An Online Handwritten Music Symbol Recognition System

Hidetoshi Miyao and Minoru Maruyama

Department of Information Engineering,
Faculty of Engineering, Shinshu University
4–17–1 Wakasato, Nagano, 380–8553 JAPAN
TEL: Int+81–26–269–5453
FAX: Int+81–26–269–5495
{miyao, maruyama}@cs.shinshu-u.ac.jp

April 13, 2006

Abstract

The objective of this study is to produce a system that would allow music symbols to be written by hand using a pen-based computer that would simulate the feeling of writing on sheets of paper and that would also accurately recognize the music symbols. To accomplish these objectives, the following methods are proposed: (1) Two features, time-series data and an image of a handwritten stroke, are used to recognize strokes; and (2) The strokes are combined, as efficiently as possible, and outputted automatically as a music symbol. As a result, recognition rates of 97.60% and 98.80% were obtained in tests with strokes and music symbols, respectively.

Keywords: Music score recognition, Online handwritten symbol recognition, Pen-based music editing system

1 Introduction

Current composers and arrangers record music by handwriting music symbols on sheets of paper. However, it would be desirable to be able to convert them into computer-readable data so that they could be easily edited and allow other functions, such as printing, automatic performance, dividing parts, and modulation.

In order to allow automatic conversions, many offline handwritten music recognition systems have been proposed [1, 2, 3, 4]. However, according to Yadid-Pecht [4], the recognition rates of the musical notes range between 80% and 90%, which is not high enough. Therefore, they are impractical because it takes too long to correct the errors.

On the other hand, software that permits the direct input of music data have also been developed. In particular, most commercial software has adopted this way and require the use of a mouse, keyboard and music keyboard. Complex notation can be written with these systems; however, the learning curve is long and difficult. Anstice[7] reported that computer systems require about 3 times longer than handwriting to record a score. Consequently, most musicians prefer to handwrite the music rather than to use computers.

Pen-based music editing systems have been proposed because they are more user-friendly and require less time [5, 6, 7, 8, 9]. In such systems, Anstice[7] and Ng[8] defined shapes for symbols that are used frequently. These symbols are easily recognized by a computer. Other symbols can be selected from menus or icons on a computer window. The systems reduce the input time required, but the user needs to memorize the symbols and shapes and learn how to use the application.

George[10] has proposed an online pen-based recognition method for fundamental twenty music symbols by using a multi-layer perceptron. This system allows a user to enter music symbols with the conventional music notation. As an experimental result, the recognition rates of the system are approximately 80% on unseen test data. For practical use, we think the system with higher recognition rates is required.

Our aim is to propose a music symbol recognition system, that is simple to use. The symbols can be recorded in a manner that is similar to traditional handwritten music, and they can still be accurately recognized. To increase the accuracy of recognition of the strokes (in this paper, the term "stroke" means that it is a connected component from pen-down to pen-up), time-series and directional features were incorporated into the system. The directional features are obtained by an image of handwritten stroke. In addition, a proposal is made for a method that allows for the recognized strokes to be efficiently combined as a music symbol.

2 Input of strokes for each symbol

Music symbols vary widely in shape compared with character symbols. In particular, the combinations of shapes in music notes are innumerable since the number of note heads, dots, and flags included in a note is unlimited. It is impossible to prepare identifiers for all kinds of music symbols in contrast with that identifiers can be generated for all characters in OCR systems. Therefore, the system proposed here deals with the individual strokes of a music symbol instead

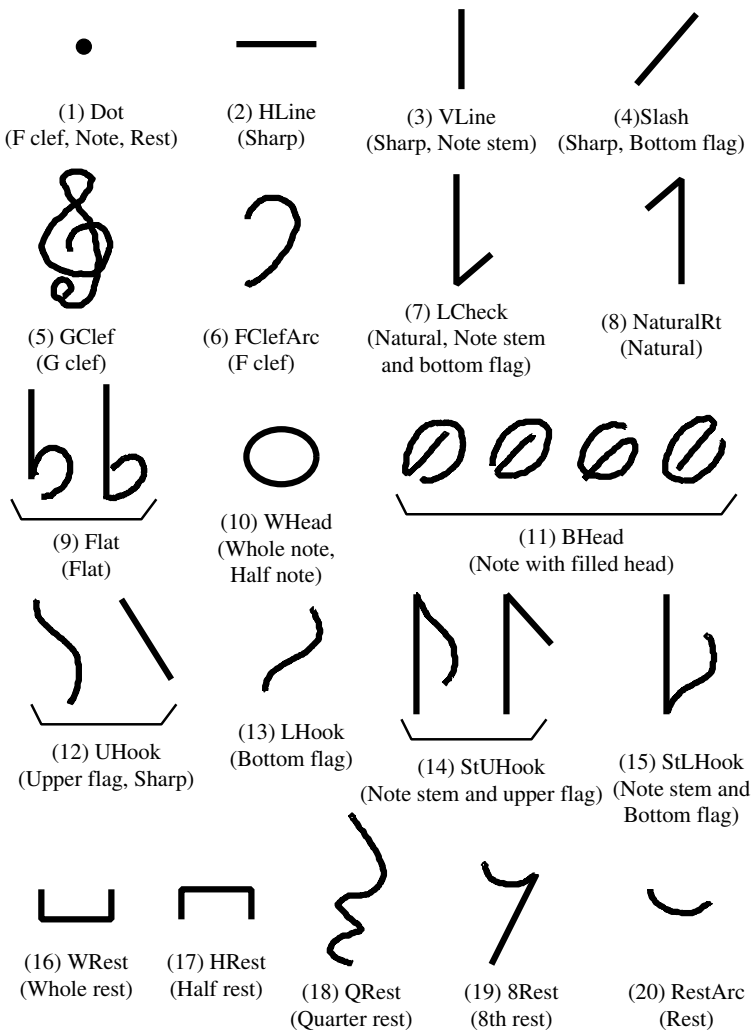


Figure 1: Individual strokes for music symbols.

of the symbol itself. Then, they are combined to make one symbol.

