ordered stroke sequences $\bar{c}_1$ and $\bar{c}_2$. We record and return this newly obtained ordered stroke sequence $\bar{c}$ for the character c.

This stroke sequence calculation method is exemplified below. For instance, consider c = 量. No stroke sequence exists for this character. Its decomposition 旦 × 里 is obtained from the database of Step 1, with "×" the vertical composition operation.

Next, we have c = 旦. No stroke sequence exists for this character. Its decomposition 日 × 一 is obtained from the database of Step 1.

So, we have $c =$ 日. This character is prime, and thus its ordered stroke sequence has already been calculated (Step 2); it is returned: ㇑, ㇕, ㇐, ㇐. Then, we have $c =$ ㇐. This character is also prime, and thus its ordered stroke sequence has already been calculated (Step 2); it is returned: ㇐.

Hence, the ordered stroke sequence for $c =$ 旦 is obtained by concatenation: ㇑, ㇕, ㇐, ㇐, ㇐. Next, we have $c =$ 里. This character is prime, and thus its ordered stroke sequence has already been calculated (Step 2); it is returned: ㇑, ㇕, ㇐, ㇐, ㇑, ㇐, ㇐. Therefore, the ordered stroke sequence for $c =$ 量 is obtained by concatenation: ㇑, ㇕, ㇐, ㇐, ㇐, ㇑, ㇕, ㇐, ㇐, ㇑, ㇐, ㇐.

It is important to note that this newly created stroke sequence database is not perfect: for example, we assume that the stroke order for a character decomposed as $c_1 \bullet c_2$ consists first of the strokes of c_1 and then of those of c_2. This is true in most cases, but there can be exceptions, albeit rare. In addition, some stroke sequences are assumed, like ㇔, ㇔, ㇀ (i.e. 氵) for 水. That is, when the character 水 is encountered, its stroke sequence is assumed to be that of its variant 氵; in this research, this variant 氵 largely supersedes 水 so this assumption is safe.

Also, all the decomposition operations identified for Chinese characters (refer to [2]) are not present in the decomposition database (i.e. the first step of the algorithm), so the calculated fingerprints are for a part of the characters. But as recalled previously, the horizontal and vertical decomposition operations cover the vast majority (80%) of characters. Hence, this suffices to obtain representative fingerprints.

5.2 Results

We give in this section raw results of the collision analysis experiment. These results are discussed in the next section. First, the distribution of the calculated ordered stroke sequences based on the stroke number is shown in Figure 4.

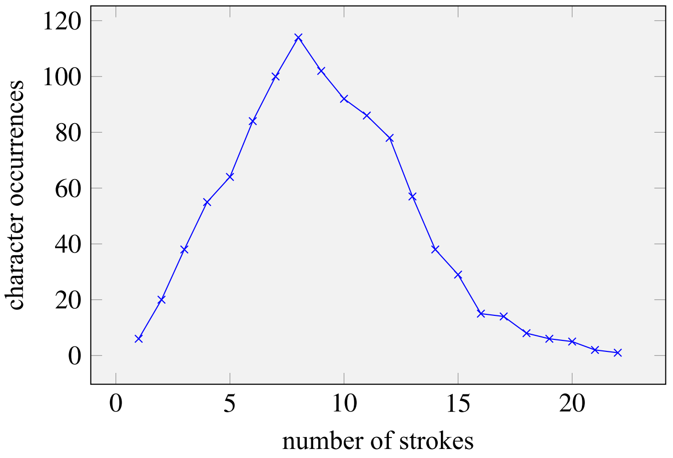

Figure 4: Distribution of the calculated ordered stroke sequences based on the stroke number

Next, we quantitatively measure collisions: 46 characters have not a unique ordered stroke sequence, and have thus not a unique fingerprint. These characters for which hash collisions would occur are summarized in Table 3. In this table, characters are grouped so that the character of a same group have the same ordered stroke sequence. The stroke number is also given for reference. The groups marked with an asterisk (*) need additional explanations: they are given in the next section.

