

應用手寫辨識於 UML 與樂譜作曲之系統

計畫類別： 個別型計畫 整合型計畫

計畫編號：NSC 98-2221-E-009-117-MY2

執行期間：2009 年 8 月 1 日至 2011 年 7 月 31 日

執行機構及系所：國立交通大學資訊工程學系(所)

計畫主持人：陳玲慧

共同主持人：無

計畫參與人員：博士班研究生-兼任助理人員：陳俊旻

博士班研究生-兼任助理人員：林芳如

博士班研究生-兼任助理人員：陳盈如

博士班研究生-兼任助理人員：歐占和

博士班研究生-兼任助理人員：林懷三

博士班研究生-兼任助理人員：楊文超

博士班研究生-兼任助理人員：李惠龍

碩士班研究生-兼任助理人員：林厚邑

碩士班研究生-兼任助理人員：詹子杰

碩士班研究生-兼任助理人員：王維綱

碩士班研究生-兼任助理人員：李宗熹

成果報告類型(依經費核定清單規定繳交)： 精簡報告 完整報告

本計畫除繳交成果報告外，另須繳交以下出國心得報告：

赴國外出差或研習心得報告

赴大陸地區出差或研習心得報告

出席國際學術會議心得報告

國際合作研究計畫國外研究報告

處理方式：除列管計畫及下列情形者外，得立即公開查詢

涉及專利或其他智慧財產權， 一年 二年後可公開查詢

中 華 民 國 100 年 8 月 1 日

中文英文摘要

中文摘要

在本計畫中，我們建構了一個 UML 線上手寫辨識系統。根據我們的觀察，UML 的圖形多半為類似方形或是菱形的圖形，因此在本系統中利用決策樹的方式，來達到辨識的效果。首先我們擷取使用者輸入圖形的幾何特徵，來進行第一階段的分類。接著從輸入圖形擷取我們需要的特徵，和其所屬分類中各個圖形的特徵向量進行比對，即可得到最後的辨識結果。本系統之優點在於可以接受使用者任意筆順的輸入，並且辨識的方法較之前更為簡單有效，正確結果出現在前三名的辨識率為 91.24%。

另外，我們也提出了一個線上的樂譜手寫辨識系統。樂譜是用來記錄樂曲的工具，作曲家常用其於創作與交流音樂。論文中，我們使用與在紙上相同的書寫方法，利用多筆劃組合出音樂符號。而筆劃的特徵有三種，高度，組成之基本圖形以及方向，可用來辨識出此筆劃所屬類型，再將其組合成所需要的音樂符號。本系統支援基本創作需要之全部音樂符號，辨識率為 98.35%，並且提供方便及完善的樂譜修改功能。

關鍵字：手寫、音樂符號、統一塑模語言

Abstract

In this project, we construct an online handwritten recognition system of UML diagrams. We use a decision tree to do recognition. According to our observation, the shapes of the notations of UML diagrams almost look like rectangles or diamonds. Based on this characteristic, an input notation is first classified to the correct category. Then some notation features are extracted from the input notation and used to do final recognition. The advantages of our system are that we can accept free style input and our method is simpler and more efficient than previous methods. The recognition rate of the top three choices is 91.24%.

We also present an online handwritten system for music score recognition. Music score is used to record a music song. People often used to compose a music score on the sheet of paper. In our system, we propose the pen based writing method and use multi-strokes to form a music notation. We extract the height, shape and direction from a stroke as the features and recognize it as a symbol. Then the symbol is combined with other symbols to form a music notation. The system is robust for a general use and supports enough music notations for composition. The recognition rate is 98.35%.

Keyword: handwritten, music score, UML

1. Introduction

In recent years, the development of the handheld devices and pen-based computing hardware, such as PDAs, electronic whiteboards and tablet computers, is grown rapidly, and the handwritten systems which can work in the freehand drawing environment are short of demand. There exists some handwritten recognition systems in some different applications, including math formula [1], engineering drawings [2], table detection [3] and geometric shapes [4-5]. However, the Unified Modeling Language (UML) is widely used in many different domains but there is no handwritten recognition system supporting them.

UML diagrams are widely used in the field of software engineering. Early in the software design cycle, software engineers need to sketch UML diagrams to represent the whole structure of the system. Engineers may draw these diagrams on paper, whiteboard or computer. There are many Computer Assisted Software Engineering (CASE) tools like Rational Rose or Visio to sketch UML diagrams on computer. The functionality of these CASE tools is robust but they have some drawbacks. The most serious drawback of CASE tools is that their design concepts are technique oriented. Technique oriented design provides strong capability but it is not convenient to use. Due to these reasons, we want to build a handwritten recognition system which can allow people enjoying the freedom of drawing UML diagrams by hand.

In 2000, Damm et al. [8] proposed the Knight Project which is a gesture based system for entering and editing UML diagrams. Gestures are some simplified shapes designed by the designer to replace the complex notations. Due to that all of the shapes are simplified, the advantage of gesture based systems is easy to recognize the input notations. However, the user needs to learn what the gestures stand for because they are designed by the designer. In Knight Project, the gestures are separated into two classes, compound gestures and eager gestures, and they use Rubine's algorithm [9] to recognize their gestures. The drawbacks of the Knight Project are that the gesture based system is not intuitional enough. Besides, they do not illustrate the notations supported by their system and there is no experimental result to show their recognition rate.

In 2001, Lank et al. [10] proposed an online recognition algorithm for UML diagrams. The algorithm is composed of the domain dependent kernel and the domain independent kernel. The domain independent kernel deals with the preprocessing steps, including capturing the input strokes, stroke grouping and so on, and the domain dependent kernel is the part of recognition. In the recognition algorithm, they use size, number of strokes, the input order of strokes and the stroke's bounding box size to recognize the input notations. Their algorithm does not allow user drawing the notations in various order because they use the input order as a feature. Besides, there is no experimental result to show their recognition rate.

In 2003, Chen et al. [11] proposed another gesture based recognition system for UML diagrams called SUMLOW. The recognition kernel of SUMLOW combines several multi-stroke shape recognition algorithms to recognize their gestures. The characteristic of SUMLOW is that they allow user modifying, copying, replacing, and deleting input notations via pen-based input technique. Their system has high recognition rate, but there are only six experienced UML designers to participate in their experiment. Thus the recognition rate is not objective.

In 2006, Costagiola et al. [12] proposed an online recognition method for hand-drawn diagrams based on grammar formalism, namely Sketch Grammars. The method uses a parse tree and the Sketch Grammar to recognize input notations. To enhance the recognition rate, the authors propose a language recognizer which

can help the original recognizer to select the best interpretation. This method can be adapted to any notation besides UML diagrams and has high recognition rate. However, a troublesome problem for this method is how the grammars train for new notations.