The recognizable strokes for the proposed system are shown in Fig.1. The shapes are similar to those used for handwritten symbols, with the exception of the filled-note head. The shape of the filled-note head is a circle with a slash, as shown in Fig.1(11); it was modified to reduce the input time. Each stroke in Fig. 1 need to be written with one stroke; however, the start point is not restricted.

With this approach, the following music symbols can be generated: G clef, F clef, sharp, flat, natural, musical notes (except for the beamed notes), and rest symbols. The names of music symbols to which the stroke can belong, are listed below the stroke names in Fig.1. For each music symbol, the strokes need to be written successively in order to combine the strokes into a

music symbol automatically (discussed in the section 4). However, the system does not require that a particular order of entering the strokes should be followed when they are written. For example, the four strokes required for the creation of a "sharp" may be written in any order; the software will automatically recognize them as a sharp.

3 Recognition of the strokes in each symbol

An online entered stroke is sampled at an even time interval and points are obtained in order of time. The stroke can be represented by lines connecting the points. The stroke is classified by using the following method just after the pen is up.

3.1 Feature extraction by DP matching

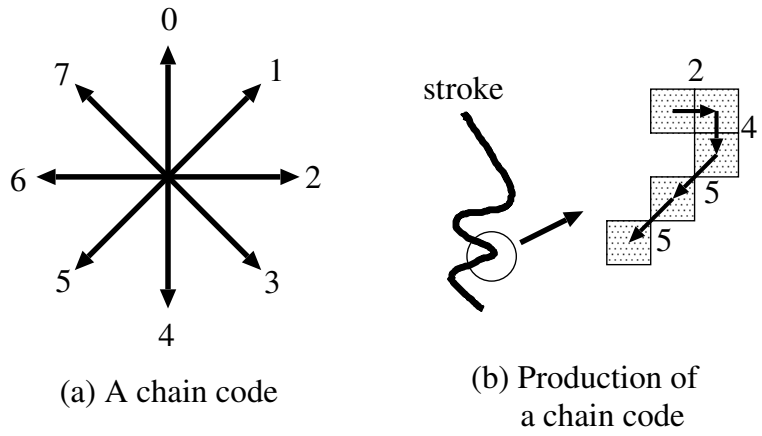


Figure 2: 8-direction Freeman Chain Code.

An 8-direction Freeman Chain Code is used to represent the time-series data of the stroke, as shown in Fig.2. Standard templates were prepared with chain codes for each stroke in each symbol. An input stroke is recognized based on matching of input and template codes. However, the length of the input code does not necessarily coincide with that of the template code. String edit distance computation based on Dynamic Programming (DP) is suitable for matching the codes [11].

The chain code series of an input stroke and the k-th template are represented by $\{a_1, a_2, \cdots, a_I\}$ and $\{b_1^k, b_2^k, \cdots, b_J^k\}$, respectively. When the two code series, $\{a_1, a_2, \cdots, a_i\}$ and $\{b_1^k, b_2^k, \cdots, b_j^k\}$,

are matched most feasibly, the accumulated distance $g_k(i,j)$ is calculated as follows:

Initial values :

$$\begin{cases} g_k(0,0) = 0 \\ g_k(i,0) = g_k(i-1,0) + 1, & for\ 1 \le i \le I \\ g_k(0,j) = g_k(0,j-1) + 1, & for\ 1 \le j \le J \end{cases} \tag{1}$$

Recurrence formula :

$$g_k(i,j) = min \begin{cases} g_k(i-1,j-1) + d(i,j) \\ g_k(i-1,j) + 1 \\ g_k(i,j-1) + 1 \end{cases} \tag{2}$$

where $d(i,j)$ means the distance between codes a_i and b_j^k. Its value is assigned to be 0 if the two codes are identical; otherwise, it is 2. Using the above formulas, the minimum accumulated distance $g_k(I,J)$ is calculated. The path length is also obtained by backtracking the calculated process, and then the normalized distance G_k can be calculated based on the length. G_k has a value from 0 to 1. If G_k is small, it means that there is a strong resemblance between the two patterns. The above process is repeated for each template code. The series of normalized accumulated distances $\{G_1, G_2, \cdots, G_M\}$ is then obtained, where M means the number of the prepared templates. Since the start point of the stroke is not restricted, the reverse chain code is also matched to each of the template codes. After examining the distances by using the ordinary chain code and the reverse chain code, the shorter distance is adopted for each template.

This process flowchart is shown in Fig.3 Part A.