Table 3: A summary of the detected collisions, that is, characters which have a non-unique ordered stroke sequence. Characters are grouped when they have the same ordered stroke sequences

Character group	Strokes	Character group	Strokes
人, 八	2	吉, 吉	6
力, 刀	2	伝, 会 *	6
土, 工, 士	3	隶, 彔 *	6
彳, 心 *	3	貝, 昊	7
日, 曰	4	呈, 里	7
太, 犬	4	昌, 田	8
六, 文	4	径, 怪	8
公, 仏 *	4	治, 冶 *	8
肉, 月 *	4	査, 相	9
召, 加	5	唄, 員	10
旦, 目, 且	5	准, 淮 *	11

Finally, we have extracted from the realized character stroke database the occurrence frequency of each of all the stroke types (refer to Table 1). The details of this analysis are given in Table 4. In this table, the stroke identifier ("Id."), visual rendering ("Str."), number of occurrences ("Occur.") and percentage are given for each detected stroke.

Table 4: Frequency of the stroke types as calculated from the realized database

Id.	Str.	Occur.	%	Id.	Str.	Occur.	%
16	㇐	2 769	30.28	31	㇟	100	1.09
17	㇑	1 565	17.12	22	㇖	95	1.04
18	㇒	1 245	13.62	25	㇙	58	0.63
20	㇔	1 004	10.98	2	㇂	33	0.36
21	㇕	546	5.97	8	㇈	29	0.32
15	㇏	518	5.66	4	㇄	26	0.28
6	㇆	216	2.36	23	㇗	21	0.23
26	㇚	205	2.24	12	㇌	16	0.17
19	㇓	182	1.99	1	㇁	12	0.13
28	㇜	180	1.97	27	㇛	12	0.13
0	㇀	176	1.92	9	㇉	5	0.05
7	㇇	127	1.39	30	㇞	4	0.04

5.3 Discussion

Ordered stroke sequences for a total of 1 014 characters were successfully calculated. The distribution of the calculated ordered stroke sequences according to the stroke number shown in Figure 4 shows that the characters for which stroke sequences were calculated have a distribution that is a on a par with Chinese characters in general (see Figure 3) and thus the validity (generality) of the results obtained in this experiment.

The collision analysis shows that only 46 characters that induce collisions were found, and this in 22 character groups. In other words, this is a collision rate of approximately 4.5%.

However, it should be noted that 14 out of these 46 characters are false positives (they are marked with an asterisk in Table 3): for example, the collisions induced by the two character pairs (准, 淮) and (冶, 治) are false positives: they are detected as collisions because of database assumptions (precisely, that 水 is assumed to be of the form 氵), and will thus not induce collisions in practice. This is also the case for the character pairs (肉, 月), (彳, 心), (公, 仏) and (伝, 会): they are found to induce collisions as well since we assumed 肉 to be of the form 月, 心 to be of the form 忄 and 亻 to be of the form 人 for the same reason we assumed 水 to be of the form 氵. Finally, this is also the case for the pair 隶, 彔: the character element 彐 was assumed as a simplification of character strokes. Hence, the collision rate in this experiment actually amounts only to 3.2% (i.e. 32 characters).

Regarding the stroke type analysis, in total 9 144 strokes were enumerated, and several stroke types were absent from the database: 24 stroke types were detected out of the total 36 (see Table 1). The frequency of the horizontal stroke 一 (more than 30%) is significantly higher than that of the rest. It is not a surprising result since the fact that this stroke is also both a radical and a character (一) shows the omnipresence of this brush drawing.

6 Conclusions

The processing, let alone representation, of Chinese characters in computer systems has been a long-standing issue. It is, for example, still not possible to input some characters into systems, albeit infrequently used ones. To tackle this problem, we have recently introduced a universal character encoding for Japanese (UCEJ). However, UCEJ still lacks a fully deterministic way of calculating the code point of a character. Directly related to this issue, in this paper we have described a non-ambiguous hashing function for fingerprinting Chinese characters. One objective of this work is to facilitate the identification and processing in general of Chinese characters by computer systems. The proposed hashing method has been both theoretically and practically evaluated so as to quantitatively show its validity, applicability and contribution.