2. UML NOTATION DATABASE

UML diagrams have thirteen different types and more than forty different notations. However, some of these notations are used rarely and their shapes are more complex. In the project, we choose 23 notations based on UML concepts and the frequency of usage to recognize. These 23 notations are shown in Figure 2.1.

In the project, we invite 20 persons to draw the 23 notations ten times for each and collect the ink data they draw. We randomly choose half of the ink data for training, and the rest for testing.

Structure						
Actor	Class	Component	Interface	Object	Package	Node
Behavior						
Activationbar	Activity	State	Use Case			
Relationship						
Aggregation	Communication	Dependency	Transition	Generalize		
Others						
Branch	End	Initial	Fork	Note	Lifeline	Swimlane

Figure 2.1 Supported notations of the system.

3. PROPOSED METHOD

The proposed method is based on a decision tree and shown in Figure 3.1. The whole process consists of four major phases : geometric feature extraction, category classifier, notation feature extraction (NFE), and final classifier. In the geometric feature extraction phase, some geometric features, such as convex hull, bounding rectangle, PA ratio and Area ratio, are extracted from the input notation. In the category classifier phase, the features extracted in the previous phase are used to classify the input notation to the belonging category. In the notation feature extraction phase, the input notation is divided into primitives and then we extract features like direction, location and distance from these primitives. In the final classifier phase, based on the extracted features, a similarity measure is provided. Based on the similarity measure, the result notation that is most similar to the input notation is determined.

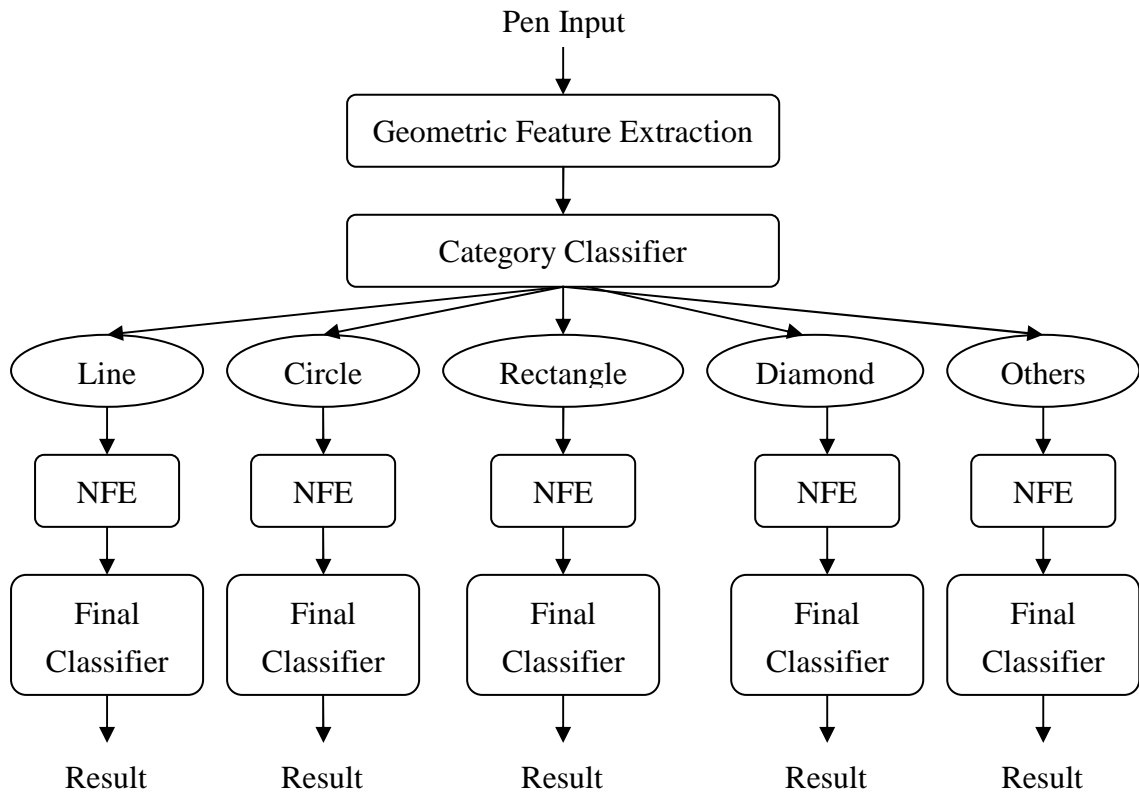


Figure 3.1 The proposed method.

3.1 Geometric Feature Extraction

According to our observation, the notations supported in the system can be divided into five categories, i.e. circle, line, rectangle, diamond, and others, based on their geometric properties. This phase extracts geometric features from input notation to classify it to the correct category. The geometric features we used include convex hull, bounding rectangle, PA ratio and Area ratio. Each of these features is described below.

3.1.1 Convex Hull

The convex hull for a set of points X is the minimal convex set containing X . Figure 3 gives two examples to illustrate convex hull. We use the Graham scan algorithm [14] to find the convex hull of the input notation. Figure 3.2 (b) shows the convex hull of an input notation “Actor”. The blue line denotes the convex hull. After finding the convex hull, we compute its perimeter and the area. These values will be used in the following section.

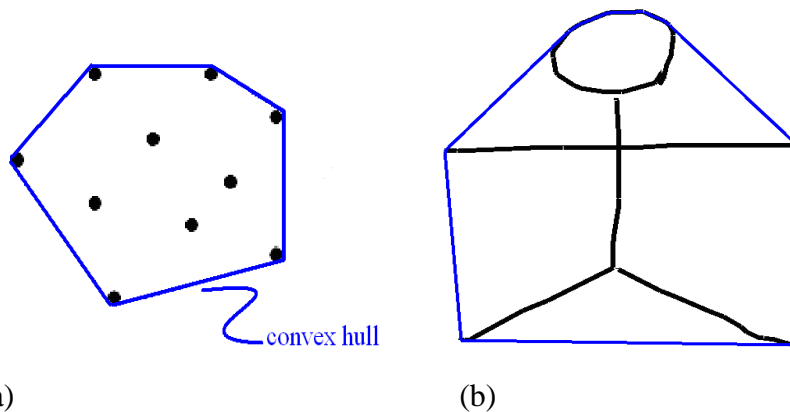


Figure 3.2 Two examples to illustrate convex hull (a) A convex hull of a set of points. (b) The

convex hull of an input notation “Actor”.