3.2 Feature extraction by SVM

The difference of the start position of a stroke may produce a wide variety of chain codes, particularly for the strokes containing a loop, such as the WHead, shown in Fig.1(10). We assume that a template chain code for a loop symbol is defined as {23456701}. It means that the start position is the top of the loop. If a test stroke is the same loop symbol, but the start position is the bottom of the loop, the chain code becomes {67012345}. Although the ideal normalized distance between them should be 0, the actual calculated distance becomes 0.5 because the chain code is shifted. For such strokes, the use of the 2-dimensional image features should be better than the use of the chain codes; therefore, they are used as well. First, the size of the circumscribed rectangle of the image is normalized to 64 dots × 64 dots, and then the image is equally divided into regions of 8

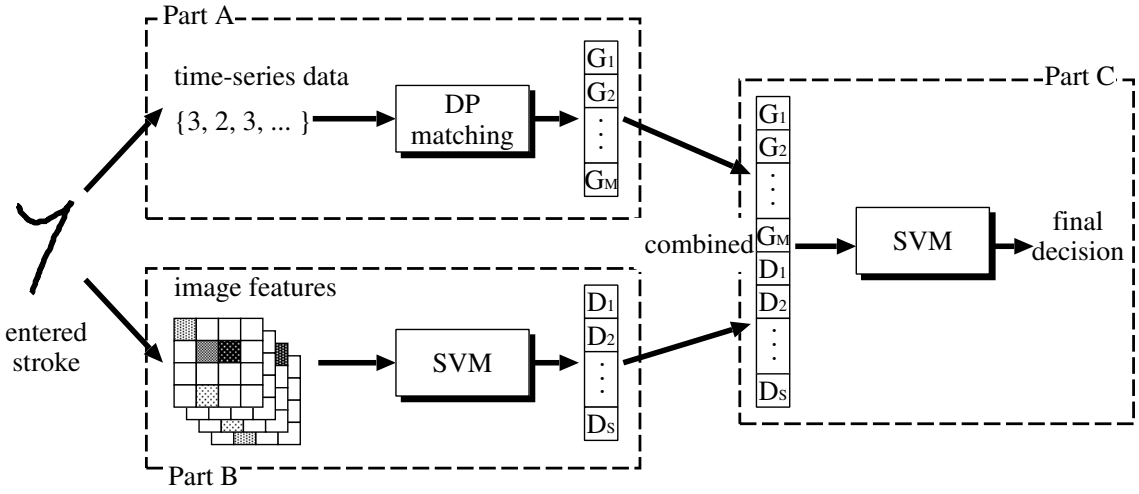


Figure 3: Flow of the stroke recognition.

dots × 8 dots. Second, the directional feature of each region is calculated. As a result, 256 (= 8 × 8 × 4) features are obtained for one stroke. (See Fig.4)

A Support Vector Machine (SVM) is applied to recognize the strokes. SVM is a learning method based on margin maximization principle [12]. SVM performs binary classification by finding optimal separating hyperplane in the feature space. Suppose that a set of training examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ $y_i \in \{-1, 1\}$ is given, the SVM classify the input $\boldsymbol{x}$ based on the function

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) - b \tag{3}$$

where $K(\boldsymbol{x}, \boldsymbol{x}_i)$ is a kernel function which defines the inner product in the feature space. The parameter b is determined by learning from examples. The class label $y_i = -1$ means that the data $\boldsymbol{x}_i$ does not belong to the class and $y_i = 1$ means that $\boldsymbol{x}_i$ does. Coefficients α_is are non-zero only for the subset of the input data called support vectors.

The performance of SVM depends on the kernel. We use RBF(Gaussian) kernel, which outperformed the other commonly used kernels in the preliminary experiments. Gaussian kernel is given as

$$K(\boldsymbol{x}, \boldsymbol{y}) = exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2\sigma^2}\right) \tag{4}$$

For classification of each class of strokes used in the symbols, an SVM classifier was developed. This was accomplished by training with the directional features as input patterns. Using the trained SVM classifiers, a series of output values $\{D_1, D_2, \cdots, D_S\}$ is obtained from all of the

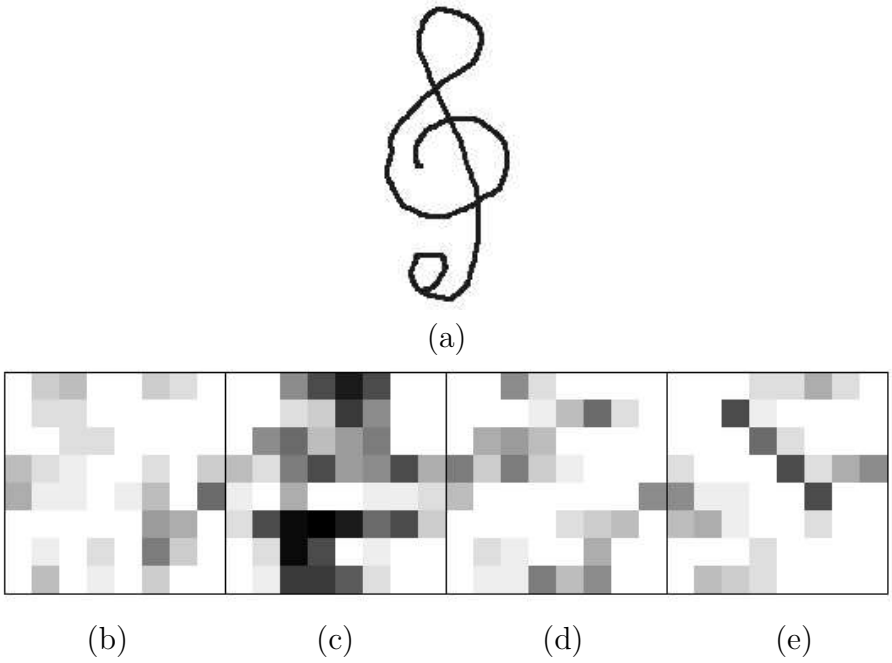


Figure 4: Feature extraction. (a) an image example, (b), (c), (d), and (e) represent directional features for vertical, horizontal, left slant and right slant, respectively.

classifiers, where S means the number of stroke classes and the output values are clamped between 0 to 1 using a sigmoid function.

This process flowchart is shown in Fig.3 Part B.