As for future works, refining the function definition in an attempt to further reduce the collision probability of the computed fingerprints is a meaningful objective. This however involves several issues, like the sparsity of the hash values, the collision rate, and the simplicity and discreteness of the established function, and because they are interdependent there is no other way but to consider them simultaneously.

References

[1] Antoine Bossard. "A Chinese Character Hash Function Based on Strokes for Fingerprinting." *Proceedings of the 34th International Conference on Computer Applications in Industry and Engineering (CAINE; 11–13 October, online)*, *EPiC Series in Computing*, 79:64–70, 2021.

[2] Antoine Bossard. *Chinese Characters, Deciphered*. Kanagawa University Press, Yokohama, Japan, March 2018.

[3] Antoine Bossard and Keiichi Kaneko. "Chinese Characters Ontology and Induced Distance Metrics." *International Journal of Computers and Their Applications*, 23(4):223–231, 2016.

[4] Antoine Bossard and Keiichi Kaneko. "Refining the Unrestricted Character Encoding for Japanese." *Proceedings of 34th International Conference on Computers and Their Applications (CATA; 18–20 March, Honolulu, HI, USA)*, *EPiC Series in Computing*, 58:292–300, 2019.

[5] Antoine Bossard and Keiichi Kaneko. "Unrestricted Character Encoding for Japanese." *Databases and Information Systems X*, *Frontiers in Artificial Intelligence and Applications*, 315:161–175, January 2019.

[6] Andrei Z. Broder. "Some Applications of Rabin's Fingerprinting Method." *Sequences II*, pp. 143–152, 1993.

[7] Osamu Fujimura and Ryohei Kagaya. "Structural Patterns of Chinese Characters." *Proceedings of the Conference on Computational Linguistics (1–4 September, Sånga-Säby, Sweden)*, pp. 1–17, 1969.

[8] Hitachi, Tokyo, Japan. *HD44780U (LCD-II) (Dot Matrix Liquid Crystal Display Controller/Driver)*, ADE-207-272(Z), '99.9, Rev. 0.0, 1998.

[9] Takehiro Ito. "辞書になかった最多画数の漢字「幽霊文字」の怪…「タイト」さんをご存じないですか?" *The Yomiuri Shimbun (online)*, November 2020. `https://www.yomiuri.co.jp/life/20201030-OYT8T50053/` In Japanese. Last accessed June 2022.

[10] Japanese Ministry of Education, Science, Sports and Culture (文部省). 筆順指導の手びき. First Edition. In Japanese, March 1958.

[11] Japanese Character Information Technology Promotion Council (一般社団法人 文字情報技術促進協議会). List of MJ characters (MJ文字情報一覧表). `https://moji.or.jp/mojikiban/mjlist/` Version 006.01. In Japanese, May 2019. Last accessed June 2022.

[12] Donald E. Knuth. *The Art of Computer Science – Volume 3*, Second Edition. Addison-Wesley, Boston, MA, USA, 1998.

[13] Kyoo-Kap Lee. "Causes of Variant Forms as a Result of Structural Changes to Character Components." *Journal of Chinese Writing Systems*, 1(1):29–35, 2017.

[14] Ken Lunde. *CJKV Information Processing*, Second Edition. O'Reilly Media, Sebastopol, CA, USA, 2009.

[15] Richard Sproat. *A Computational Theory of Writing Systems*. Cambridge University Press, Cambridge, England, 2000.

[16] The Unicode Consortium. *The Unicode Standard 5.0*. Addison-Wesley, Boston, MA, USA, 2007. More recent versions accessible online at `http://www.unicode.org/versions/latest/` Last accessed June 2022.

Antoine Bossard received the B.S. and M.S. degrees from Université de Caen Basse-Normandie, France in 2005 and 2007, respectively, and the Ph.D. degree from Tokyo University of Agriculture and Technology, Japan in 2011.

He is an Associate Professor of the Graduate School of Science, Kanagawa University, Japan. His research is focused on graph theory, interconnection networks, and dependable systems. For several years, he has also been conducting research regarding Chinese characters and their processing by computer systems. He is a member of ACM, ACIS, ISCA, and TUG.