On a Technique for Compact Generation of Symbols —Generation of Hiragana—

RYUICHI SUZUKI*, KATSUO IKEDA* AND TAKESHI KIYONO*

1. *Introduction*

Hiragana listed in Fig. 1 is the Japanese cursive syllabaric character. In the generation of Hiragana symbols, the technique of approximating symbols by line segments and storing the sequence of these lines is rather inefficient, because Hiragana are fundamentally constructed by curves. Therefore in order to reduce the data for Hiragana without loss of legibility of symbols, a contraction of data is necessary in some way.

As a technique for data contraction, approximation of Hiragana by arc segments is examined in this paper. Using this technique, very legible Hiragana symbols are generated requiring less than 64 bits per symbols.

んわらやまはなたさかあ
り みひにちしきい
るゆむふぬつすくう
れ めへねてせけえ
をろよもほのとそこお

Fig. 1. Hiragana.

2. *The Method of Construction of Hiragana*

Hiragana are constructed by orderly connecting a number of basic constructional points by some basic patterns. In this paper, in order to minimize the data requried for each Hiragana symbol, we take the strategy of dividing Hiragana into characteristic patterns, and selecting basic constructional points and basic patterns from the distribution of the dividing points.

The characteristic pattern forming Hiragana is arc. In order to investigate what sort of arcs are most suitable to approximate curves contained in Hiragana, we pick out only simple arcs in Hiragana such as "(" and "⌣" in "あ". These arcs have the inclination that the longer the arc, the lower its curvature, and vice versa. Furthermore, all these arcs tend to have a constant amplitude.

Conversely, we try to divide curves contained in Hiragana according to the inclination of these simple arcs. As an example, "お" divided into arcs with a

This paper first appeared in Japanese in Joho Shori (the Journal of the Information Processing Society of Japan), Vol. 12, No. 5 (1971), pp. 280-285.
* Department of Information Sience, Kyoto University.

constant amplitude is shown in Fig. 2. By dividing all Hiragana in this way, biased distribution is observed in dividing points, and the 25 points are scattered regularly like lattice points, as shown in Fig. 3.

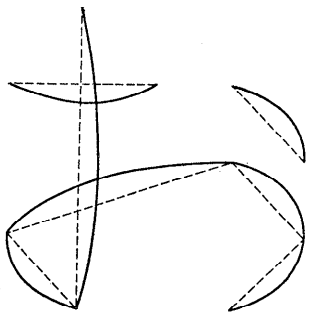


Fig. 2. Division of "お" into arcs.

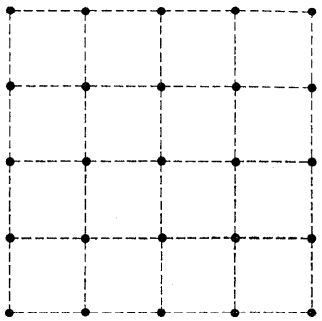


Fig. 3. The distribution of the characteristic points.

3. *Code Allocation Policy*

We try to pack the data for Hiragana within 64 bits. Thinking over "ほ" as an example, the ordinary drawing sequence is the following, as shown in Fig. 4. If coded it becomes

$$N_1AN_2BN_3AN_4BN_5AN_6BN_7AN_8AN_9AN_{10}$$

where A and B denote an arc and a blank line, respectively.

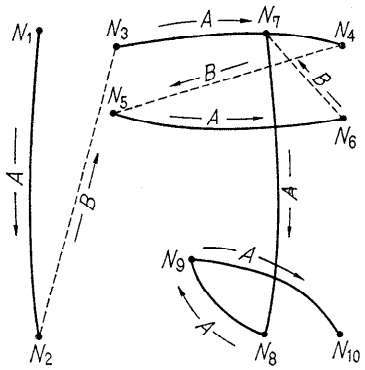


Fig. 4. The construction of "ほ".

Allocating the same length code to each, a code of 5-bits length is required and the total data length is 95 bits. Now by assigning empty codes to the arcs which appear more frequently, the total data length is reduced to 65 bits which exceeds the permitted 64 bits by one bit, thus code of different lengths are allocated to points and patterns.

A code of 5-bits is allocated to basic constructional points and to simplify the handling of different length codes, the number of these points is reduced by 1 to 24. By assigning the code for points as above, from the restriction of

the total data length permitted, the maximum length of a code for patterns becomes 4 bits. Thus, 5 kinds of patterns can be designated by using the left 2 bits to handle the difference in code length.