3.1.2 Bounding Rectangle

The bounding rectangle is the minimum rectangle containing the input notation. We scan all points of input notation to find the minimum values of x and y coordinates, and the maximum values of x and y coordinates. After finding these coordinates, we use them to establish the bounding rectangle of the input notation. Figure 3.3 shows an example of the bounding rectangle of an input notation “Actor”. The bounding rectangle is shown by red lines. After finding the bounding rectangle, we compute its perimeter and area. These values will be used in the following section.

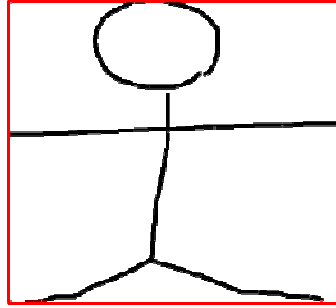


Figure 3.3 An example of the bounding rectangle of an input notation “Actor”.

3.1.3 PA Ratio

PA ratio proposed by Kimura [6] is defined as :

$$PA \text{ ratio} = \text{Perimeter}_{CH}^2 / \text{Area}_{CH}, \quad (1)$$

where Perimeter_{CH} denotes the perimeter of the convex hull of the input notation, and Area_{CH} denotes the area of the convex hull of the input notation. Note that the perimeter and area partly define the shape of an object.

This ratio will be a constant for some kinds of shape. For instance, PA ratio = 16 for any square rectangle and PA ratio = 4π for any circle. Size independent is the main advantage of PA ratio. In the project, PA ratio is used to classify circle and line.

3.1.4 Area Ratio

Area ratio is also proposed by Kimura [6]. The ratio is defined as :

$$\text{Area ratio} = \text{Area}_{CH} / \text{Area}_{BR}, \quad (2)$$

where Area_{BR} is the area of the bounding rectangle of an input notation.

Area ratio also has the property of size independent. In the project, we use this ratio to distinguish the rectangle and the diamond shape.

3.2 Category Classifier

After extracting geometric features, we use these features to classify the input notation to the correct category. The 23 supported notations are separated to five categories including circle, line, rectangle, diamond, and others. The classification of each notation is shown in Figure 3.4. Four different filters, namely circle

filter, line filter, rectangle filter and diamond filter, are provided to distinguish the five categories in the category classifier. The flowchart of the category classifier is shown in Figure 3.5.

3.2.1 Circle Filter

In the category classifier, we use the circle filter to check for circles first. In the project, we use PA ratio for circle filter. PA ratio of a perfect circle of any size is a scalar 4π . Due to that the input may not be a perfect circle, we need to train a threshold range around 4π to classify the input notation. To train the threshold, we compute the PA ratio of the notations belonging to the circle category in the training database first. Then we find a maximum and a minimum as the upper bound and the lower bound of threshold range.





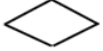






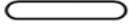
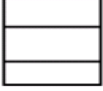


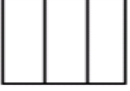
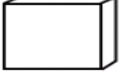
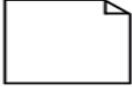
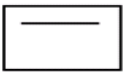
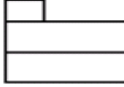


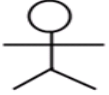

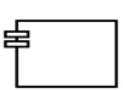

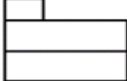
Circle						
Line						
Diamond						
Rectangle						
						
Others						

Figure 3.4 The classification of each notation.

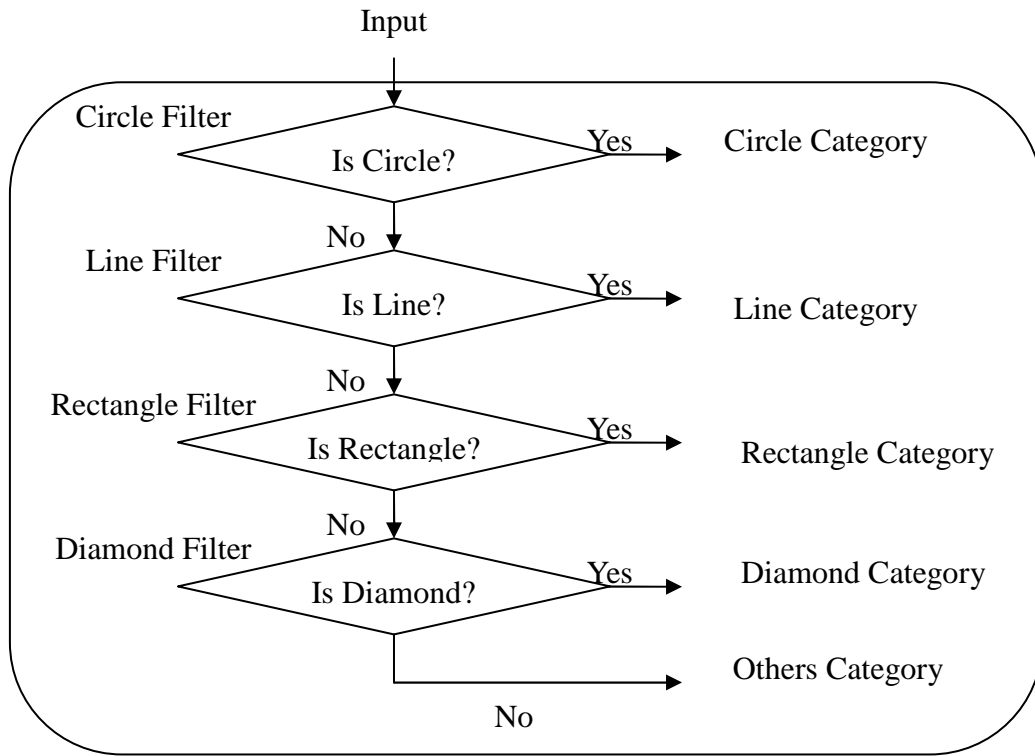


Figure 3.5 The flowchart of the category classifier.

3.2.2 Line Filter

If the input notation does not belong to the circle category, it will be checked by the line filter. Here, we use PA ratio for line filter. Due to the $Perimeter_{CH}$ of a line is close to twice of the length of input notation and the $Area_{CH}$ of a line is closed to the product of the length of input notation and Δh which is the maximum distance between input stroke and its convex hull, the PA ratio of a line can be approximated by

$PA \text{ ratio} \approx \frac{(2l)^2}{l \times \Delta h} = \frac{4l}{\Delta h}$. Since $\Delta h \ll l$ the PA ratio should be large. Here, we take 120 as a threshold value obtained by training. Figure 3.6 shows two examples to explain why the PA ratio is greater than a threshold. In Figure 3.6, the black line is user's input and the red line is the convex hull. To avoid the error of dividing zero, we set the PA ratio equal to 200 when the area of the convex hull of a line is equal to zero.