3.3 Stroke recognition based on the outputs of the two classifiers

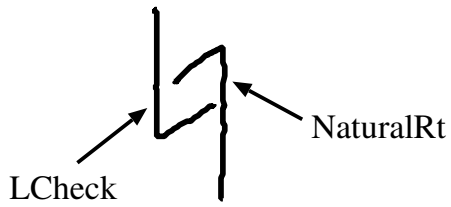
Ultimately, the plan was to classify an input stroke for a symbol based on the output values calculated by two types of classifiers as shown in Fig.3 Part C. An SVM classifier with large capacity for classification is also used. The output values $\{G_1, G_2, \cdots, G_M, D_1, D_2, \cdots, D_S\}$ are given to an SVM classifier as input patterns.

To apply SVM (binary classifier) to multiclass recognition problem, we use one-versus-rest (1vr) type method [14]. Suppose that we are dealing with an n-class problem. In 1vr type method, n SVMs $f_i(i = 1, \cdots, n)$, each of which classifies a single class from the other classes, are learned from examples. The resultant class c is determined as $c = \arg\max_i f_i(\boldsymbol{x})$.

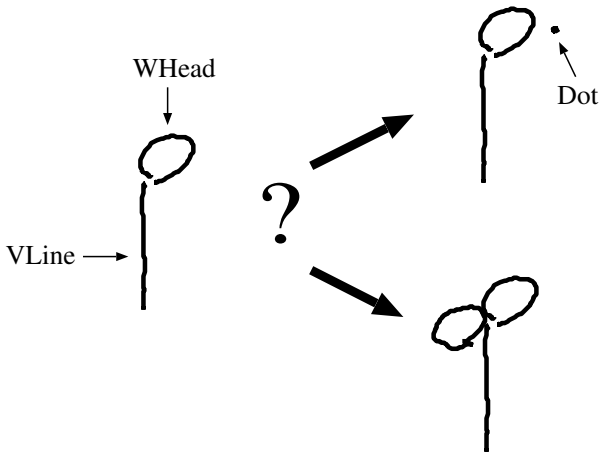
The stroke Dot (Fig.1(1)) needs special care; therefore, it is recognized based on the size of circumscribed rectangle in advance of the classification process for the other strokes.

4 Combining the recognized strokes into a music symbol

The next step is to combine the recognized strokes into a music symbol. This process is applied to each recognized stroke just after the time when recognition of the stroke is completed. The trivial method requires user's direction; when all of the strokes from one symbol are entered, the user indicates so, and the computer can combine the strokes and output a music symbol as a result. This method increases the difficulty of the operation. Intuitively, it appears that the strokes can be combined automatically when they form a specific music symbol. However, it is not that simple. Some symbols can be determined as the specific strokes are entered ("determinable symbols"), but others cannot be ("uncertain symbols"). For example, a natural symbol is formed uniquely when two strokes, an LCheck and a NaturalRt, are entered as shown in Fig.5 (a). On the other hand, a half note can not be formed uniquely because it is unknown that how many WHeads or Dots will be entered in the following steps. (See Fig.5 (b))



(a) Determinable symbol



(b) Uncertain symbol

Figure 5: Examples for determinable and uncertain symbols.

Now, "C" and "P" represent a newly entered stroke and already entered strokes that have not been combined yet, respectively. In order to combine the strokes and output the symbol, the

following methods based on the relative positions among the entered strokes are used:

if(P exists){
  if((C and P form a meaningless symbol(*1)) **OR**
  (C is written far(*2) from all positions for strokes P)){
    if(P coincides with one of the uncertain symbols){
      P is combined and outputted as a result. *# the situation in Fig.7*
    }
    else{
      P is deleted.
    }
    if(C coincides with one of the determinable symbols){
      C is combined and outputted as a result.
    }
    else{
      Do nothing
    }
  }
  else if(C and P coincide with one of the determinable symbols){
    C and P are combined and outputted as a result. *# the situation in Fig.6*
  }
  else{
    Do nothing
  }
}
else if(C coincides with one of the determinable symbols){
  C is combined and outputted as a result.
}
else{
  Do nothing
}

where "meaningless symbol(*1)" means that the symbol is not included in the prepared list shown in Table.1 and "far(*2)" strictly means that the minimum horizontal distance between the bounding box for stroke(s) P and that for stroke C, is more than a certain threshold value d.

In this process, a specific combination of strokes is used for each music symbol shown in Table.1. For example, the data structure of an F clef symbol is defined as a determinable symbol that consists of one FClefArc and two Dots, and a half note is defined as an uncertain symbol that consists of one or more WHead(s), one VLine, and zero or more Dots. Moreover, in order to permit a stroke distortion, some music symbols such as a sharp symbol have several possibility of combination of strokes.