4. *The Frame for Hiragana*

The frame for Hiragana is constructed by 24 basic constructional points selected from the distribution of the dividing points. This is realized by eliminating one of the 25 points shown in Fig. 2, and in order to keep legibility of Hiragana symbols, it is reasonable to discard a point which makes minimum contribution to the whole Hiragana symbols. By investigating the number of times each point is used actually, the 24 points shown in Fig. 5 are set up as the frame for Hiragana. Symbols using the point eliminated are "む", "り", "を", deteriorated a little, but these symbols are still legible enough.

The hexadecimal numbers added to points are codes indicating their position in the frame for Hiragana.

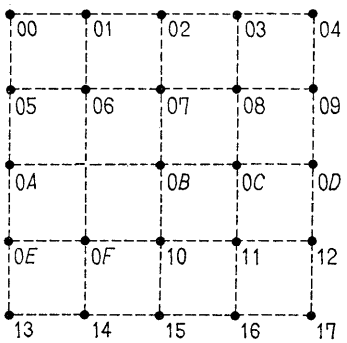


Fig. 5. The frame for "Hiragana".

5. *Selection of Basic Patterns*

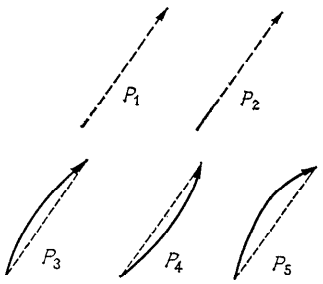
Blank line is necessary as a link to unconnected points.

A sweeping line "ノ" as contained in "い", is observed frequently in Hiragana. These can be expressed as arc, but considering the legibility and data for symbols, it is rather advantageous to set up a sweeping line pattern.

Though arcs are regarded as characteristic patterns, from both points of legibility and amount of data, we use three types of arcs, clockwise and counter-clockwise arcs of low curvature and clockwise arcs of high curvature.

These five patterns selected as explained above are to be the basic patterns forming Hiragana, and are denoted by the following symbols.

P_1: blank line (leading to seperate points)
P_2: sweeping line (unblank segment following blank segment)
P_3: arc I (low curvature, clockwise)
P_4: arc II (low curvature, counter-clockwise)



P_1: blank line
P_2: sweeping line
P_3: clockwise arc of low curvature
P_4: counter-clockwise arc of low curvature
P_5: clockwise arc of high curvature

Fig. 6. Basic patterns forming "Hiragana".

P_5: arc III (high curvature, clockwise)

Each pattern is shown in Fig. 6 and the following codes are allocated to them.

$$(P_1, P_2, P_3, P_4, P_5)=((11)_2, (10)_2, \quad , (01)_2, (00)_2)$$

Pattern P_3 assigned empty code is distinguished from the sequence of codes by the method described in sec. 6.

6. *Data Structure*

Since the codes have different lengths, in order to distinguish the code for points in the frame for Hiragana from the code for basic patterns, the prefix code k,

$$k=(11)_2$$

is considered, and is affixed to the upper side of the code for patterns. As a result of this, codes for points are $(00***)_2$, $(01***)_2$, $(10***)_2$, and codes for patterns have the form of $(11**)_2$. The number of points in the frame for Hiragana is limited to 24, and therefore the codes may be readily distinguished by this prefix code.

In order to reduce the data for symbols, empty code is allocated to pattern P_3 (low curvature, clockwise) which appear most frequently in Hiragana. Furthermore, pattern P_5 (high curvature, clockwise), used at the latter half of "ゐ" and like, appears contiguously in many cases. To make the data structure compact, the above considerations help in establishing the following rules for designation of the patterns.

(i) P_1, P_2, P_4 modify only the points following them.

(ii) P_5 modifies all points continuously, until P_1, P_2, or P_4 follow, which is the case (i).

(iii) continuation of points not following P_5 are considered to be modified by P_3.

PATTERN	P_4	P_2	P_3	P_1	P_3	P_5	P_5	data end
SEQUENCE	00-k1-13-k2-06-09-k3-03-16-k0-10-17-k0							
BINARY	00000 1101 10011 1110 00110 01001 1111 00011 10110 1100 10000 10111 1100							
HEXDEC	0 6 C F 8 C 9 F 1 D B 2 1 7 C							

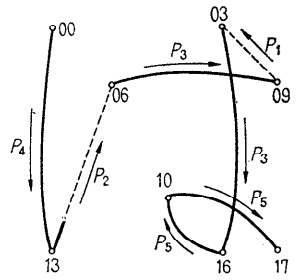


Fig. 7. Construction of "は".

In order to process the data efficiently, the end of data is known by the following.

(1) when data begins no code for points,

(2) when data exceed 64 bits,

(3) when two consequent occurances of P_5 (not necessary contiguously)

Where, the rule (3) indicates data end by a sequence not considered as data for any symbol. In Fig. 7 data for "は" are shown as an example.