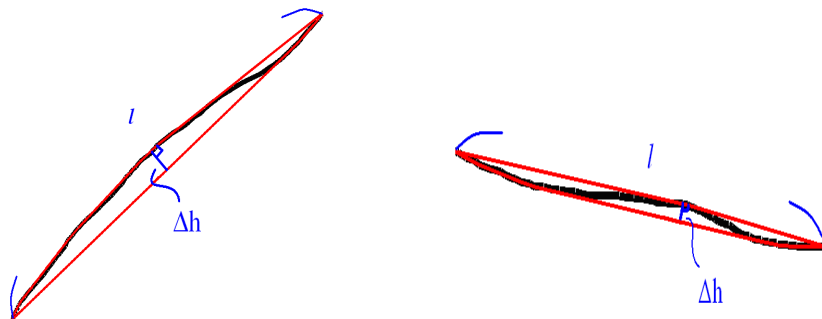


Figure 3.6 Two examples to show the PA ratios of lines.

3.2.3 Rectangle Filter

Rectangle filter will be used when the notation does not belong to the circle or line category. In the project, we use Area ratio for rectangle filter. According to the fact that the $Area_{CH}$ of a rectangle is almost equal to the $Area_{BR}$ of the rectangle, the Area ratio of a rectangle is close to 1. Figure 3.7 shows two examples to explain the fact mentioned above. In Figure 3.7, the black line is user's input, the red line is the convex hull and the green line is the bounding rectangle. To get a threshold range, we also train the rectangle notations in the training database.

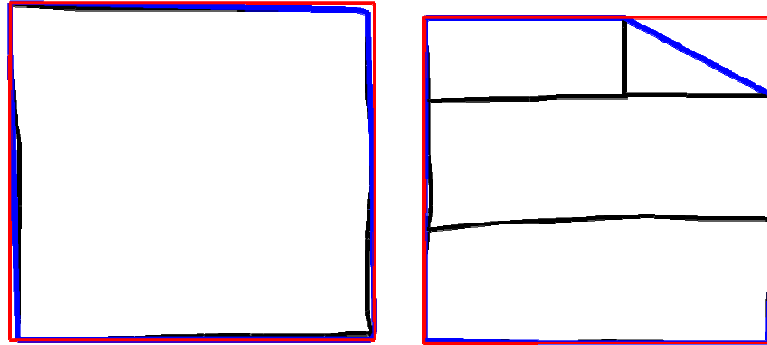


Figure 3.7 Two examples to show the Area ratios of rectangles.

3.2.4 Diamond Filter

If input notation is not considered as a circle, a line or a rectangle, it will be checked by the diamond filter. In the project, we use Area ratio for diamond filter. We assume that the notations belonging to the diamond category are all upright patterns. The $Area_{BR}$ of a diamond is nearly two times of the $Area_{CH}$ of a diamond based on our assumption. In other words, the Area ratio of a diamond is nearly 0.5. Figure 3.8 shows two examples to explain why the Area ratio of a diamond is nearly 0.5. In Figure 3.8, the black line is user's input, the blue line is the convex hull and the red line is the bounding rectangle. We use a threshold range which is trained using the diamond notations in the training database to check whether the input notation belongs to the diamond category or not.

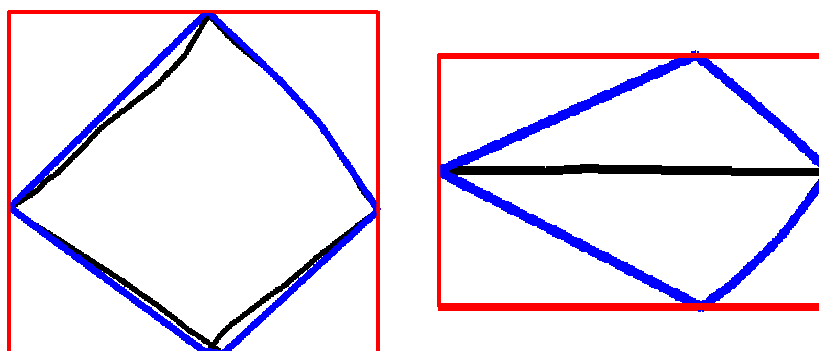


Figure 3.8 Two examples to show the Area ratios of diamonds.

3.2.5 Other Notations

If the input notation does not belong to any category mentioned above, it will be classified to the others

category. In our experiments, after category classification the others category contains Actor and several rectangle notations which are ill-written.

3.3 Notation Feature Extraction

After the input notation is classified to a category, some notation features will be extracted for the final classification. Before extracting notation features, we will first segment the notation into several primitives, which will be described in the following subsection. The notation features extracted include the number of primitives, the direction of each primitive, the location of each primitive, the length of each primitive, and the hollowness of the notation. In the following subsections, we will describe how to extract features.

3.3.1 Primitive

A primitive is defined to be the minimum unit of a notation, which may be a line or an arc. The advantage of segmenting a notation to primitives is that it is much easier for the shape matching procedure to find the matching notation. All the notation features are extracted in primitive level except hollowness.

To divide a notation to many primitives, we use 4-way chain code and the curvature of each point. The 4-way chain code is shown in Figure 3.9. First we compute the chain code for each point. Then we compute the curvature of each point by

$$Cr_{p_i} = \cos^{-1} \left(\frac{x(i-1) - x(i+1)}{\sqrt{(x(i-1) - x(i+1))^2 + (y(i-1) - y(i+1))^2}} \right), \quad (4)$$

where $x(i)$, $y(i)$ denotes the x , y coordinates of point p_i and Cr_{p_i} is the curvature of point p_i . After computing the curvature, we evaluate the curvature difference between two neighboring points to find the dominate points, which have curvature difference greater than a threshold. Finally, the notation is divided into several segments using the dominate points as cut points, each segment is considered as a primitive of the notation. When the notation is segmented to many primitives, we take the number of primitives, N , as the first feature. Note that we have two kinds of primitives: line and curve, which are decided by the sequence of chain codes of the primitive. To decide what kind of a primitive is, we evaluate the chain code difference between each two neighboring points in the chain code sequence and sum all of them. If the summation is larger than a threshold, we will decide that it is a curve; otherwise, it is a line.

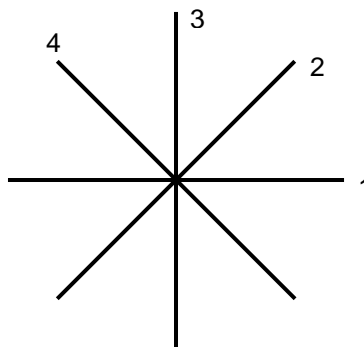


Figure 3.9 4-way chain codes

3.3.2 Direction and Location Feature

The direction of a line primitive is defined as the chain code which appears most frequently in the primitive. If the primitive is an arc or a curve, we set 5 to be its direction. In order to record the directions of the extracted primitives as a feature vector, we should give an unique id to each primitive. The primitives get their unique ids based on the relative locations on the notation. Since some notations have some rotation varieties with 90, 180, and 270 degrees, we provide an algorithm to find relative location.