Table 1: List of music symbols with the set of strokes forming them

Music symbol name	Type	Component (strokes)
F clef	Determinable	2 Dots, 1 FClefArc
Sharp	Determinable	2 HLines, 2 VLines
	Determinable	2 Slashes, 2 VLines
	Determinable	1 HLine, 1 Slash, 2 VLines
	Determinable	2 UHooks, 2 VLines
	Determinable	1 HLine, 1 UHook, 2 VLines
	Determinable	1 Slash, 1 UHook, 2 VLines
G clef	Determinable	1 GClef
Natural	Determinable	1 LCheck, 1 NaturalRt
Flat	Determinable	1 Flat
Whole note	Uncertain	0 or more Dot(s), 1 or more WHead(s)
Half note	Uncertain	0 or more Dot(s), 1 VLine, 1 or more WHead(s)
Note with filled head	Uncertain	1 or more BHead(s), 0 or more Dot(s), 0 or more UHook(s),1 VLine
	Uncertain	1 or more BHead(s), 0 or more Dot(s), 0 or more LHook(s), 0 or more Slash(es), 1 VLine
	Uncertain	1 or more BHead(s), 0 or more Dot(s), 1 StUHook, 0 or more UHook(s)
	Uncertain	1 or more BHead(s), 0 or more Dot(s), 0 or more LHook(s), 0 or more Slash(es), 1 StLHook
	Uncertain	1 or more BHead(s), 0 or more Dot(s), 1 LCheck, 0 or more LHook(s), 0 or more Slash(es)
Whole rest	Uncertain	0 or more Dot(s), 1 WRest
Half rest	Uncertain	0 or more Dot(s), 1 HRest
8th rest	Uncertain	1 8Rest, 0 or more Dot(s), 0 or more RestArc(s)
Quarter rest	Uncertain	0 or more Dot(s), 1 QRest

In the Figs.6,7, examples of the combining processes for a determinable symbol "Natural" and an uncertain symbol "Half note" are shown, respectively.

Suppose that a stroke LCheck is drawn in the first step as shown in Fig.6(a). Since this stroke does not coincide with any determinable symbols, the system does nothing. Next, a stroke NaturalRt is drawn. In this figure, the already entered stroke LCheck is denoted by "P" and the newly entered stroke NaturalRt is by "C," respectively. The strokes C and P are combined and outputted as a natural symbol since the strokes are satisfied with the following two conditions:

- The stroke P exists.
- C and P coincide with the determinable symbol "Natural."

Fig.7(a) shows the example in which WHead and VLine are drawn in the first step. They are denoted by "P." Next, a stroke Dot denoted by "C" is drawn. In this case, the system does nothing. In the next step shown in Fig.7(b), the already entered strokes are denoted by "P" and another WHead denoted by "C" is added. The strokes P are combined and outputted as a dotted half note since the strokes are satisfied with the following three conditions:

- The strokes P exist.
- The stroke C is written far from all positions for the strokes P.
- The strokes P coincide with the uncertain symbol "Half note."

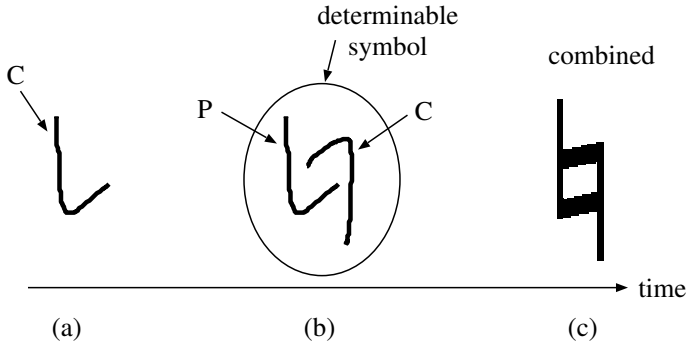


Figure 6: Combining process for a determinable symbol.

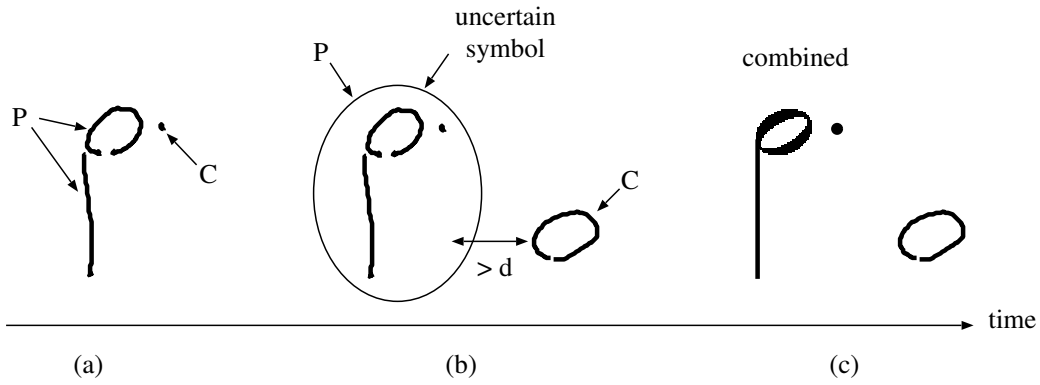


Figure 7: Combining process for an uncertain symbol.

5 Experimental results and discussion

For these experiments, a PC (Pentium 4 CPU; 1.8GHz; 512MB memory) and an A4 tablet were used. Twenty-nine standard template chain codes for DP matching and twenty-one items shown in Table.1 for combining strokes were prepared. The threshold value d used for combining strokes was 0.7 $\times$ (interval of staff lines), it was decided through preliminary experiments. SVMs were built with SVM^{light} software[13]. Their various parameters were set as the most suitable value through the experiment.

To build the SVM classifiers, 6,509 strokes, written by 6 users (who are not expert musicians and do not have any knowledge about this method), were used. For 7,292 strokes written by other 5 users, the recognition result for each stroke is shown in Table.2 where the "recall" and "precision" are defined as follows:

$$recall = \frac{True\ Positive}{True\ Positive\ +\ False\ Negative}, \tag{5}$$

$$precision = \frac{True\ Positive}{True\ Positive\ +\ False\ Positive}. \tag{6}$$

In general, the term "recognition rate" means the recall rate. The "Top2" and "Top3" means the cumulative recognition rate from the best 2 and 3 candidates, respectively. In this table, the total recognition rate of 91.40% was obtained when their time-series features ("Feature1" as shown in Table.2) were only used, the rate of 96.87% was obtained when the directional features ("Feature2" as shown in Table.2) were only used, and the rate of 97.49% was obtained when the both features were used. From this result, it is considered that the combination method provided satisfactory results. In particular, for the loop stroke WHead, it is difficult to recognize only

used by their chain codes, but the recognition rate improved from 64.29% to 99.71% when the combination method was applied. Finally, when the recognition results for stroke Dot, 99.73% (=373/374), were added, the total recognition rate for the strokes was 97.60%.