7. *Approximation of Arcs*

Making use of the fact that the amplitude of the arcs is constant, arcs connecting the two end points are drawn.

Now let us consider the arc connecting (X_C, Y_C) to (X_N, Y_N). In order to simplify the process, the arc is approximated based on the four equally dividing points (x_i, y_i) $(i=1, 2, 3$ used similarly in the following) of the line from (X_C, Y_C) to (X_N, Y_N). Supposing $\Delta X=X_N-X_C$, $\Delta Y=Y_N-Y_C$, is the value of amplitude (the distance from line to arc),

(i) when $\Delta X \geq \Delta Y$

$$(X_i, Y_i)=(x_i, y_i \pm \Delta_i \operatorname{sign} \Delta X)$$

(ii) when $\Delta X < \Delta Y$

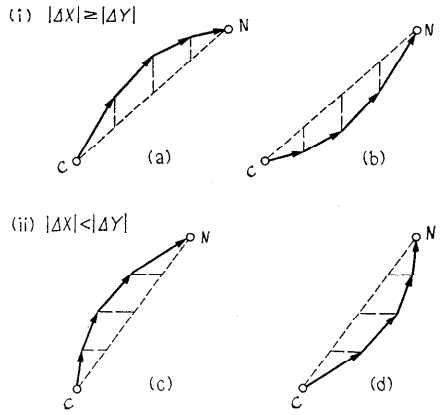
$$(X_i, Y_i)=(x_i \mp \Delta_i \operatorname{sign} \Delta Y, y_i)$$

where $\operatorname{sign} \alpha = \begin{cases} 1 & \alpha \geq 0 \\ -1 & \alpha < 0 \end{cases}$

$\begin{cases} \text{clockwise} \\ \text{counter-clockwise} \end{cases}$

The approximate points (X_i, Y_i) are computed by single addition and subtruction, furthermore approximated arcs sufficiently resemble original arcs, as shown in Fig. 8.

In the generation of Hiragana by this technique, the ratio of the value of amplitude $(\Delta_1, \Delta_2, \Delta_3)$ to the unit length U (the minimum distance between



(i) arc of a gradient of less than 45 degrees, (a) clockwise, (b) counter-clockwise
(ii) arc of a gradient of more than 45 degrees (c) clockwise, (d) counter-clockwise

Fig. 8. Approximation of arcs.

adjacent points) comes into question. As a result of actually examining generated Hiragana for legibility (see Fig. 9), we obtain

(i) low curvature (U: Δ_1: Δ_2: Δ_3)=(12 : 2.5 : 3 : 2.5)

(ii) high curvature (U: Δ_1: Δ_2: Δ_3)=(12 : 4 : 5 : 4)

The proportion of unblank lines to blank lines in sweeping line is taken as one to three.

All Hiragana symbols generated are shown in Fig. 10, and the data for each symbol are listed in Table 1.

9. *Conclusion*

The generation of legible Hiragana symbols with a minimum amount of data can be achieved by approximating the symbol with arcs, setting up the frame for Hiragana, and adopting codes of different length. In this way, the data for

Fig. 9. Improving the quality of "Hiragana" through curvature of arcs.

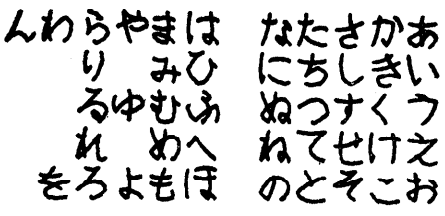


Fig. 10. Generated "Hiragana".

Table 1. Data for "Hiragana".