First, we extract the directions of primitives. Then the primitives with the same direction are collected and sorted according to their top left corner points. Finally, each primitive gets its unique id based on the sorted list. When all the primitives get their unique ids, we combine their directions into a direction feature vector. An example is shown in Figure 3.11; the blue number in the figure denotes the id of a primitive. The provided algorithm is stated below.

Algorithm to Find Unique Id

1. Setting variable i to 1.
2. Collecting the primitives with direction i to a temp list.
3. Sorting the temp list according to the top left corners point of primitives.
4. Giving a unique id to each primitive in the sorted temp list according to its order in the list.
5. Increasing 1 to i . If i is less than 6, go to step 2; otherwise, stop.

When the algorithm is finished, all the primitives have unique ids and we group the directions of primitives according to their ids into a vector, DIR, which is considered as the second notation feature. The notation in Figure 3.10 has (1, 1, 1, 1, 3, 3) as its direction feature.

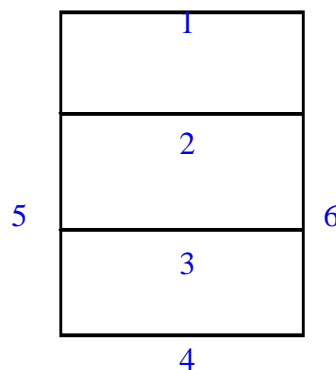


Figure 3.10 An example of the relative location of each primitive in a notation with the direction feature vector is (1, 1, 1, 1, 3, 3).

3.3.3 Length Feature

The length feature is a binary value which represents that a primitive is long or short. To extract this feature, we first find the longest primitive in a notation. Then each primitive is compared to the longest one. If the length of the primitive is larger than half of the longest one, it is considered as a long primitive; otherwise, it is a short one. The length feature is calculated by

$$LEN(i) = \begin{cases} 1 & \text{if } len(i) < \frac{1}{2} \max len(j) \\ 2 & \text{otherwise,} \end{cases} \quad (5)$$

where $len(i)$ denotes the length of the i^{th} primitive, and $\max len(j)$ denotes the length of the longest primitive in the notation.

3.3.4 Hollowness

The hollowness feature is the only feature extracted in the notation level. Hollowness means whether the shape is a solid one or not. A hollow shape has a property that there are no points near the gravity center of the shape. According to this property, we locate a rectangle with size 60% of the convex hull, and the center of the located rectangle is the same as that of the convex hull. If the number of points inside the rectangle is smaller than a threshold, the notation is considered as a hollow shape. Otherwise, the notation is not a hollow shape. Figure 3.11 shows examples of hollowness. Figure 3.11 (a) is a hollow shape, and Figure 3.11 (b) is a solid shape. The hollowness feature, H , is also a binary value and defined by

$$H = \begin{cases} 1 & \text{If } P_{rec} < t \\ 2 & \text{otherwise,} \end{cases} \quad (6)$$

where P_{rec} denotes the number of points inside the located rectangle, and t is a threshold value.

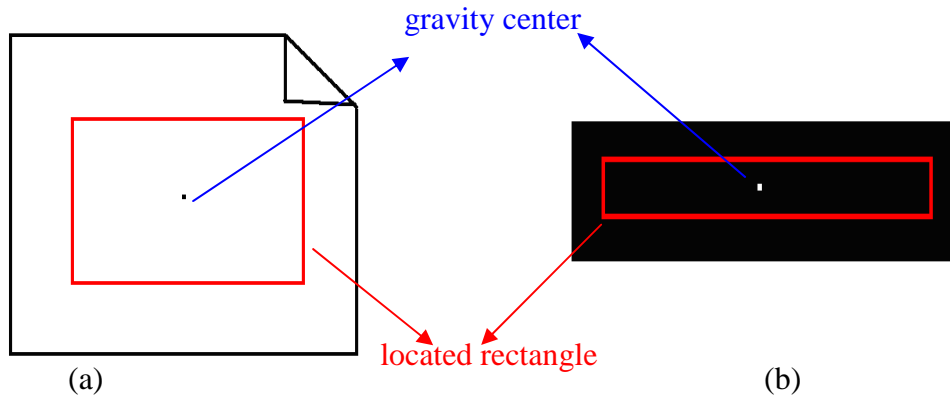


Figure 3.11 Examples of hollowness. (a) A hollow shape. (b) A solid shape

3.4 Final Classifier

Feature vectors extracted from the operations described above, including N , DIR , LEN , and H , are taken for pattern matching at this phase.

In order to obtain the most likely notation for the input notation, we use the inverse of sum-of-absolute-difference (SAD) as the similarity measure. Let notations T and T' be the database notation and the input notation respectively, the similarity between T and T' is calculated by

$$SAD(T) = \sum_{i=1}^4 \frac{|F_i' - F_i|}{K_i}, \quad S(T) = \frac{1}{SAD(T)}, \quad (7)$$

where F_i (F_i') denotes the i th feature vector of T (T'), and $F_i \in \{N, DIR, LEN, H\}$. K_i denotes the number of elements in the feature vector F_i .

Due to the dimension of direction feature vector and the length feature vector are dependent on the number of primitives, we will pad zero to the smaller vector between F_i and F_i' for computing SAD.

Let $T^* = \arg \max_T S(T)$, the input notation is considered to be notation T^* .

3.5 MBSAS Algorithm for Database Creation

The final classifier step uses the inverse SAD to classify the notation. If we calculate SAD between the input notation and all the notations in the database which is described in Chapter 2, the processing time will be very long. Therefore, we use MBSAS to reduce the database and get some representative feature vectors for reducing the processing time.

Modified Basic Sequential Algorithm Scheme (MBSAS) [13] is a clustering algorithm. More specifically, it is an algorithm to group the objects based on attributes. MBSAS does not need to know the number of clusters. It contains two phases. The first phase determines the number of clusters; the second phase is the pattern classification.

4. EXPERIMENTAL RESULTS

In order to evaluate the recognition rate of the proposed method, we invite 20 persons, with poor experience using tablet digitizer and tablet PC, to sketch 23 supported notations about ten times for each notation. We use a tablet digitizer, Wacom Graphire4 CTE-440, and a tablet PC, HP Compaq tc4200, to collect the ink data. In the experiment, we randomly choose half of the ink data for training and the rest for testing. Table 1 shows the recognition rate of the proposed method. The first column shows that only the top one is chosen and the recognition rate is 84.62%. The second column shows that the top three ones are taken, and the recognition rate increases from 84.62% to 91.24%. We can observe that the notations belonging to the Line, Circle, and Diamond categories are classified very well.