Table.2 shows that the combined recognition rates for certain strokes are lower than those of Feature1(DP) or Feature2(SVM). In particular for Slashes, the recognition rate of Feature1 is 96.07% and the recognition rate of Feature2 is 90.45%, but the recognition rate of combined feature decreases to 85.67%. However, taking into account the fact that the precision rate of Feature1 was low (78.62%), many strokes were misclassified into Slash symbols in the recognition process, and it had bad effect on the other stroke classification. On the other hand, the precision rate of the combined method was 96.52%. With respect to the precision rate, it shows the performance was improved. It is desirable that both of the recognition rate (recall) and precision rate are high enough. Since the average precision rate for the combined method was 97.49% that is higher than the rates 92.14% for Feature1 and 96.87% for Feature2, it is considered that the combined method is also much superior.

Table.3 shows the misclassified strokes and their percentages, where the percentage for the ith misclassified stroke class is calculated as follows:

$$\frac{False\ Negative\ for\ the\ i\ th\ class}{True\ Positive\ +\ False\ Negative}. \tag{7}$$

In this table, we can see that 13.76% of Slash symbols were misclassified as HLines and 9.46% of StLHook symbols were misclassified as LChecks. Apparently, they are difficult to discriminate. Although stroke recognition has some errors, we can recover and correctly recognize the symbol by using the combining method.

Moreover, to evaluate the robustness of the stroke recognition method, other 400 test strokes were also classified. These stroke shapes were intentionally deformed by a test user. As a result, the top 1 recognition rate of 94.50% and the top 2 recognition rate of 98.25% were obtained. It is shown that our system can deal with patterns with the different cursive degrees.

Next, 250 music symbols including 637 strokes were written by one of the authors who did not create the learning patterns. As a result, two music symbols were deleted; one was improperly recognized; and the other 247 were recognized and combined appropriately (for the note symbol and accidentals, the relative position to the staff lines was also checked), which means that the recognition rate was 98.80%. The following errors were observed: (1) Two strokes that were not a part of a particular symbol were combined because of their location; (2) Before the combining

process began, the stroke recognition process failed. However, the recognition rate was high enough. It was relatively easy to correct or redraw the errors.

We will consider the error case (1) in more detail. If the following conditions are satisfied:

- the strokes P exist,
- the stroke C is not written far from one of the positions for the strokes P,
- the strokes P and C coincide with one of the uncertain symbols,
- the stroke C belongs to a different music symbol from the symbol to which the strokes P belong,

the system miscombines C and P into a music symbol. For example, in order to input two quarter notes, a VLine, a BHead, a BHead and a VLine are written in this order. Ideally, the first quarter note consists of the first VLine and the first BHead, and the second note consists of the second BHead and the second VLine. But if the horizontal distance between the two BHeads is less than d, the second BHead is combined to the first quarter note. To solve this problem, the following solutions are considered:

- When a user writes a different music symbol following an already entered uncertain symbol the strokes of which have not been combined yet, he/she is forced to input the symbol with enough space (i.e., the threshold value d) between them.
- We rebuild the combining process with taking into account more precise spatial relations among strokes.

In the combining process, a little distortion of the stroke was permitted. For this reason, the recognition rate for the symbols (98.80%) was higher than the rate for the strokes (97.60%). For example, ideally, the data structure of a sharp symbol is defined as a determinable symbol that consists of two Slashes and two VLines, but the stroke Slash may be substituted by a stroke HLine or UHook, so that some alternative data were prepared to adapt such distortion. (See Table.1)

Figure 8 is an example of the system's output. The staff lines are prewritten. As the strokes are drawn in the upper window, the symbols are automatically displayed in the bottom window as soon as the recognition and combining processes are completed.

Next, we will consider the response time of the recognition process. The interval time between a stroke is completely entered (i.e., the pen is up) and a result is outputted on a computer display (i.e., next stroke can be entered), can be calculated by adding of the following two times:

1. Time required for recognition of the entered stroke and combining the recognized strokes into a music symbol

2. Time required for displaying the result on a computer display

For the same 250 music symbols used as the test patterns, the times of the above items 1. and 2. were measured. As a result, the average times of the items 1. and 2. were 71.9ms and 1.2ms, respectively, and the average total response time was 73.1ms. Therefore, a user hardly needs to wait while entering symbols.

To evaluate the usability of our system, the writing method is considered. In the traditional methods [7, 8], almost all music symbols are written not in accordance with the conventional music notation, so that a user have to learn how to draw all of the symbols, while our system allows a user to draw symbols with the conventional music notation, except for filled note heads. Although taking into account the fact that a user must learn a few writing rules of our system, it is obvious that the usability of our system is superior to that of the traditional methods at least for beginners.

6 Conclusion

To obtain a high rate of recognition for handwritten music symbols and good user interface that permits handwriting, the following methods were proposed: (1) The classification of the strokes was performed by SVM based on the output values of two methods. One is DP matching, which uses time-series data, and the other is SVM, which uses the directional features of a stroke image. (2) The strokes for each symbol are efficiently combined and automatically outputted as a music symbol. For 7,666 test strokes and 250 test symbols, recognition rates of 97.60% and 98.80% were obtained, respectively. The average response time of the recognition process for one stroke was 73.1ms. The results indicate that the method is practical.