	Drawing Sequence	Data		Drawing Sequence	Data
あ	09-05-k3-15-02-k3-0C-14-0E-0C-12-16-k0-k0	497E A2F6 51CC 95B3	ね	14-01-k3-13-06-05-k3-13-k0-08-16-11-17-k0	A07E 662F CF11 68DF
い	00-k1-14-k2-08-12-k0-k0	06D3 912C C000	の	07-14-0E-k0-07-0D-16-k0	3D1D 876D B000
う	01-08-k3-05-09-k0-15-k0	0A3C A9CA E600	は	00-k1-13-k2-06-09-k3-03-16-k0-10-17-k0	06CF 8C9F 1DB2 17C0
え	01-07-k3-05-k1-09-13-k3-0F-15-k1-17-k0	09FC BA99 FBEB B7CC	ひ	0D-08-11-15-14-k0-0A-02-00-k0	6A23 5A62 840C
お	07-05-k3-01-14-0E-0C-12-16 k3-08-0D-k0	397C 3473 256F 4373	ふ	02-08-07-11-15-k3-14-0D-17-k3-14-k0-0A-k0	120F 1AFD 1B7F A62B
か	01-13-k3-05-08-15-k2-05-k3-03-k0-0D-k0	0CFC A8AF 17C7 8DCC	へ	17-06-0A-k0-k0	B995 9800
き	03-05-k3-09-0A-k3-16-14-0F-12-02-k0-k0	197D 2AFB 51F2 1660	ほ	13-00-k3-01-04-k3-09-06-k3-03-16-10-17-k0	983C 24F4 9BC7 685F
く	02-0A-15-k0-k0	12AB 9800	ま	05-08-k3-0C-0A-k3-02-15-k0-0E-16-k0	2A3D 8AF1 571D 7C00
け	00-k1-13-k2-06-09-k3-03-16-k0-k0	06CF 8C9F 1DB3	み	05-k1-07-14-0E-12-k3-0C-16-k0-k0	2E9E 8E97 B2D9 8000
こ	06-k1-08-k2-0E-k1-14-k1-17-k0-k0	36A3 9DB4 DBE6	む	07-05-k3-01-0F-0A-0F-k3-0D-08-16-14-0F-k0	397C 2F53 FDA8 B51E
さ	09-0A-k3-16-14-0F-11-02-k0-k0	4ABE D47C 4598	め	15-00-k3-02-14-0E-k0-08-0D-16-k0	A83C 5476 21B6 C000
し	12-15-0F-01-k0-k0	955E 1CC0	も	08-05-k3-0C-0A-k3-11-16-14-02-k0-k0	417D 8AF8 DA82 CC00
す	09-05-k3-02-10-k0-0B-14-k0	497C 50C5 D330	や	03-08-01-k1-15-k3-0A-09-k0-0D-0B-k0	1A03 B5F5 271A BCC0
せ	09-0A-k3-03-0C-k3-17-14-01-k0-k0	4ABC 6CFB D039 8000	ゆ	0E-05-k3-02-15-k3-0E-k0-08-0D-0F-k0	717C 55F7 621F FC00
そ	03-01-k3-16-10-09-0A-k1-03-k0-k0	187E D04A B479 8000	よ	09-07-k3-02-15-k0-0F-17-k0	49FC 55C7 DF00
た	07-05-k3-01-13-k3-0B-0D-k3-17-k0-0F-k0	397C 33F5 B7EF 8FCC	ら	01-08-k2-06-0F-k0-0C-12-15-k0	0A38 CFC6 4AB8
ち	09-05-k3-02-0F-0C-12-16-14-k0-k0	497C 4F64 AD4C C000	り	01-k1-0F-k2-03-0C-15-k0-k0	0EBF 86CA E600
つ	0A-08-0D-12-15-k0-k0	521B 2AE6	る	01-k1-03-0E-k0-0C-12-16-10-16-k0	0E8D D8C9 5A16 C000
て	17-10-0B-04-00-k0-k0	BC16 4066	れ	14-01-k3-17-16-08-13-06-05-k0-k0	A07E F644 CC5C
と	01-0B-k3-17-14-0E-k0-09-k0	0AFE F476 2730	ろ	01-k1-03-0E-k0-0C-12-15-k0	0E8D D8C9 5700
な	07-05-k3-01-13-k3-08-0D-k1-0C-16-10-17-k0-k0	397C 33F4 3759 685F	わ	14-01-k3-13-06-05-k3-13-08-k0-0D-16-k0	A07E 662F CD18 DB60
に	00-k1-13-k2-06-09-k3-17-k0-0F-k0	06CF 8C9F BE3F 3000	を	08-05-k3-02-0A-11-k3-17-14-0F-0D-k0-k0	417C 4A8F DE8F 6E60
ぬ	15-00-k3-02-14-0E-k0-08-0D-16-11-17-k0	A83C 5476 21B6 8DF0	ん	02-13-10-16-k1-12-k0-k0	14E1 6D96 6000

any symbol can be packed within 64 bits. Hiragana are generated efficiently from these reduced data by simplifying the processing of arcs, and as a result, legible symbols as expected can be generated. Program for the processing routine is about 300 steps.

In the generation of symbols in ISO-code reported previously, more legible symbols can be generated under the same restriction by approximating curves by arcs.

Furthermore, generation of chinese characters by the same technique as mentioned in this paper and reference seems to be fundamentally possible. Further considerations hint that a proper classifications of chinese characters considerably reduces the amount of data required.

Reference

[1] Suzuki, R., K. Ikeda and T. Kiyono: A Technique for Compact Generation of Symbols. *J. Information Processing Society of Japan, 11*, 9, pp. 517-522 (1970).

[2] Suzuki, R., K. Ikeda and T. Kiyono: A Technique for Compact Generation of Symbols, —Generation of Hiragana—. *J. Information Processing Society of Japan, 12*, 5, pp. 280-285 (1971).