In order to show that our proposed method has higher recognition rate than other methods, we compare our method to SkGs method [12]. In SkGs method, there are five students to participate the experiment, and each student draw 20-25 symbols of Use Case diagram. The recognition method proposed in [12] has two parts. The first part only used the Grammar based method to recognize symbol, and the second part combined the Grammar based method and the language recognizer. Our results will be compared to these two parts. In the comparison, we also invite five persons drawing the symbols supported in SkGs method, and the recognition rate is shown in Table 2. In Table 2, we can see that the results of the proposed method are better than these two parts besides Actor. Thus, our recognition rate is superior to SkGs method.

Table 1. The recognition rate of top 1 choice and top 3 choices.

Shape	Top 1 Accuracy%	Top 3 Accuracy%
Activity	73(73/100)	86(86/100)
Aggregation	88.78(87/98)	91.84(90/98)
Activationbar	87.78(79/90)	88.89(80/90)
Actor	87.78(79/90)	92.22(83/90)
Branch	90.91(90/99)	100(99/99)
Class	84.44(76/90)	92.22(83/90)
Component	73.81(62/84)	86.9(73/84)
Communication	100(98/98)	100(98/98)
Dependency	81(81/100)	86(86/100)
End	92(92/100)	92(92/100)
Fork	89.29(75/84)	89.29(75/84)
Generalize	97.96(96/98)	98.98(97/98)
Initial	77(77/100)	81(81/100)
Interface	78.65(70/89)	85.39(76/89)
Lifeline	100(89/89)	100(89/89)
Node	72.22(65/90)	86.67(78/90)
Object	87.78(79/90)	88.89(80/90)
Package	75.56(68/90)	92.22(83/90)
State	71(71/100)	84(84/100)
Swimlane	80.9(72/89)	91.01(81/89)
Transition	94(94/100)	97(97/100)
Use Case	89.89(80/89)	96.63(86/89)
Total	84.62 (1816/2146)	91.24 (1958/2146)

Table 2. Comparison with SkGs method

Shape	SkGs without Language Recognizer (%)	SkGs with Language Recognizer (%)	Proposed Method (%)
Actor	76.92(10/13)	92.31(12/13)	92.31(12/13)
Use Case	83.3(45/54)	90.74(49/54)	96.30(52/54)
Communication	100(21/21)	100(21/21)	100(21/21)
Dependency	72.73(16/22)	72.73(16/22)	95.45(21/22)
Generalize	81.82(9/11)	100 (11/11)	100(11/11)
Transition	88.89(8/9)	88.89(8/9)	100(9/9)
Total	80.99(98/121)	91.74(111/121)	96.69(117/121)

5. Conclusion

The project proposed an online handwritten recognition system of UML diagrams based on decision tree. First, some geometric features are extracted for classifying the input notation to the corresponding category. Then we extract several notation features in primitive level and notation level to create the feature vectors. Finally, the similarity measure based on SAD is calculated for getting the final result.

In the system, users can sketch UML diagrams using tablet computer, digital tablet, and mouse. Users can sketch any notation in any kind of order in the system. After sketching a notation, the standard notation will replace the hand-drawn one and be displayed with the correct position and size. We also support user self-definition function which allows user defining gestures representing the UML notations. Besides these characteristics, the most important property of the system is that it is relative efficient and simple to other methods mentioned above because we use decision tree and reduction database to reduce the comparison time.

Although the system provides many functions of sketching UML diagrams, it is still not enough. In the future, we will add more functions, such as forward/backward engineering, modularity, supporting the multi-layer diagrams, and supporting more UML notations to make the system become a practical tool.

1. Introduction

Music score is a handwritten or printed form of music notations, and it is often used in music composition and music representation. It consists of staff, clefs, notes, rests, and signatures ..., etc.

The common way to record a music score is to write the score on sheets of papers by pencil or pen. As the computer technology grows rapidly, Musicians use computer to aid their composition. In early period, optical music recognition (OMR) is used to recognize the music score which is scanned to an image. However, the error rate of OMR system is relatively high and the editing work of the music score is slow and tedious [15]. Due to the inconvenience of OMR, the online music editing system is proposed. The system can directly output the editing resultant to musicians. Besides, more convenient systems are rapidly developing for user to write on the tablet. One is the “point and click” system, such as MagicScore Maestro [16] and Allegro [17], which selects music notations from menus or icons. Hence, the system can directly input the music notations without recognizing them. Nevertheless, the input processes are tedious and complicated due to many pen and mouse movements [18].

In order to reduce the tedious input processes, gesture-based music score recognition systems are developed. Musicians could use specific gestures to represent specific notations defined by systems. Forsberg *et al.* [19] proposed such a system which uses gesture and voice to input the music notations. In the gesture part, it combines Calligrapher system [20], Rubine’s gesture recognition system [21] and their recognizer to recognize the input gesture. The supported music notations are limited and are not sufficient for professional music editors, and some gestures are irrelative to the shapes of the corresponding music notations. This makes learning curve long and difficult. Anstice *et al.* [15] also proposed a gesture-based system called Presto. After that, Ng *et al.* [22] proposed an improved version denoted as Presto2, which improves both usability and speed of input, but the gestures in the system have little relation with the actual writing. The recognition accuracy of gesture-based system may be acceptable. However, in the gesture-based systems, users must learn and remember these miscellaneous gestures. Therefore, the gesture-based system is often very constraining for the user.

Instead of learning miscellaneous gestures, pen-based handwritten systems are developed to catch the human writing styles. The characteristic of pen-based systems is that the writing styles is as the same as on sheets of papers. There are several methods proposed, like neural network, context-free grammar and SVM. In 2003, George *et al.* [18] proposed such a system with artificial neural networks. They used a multi-layer perception to learn music notations and extract the features. The inputs of these handwriting systems are natural and direct for users, but the error rate may be alarmingly high. Subsequently, music notations can be recognized by the trained neural networks. Taubman *et al.* [23] proposed a handwritten music recognition system based on statistical moments. Nevertheless, the current system is not stable and not robust enough for a general use. In 2005, Macé *et al.* [24] proposed a generic method which recognizes the music score by context-free grammars and lots of recognizers. Unfortunately, the user must follow the writing orders and writing locations that are defined by professional musicians, and it is not friendly for the users that are not familiar with the music theory. Miyao and Maruyama [25] proposed a handwritten system based on time series data and image features. Their system uses dynamic programming and SVM algorithm to recognize handwritten music notations. However, only a small part of music notations is supported in the system. In

other way, the system does not support modification operations, such as deleting or moving a notation, and this makes the system impractical.