The proposed method suggests that time-series data can be transformed into input patterns for an SVM classifier. For an SVM classifier, in general, the lengths of input patterns have to be the same, so that various length patterns like time-series data can not be treated. But, DP matching can transform time-series data into fixed length data and an SVM can treat them.

Table 2: Recognition result of stroke symbols

Stroke name	Feature1 [%]		Feature2 [%]		Combined(Top1)[%]		Top2[%]	Top3[%]
	recall	precision	recall	precision	recall	precision	recall	recall
8Rest	94.89	99.11	100.00	99.72	100.00	100.00	100.00	100.00
BHead	99.56	84.39	97.64	100.00	99.71	99.85	100.00	100.00
FClefArc	93.98	99.70	97.99	94.21	97.71	93.68	99.71	100.00
Flat	99.78	99.35	97.82	99.78	98.69	100.00	100.00	100.00
GClef	100.00	100.00	100.00	100.00	100.00	99.71	100.00	100.00
HLine	79.91	93.94	96.00	87.96	98.29	87.31	100.00	100.00
HRest	98.22	100.00	100.00	99.71	100.00	100.00	100.00	100.00
LCheck	84.53	79.95	99.71	87.00	99.14	90.81	100.00	100.00
LHook	79.60	93.27	90.23	99.37	89.37	99.36	93.39	94.25
NaturalRt	99.15	96.67	100.00	100.00	100.00	100.00	100.00	100.00
QRest	93.47	100.00	96.02	99.12	95.45	99.70	99.72	99.72
RestArc	92.84	87.33	99.43	98.02	98.85	98.85	99.71	99.71
Slash	96.07	78.62	90.45	95.27	85.67	96.52	91.29	91.29
StLHook	80.80	84.68	83.38	95.10	89.97	95.73	99.71	99.71
StUHook	91.09	100.00	98.70	99.78	98.91	99.78	100.00	100.00
UHook	89.30	89.69	100.00	92.71	100.00	93.85	100.00	100.00
VLine	93.70	76.58	90.83	99.37	97.42	100.00	99.43	99.43
WHead	64.29	88.58	100.00	94.34	99.71	97.49	99.71	99.71
WRest	96.31	98.83	100.00	99.15	100.00	99.72	100.00	100.00
Total	91.40		96.87		97.49		99.16	99.22

The future works are as follows:

- It has been shown that the proposed system can easily combine entered strokes into a music symbol by only using the list shown in Table.1 and the horizontal distance among the strokes. It is sufficient to treat the basic music symbols. However, in order to treat many other symbols and combine their strokes more precisely, it would be necessary to use knowledge on spatial relations among the strokes. For example, a staccato symbol is represented by a Dot symbol and it is located over or under a note head. On the other hand, a time dot attached to a note head is also represented by a Dot, but it is located at the right side of the note head. In the present system, even if a Dot symbol is written over or under a BHead, it is recognized as a time dot. To distinguish between them, the spatial relation between the Dot and the BHead should be considered. It is also necessary to extend our method so that it can treat the symbols that affect two or more music symbols, such as a

Table 3: Misclassified strokes (False Negative)

Stroke name	Misclassified strokes ($FN_i/(TP+FN)$[%])
8Rest	
BHead	WHead(0.29)
FClefArc	WHead(2.01), UHook(0.29)
Flat	StLHook(1.31)
GClef	
HLine	RestArc(1.14), UHook(0.57)
HRest	
LCheck	StLHook(0.86)
LHook	FClefArc(6.61), Slash(2.01), UHook(1.72), QRest(0.29)
NaturalRt	
QRest	UHook(4.26), LCheck(0.28)
RestArc	HLine(0.29), LCheck(0.29), StLHook(0.29), UHook(0.29)
Slash	HLine(13.76), LHook(0.56)
StLHook	LCheck(9.46), StUHook(0.29), WRest(0.29)
StUHook	StLHook(0.87), GClef(0.22)
UHook	
VLine	UHook(1.43), Slash(1.15)
WHead	BHead(0.29)
WRest	

beamed note, a slur and a tie.

- In response to the progress of drawing strokes, music symbols to which the written strokes belong should be limited. Taking into account the fact, if target symbols can be limited step by step in the progress, the time required for recognizing the symbols can be reduced.

- The feature extraction method by DP matching for the strokes containing a loop can be improved by applying the cyclic string matching method [15].

- The proposed system combines the strokes using the first candidates of recognition. However, if second candidates were considered, the recognition rate would improve because the cumulative recognition rate of the top two candidates increased to as high as 99.16%.

References

[1] A. Bulis, R. Almog, M. Gerner and U. Shimony, "Computerized Recognition of Hand-Written Musical Notes", *Proc. of International Computer Music Conference*, pp.110–112, 1992.

[2] K. Ng, D. Cooper, E. Stefani, R. Boyle and N. Bailey, "Embracing the Composer: Optical Recognition of Handwritten Manuscripts", *Proc. of International Computer Music Conference*, pp.500–503, 1999.

[3] G. Watkins, "A Fuzzy Syntactic Approach to Recognising Hand-Written Music", *Proc. of International Computer Music Conference*, pp.297–302, 1994.

[4] O. Yadid-Pecht, M. Gerner, L Dvir, E Brutman and U. Shimony, "Recognition of handwritten musical notes by a modified Neocognitron", *Machine Vision and Applications* 9 (2), pp.65–72, 1996.

[5] W. Buxton, R. Sniderman, W. Reeves, S. Patel and R. Baecker, "The Evolution of the SSSP Score Editing Tools", *Computer Music Journal* 3 (4), pp.14–25, 1979.