2. Stroke Database

A stroke is a collection of points from pen-down to pen-up. A music notation or notation is the basic unit to record music, including staff, clefs, notes, rests, and signatures ..., etc. When we are writing, there are some notations we cannot write in a single stroke, like natural or sharp. We have to write multiple strokes to represent a notation. In other way, some notations have innumerable dots, heads, or flags, and we cannot assure the exacted strokes in these notations. Here, we divide the strokes into 17 kinds of symbol categories, as shown in Table 2.1.

In Table 2.1(1), categories (1) to (6) are called “simple symbols, “ which means that they could be recognized quickly by some extreme properties, like the stroke length. The others are called “complex symbols,” which means they need to extract the features and be classified by the complex symbol classifier which will be elaborated in the next section.

In our database, we obtained the strokes using a WACOM digital tablet written by 14 users. The users are not expert musicians and do not have any knowledge about the music theory. For robustness, the procedure would be carried out at least 1000 times to each user.

Table 2.1 Supported symbols.


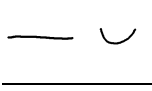

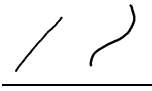











				
(1) Dot	(2) HLine	(3) VLine	(4) Slash (Flag)	(5) UHook (Flag)
				
(6) GClef	(7) FClefArc	(8) Flat	(9) NaturalRt	(10) LCheck
				
(11) StUHook	(12) WHead	(13) BHead		
				
(14) WRest	(15) HRest			

Table 2.2 shows all the supported music notations in this system. There are four types of notations supported.

Table 2.2 Supported music notations.

(a) Bar line
(b) Group
(c) Determinable note
(d) Uncertain note
(d) Uncertain note

3. The Proposed Method

In this system, we recognize the input stroke as a symbol and then combine the symbol with other symbols to form a music notation.

The flow diagram of the symbol recognition is shown in Fig. 3.1. The whole process consists of 4 major phases: preprocessing, simple symbol classifier, feature extraction and complex symbol classifier.

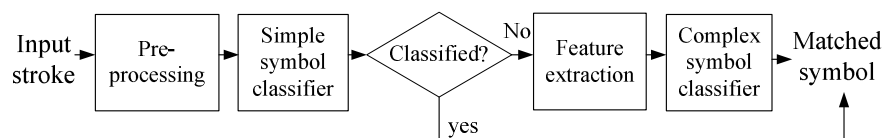


Fig. 3.1 Flow diagram of the symbol recognition.

After the symbol recognition, the notation recognition is conducted. Based on the semantic information, the output symbol would be combined with other existed symbols to form a notation. Finally, the system outputs the printed music notation and puts it at the exact location on staff.

3.1 Preprocessing

In order to reduce the noise and variety in the stroke, we apply the preprocessing, including smoothing filter, gap filter and slipped segment remover.

3.1.1 Smoothing Filter

The reason why a stroke jagged is that some errors occurred in the digital tablet or the unstable state the user is writing in. In order to eliminate these jags, we apply Gaussian filter [26] to make the stroke more smooth and keep the global information of corners in the stroke.

3.1.2 Gap Filter

Because the digital tablet samples points with a fixed time interval, the writing speed makes the distances between two points to be different. There would be some gaps in the stroke. These gaps would affect the curvature detection in later process.

For each two adjacent points, let dx be the x difference between the two points, dy be the y difference between the two points. Then if $\max(dx, dy) > 1$, we interpolate $\max(dx, dy)$ points between them by linear interpolation.

3.1.3 Slipped segment remover

Slips are the action that user's pen move to the unexpected direction on the digital tablet. In the beginning and ending to write a stroke, it is easy to generate surplus slipped segments. We could remove slipped segments by detecting whether the length of the first segment or the last segment in a stroke is shorter than a given threshold.

In order to eliminate the slipped segments, the first step is to find the candidates of slipped segments. Li and Hall proposed a method [27] to find dominant points in a stroke using a support region based on 8 ways chain codes. Then we divide the stroke into several segments by dominant points. The first segment and last one are the candidates of the slipped segments. If the length of the candidate is less than a threshold, it is a slipped segment and would be removed. The threshold is set as half of the gap's height on staff in music score.

3.2 Simple Symbol Classifier

By observing the 17 kinds of symbols, we find that some symbols can be classified quickly by the extreme properties. We call these symbols as simple symbols, including Dot, the straight line of HLine, VLine, the straight line of Slash, the straight line of UHook and GClef. Here, we will discuss how to classify simple symbols.

Among all symbols, the length of GClef is longest obviously. By this property, we could easily recognize a stroke as a GClef symbol if the length of the stroke is longer than the length threshold. The length threshold is set as 12 times gap's height.

By observing the width and the height of a symbol, the Dot symbol has the smallest width and the smallest height in symbols. Therefore, the stroke would be recognized as a Dot symbol if the width and the height of the stroke are both shorter than a given threshold. The threshold is set as half of gap's height on staff.

To classify if a stroke is a straight line, a linearity measure is defined as

$$linearity = \frac{L}{G(P(s), P(e))}, \quad (1)$$

where L denotes the length of the stroke. $G()$ denotes Euclidean distance. $P(s)$ denotes the starting point of the stroke and $P(e)$ denotes the ending point of the stroke. If a stroke is a straight line, the linearity should approach to 1. Thus, if the linearity is smaller than the threshold, 1.07, we consider the stroke as a straight line

and recognize it as HLine, VLine, Slash or UHook according to its slope. Once the stroke is recognized as a simple symbol, it would be output and exit the symbol recognition.

3.3 Feature Extraction

If a stroke is not classified as a simple symbol, we will do feature extraction from the stroke. Here, we take three kinds of features: height, shape and direction.








3.3.1 Height

Notations in music theory have height limitation. Since notations are formed by symbols, symbols also have the height limitation. We could extract the height of a stroke as a feature for rough classification.

3.3.2 Shape

As described in Section 3.1.3, each stroke will be divided into several segments. Every segment has its special shape. The number of shapes would be useful for classifying stroke. There are 7 kinds of basic shapes shown in Table 3.1.

Table 3.1 The 7 basic shapes. (1) Horizontal line. (2) Vertical line. (3) Slash. (4) Backslash. (5) Clockwise curve. (6) Counter-clockwise curve. (7) Circle.

						
(1)	(2)	(3)	(4)	(5)	(6)	(7)

By the linearity and slopes mentioned in Section 3.2, we could determine if a segment is a horizontal line, vertical line, slash or backslash. If the segment does not belong to straight line, it may be a clockwise curve, a counter-clockwise curve, or a circle.