[6] K. Silberger, "Putting Composers in Control", *IBM Research*, Vol.23, No.4, pp.14–15, 1996.

[7] J. Anstice, T. Bell, A. Cockburn and M. Setchell, "The Design of a Pen-Based Musical Input System", *OzCHI'96: The Australian Conference on Computer-Human Interaction*, pp.260–267, 1996.

[8] E. Ng, T. Bell and A. Cockburn, "Improvements to a Pen-Based Musical Input System", *OzCHI'98: The Australian Conference on Computer-Human Interaction*, pp.239–252, 1998.

[9] A. Forsberg, M. Dieterich and R. Zeleznik, "The Music Notepad", *ACM Symposium on User Interface Software & Technology*, pp.203–210, 1998.

[10] Susan E. George, "Online Pen-based Recognition of Music Notation with Artificial Neural Networks", Computer Music Journal 27(2), pp.70–79, 2003.

[11] H.Bunke and U.Bühler, "Applications of approximate string matching to 2D shape recognition", Pattern Recognition, Vol.26, No.12, pp.1797–1812, 1993.

[12] R.O.Duda, P.E.Hart and D.G.Stork, *Pattern Classification*, John Wiley & Sons, 2nd edition, 2000.

[13] SVM^{light}: http://www.cs.cornell.edu/People/tj/svm_light/index.html

[14] J.C. Platt, N.Cristianini and J. Shawe-Taylor, "Large margin DAGs for multiclass classification", Adv. Neural Inform. Process. Syst. 12, pp.547–553, 2000.

[15] M.Maes, "On a cyclic string-to-string correction problem", Information Processing Letters, Vol.35, pp.73–78, 1990.

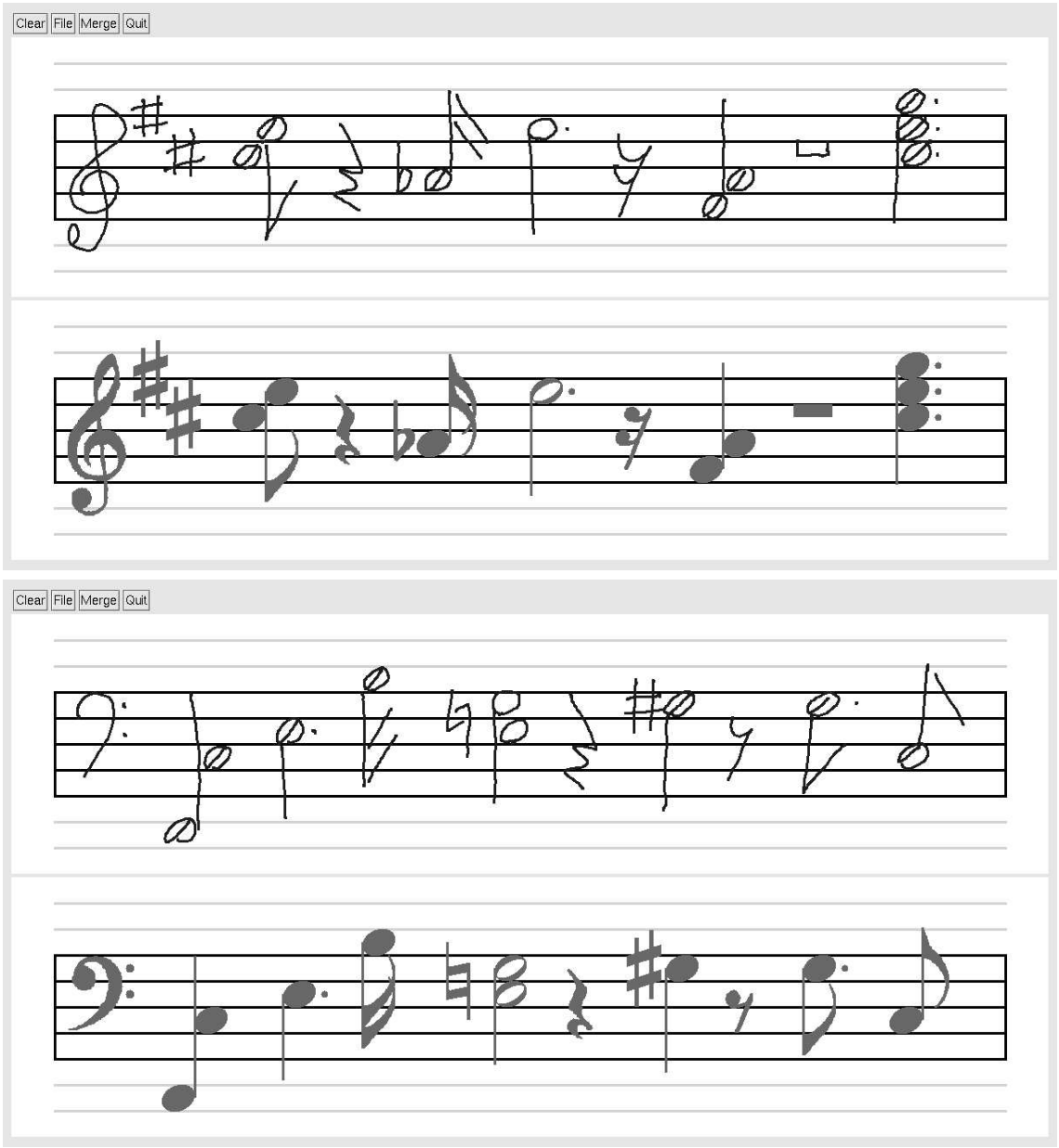


Figure 8: Examples of the system's output.