The difference between the straight line and the curve is that the curve changes its direction very often. We could accumulate the direction change value to detect what kind of curve the segment is like.

The last step is to calculate the number of every kind of shapes in the stroke. The vector of dimension 7 containing numbers of seven shapes is viewed as a shape feature.

3.3.3 Direction

The direction is the sequence of writing direction in time order, and it could reflect the writing style of a symbol. The direction extracted from the stroke could help us clarify the difference among some symbols with similar shapes. We use the 8 way chain codes to represent the direction. We could extract the direction by eliminating the duplicate chain codes in the same direction. Note that the dimensions of the direction features of different strokes may be different.

3.4 Complex Symbol Classifier

Symbols except simple symbols are complex symbols. Features extracted from a stroke, including height, shape and direction, are taken for complex symbol matching at this phase.

Based on these three features, we construct three classifiers separately, including height classifier, shape classifier and direction classifier. Because some symbols in some classifiers are similar and are hard to separate them, we build a three level decision tree to deal with this problem. The first level is the height classifier which roughly classifies symbols by the height feature. The second is the shape classifier which classifies symbols by the shape feature. The third level is the direction classifier which classifies symbols by

the direction feature and outputs the recognized result.

3.4.1 Height Classifier

The heights of printed music notations are ruled by the music theory. Because the music notations are formed by symbols, symbols also have the height limitations in writing. By a height threshold, some symbols will be considered as high and some will be considered as low. However, due to the writing distortion, some symbols sometimes will be considered as in the high, sometimes low. We consider these symbols with unsure heights as variant. Fig. 3.2 shows the symbols in the low group, the high group, and the variant group.

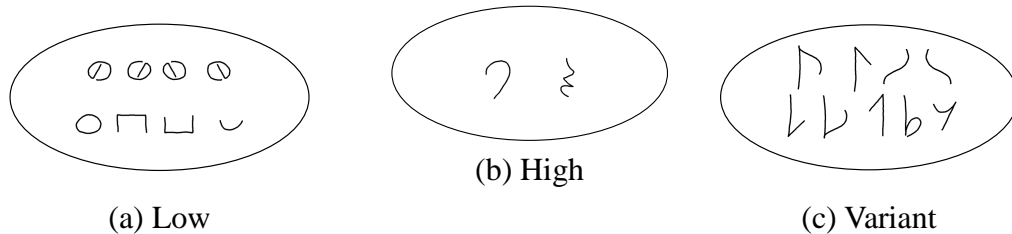


Fig. 3.2 Three groups in height. (a) Low group. (b) High group. (c) Variant

In the height classifier, we use 2 times gap's height on staff as a threshold to classify symbols into the high group and the low group. Fig. 3.3(a) shows the low group in the height classifier. Fig 3.15(b) shows the high group in the height group. The symbols surrounded by the dotted line and originally belonging to the variant group will be handled later.

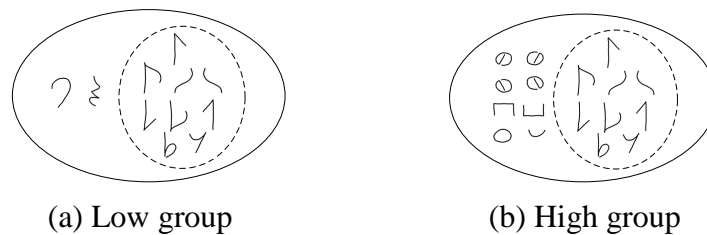


Fig. 3.3 Two groups in the height classifier. (a) Low group. (b) High group.

When a new stroke is coming, this classifier classifies the stroke to the high group or low group based on the height feature.

3.4.2 Shape Classifier

In this stage, we group symbols with similar shape features. Fig. 3.16 shows the groups with similar shape features. The shape difference, SD , between two shape features is defined as follows:

$$SD = \sum_{i=1}^7 \sqrt{S1(i)^2 - S2(i)^2}, \quad (2)$$

where $S1$ is the vector of the shape feature 1. $S2$ is the vector of the shape feature 2.

As a new stroke is coming, the classifier could measure the shape distance between the stroke and the shape templates in each group and applies KNN to find the group with the nearest distance. In H1 of Fig 3.4(b), there is only one possible symbol in the group, which will be output directly without further processing.

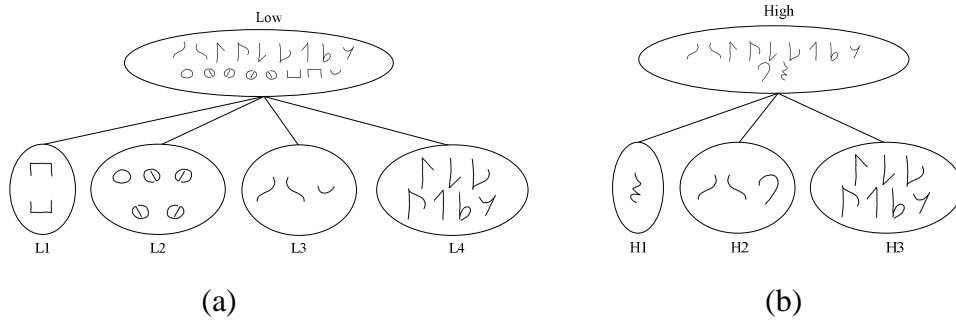


Fig. 3.4 Groups in the shape classifier. (a) For low group. (b) For high group.

3.4.3 Direction Classifier

In the third level of the decision tree, we would find most likely symbol according to the direction feature. We measure the distance between the direction feature of the stroke and the direction templates in database. Because the direction features are variable in dimension, the distance measure could be considered as the string matching problem. We apply the dynamic programming to obtain the distance.

Let $\{a_1, a_2, \dots, a_l\}$ denotes the direction feature and $\{b_{k1}, b_{k2}, \dots, b_{kJ}\}$ denotes the k th template in database. The accumulated distance $g_k(i, j)$ is calculated as follows:

Initial values:

$$\begin{cases} g_k(0,0) = 0 \\ g_k(i,0) = \infty \\ g_k(0,j) = \infty \end{cases}, \quad (3)$$

Recurrence formula:

$$g_k(i, j) = \min(g_k(i-1, j-1), g_k(i-1, j), g_k(i, j-1)) + \text{differnece}(a_i, b_{kj}), \quad (4)$$

where the difference is the chain code difference between chain code a_i and chain code b_{kj} . The difference is defined in Table 3.2.

Table 3.2 Difference between two chain codes.

$a_i \backslash b_{kj}$	0	1	2	3	4	5	6	7
0	0	1	2	3	4	3	2	1
1	1	0	1	2	3	4	3	2
2	2	1	0	1	2	3	4	3
3	3	2	1	0	1	2	3	4
4	4	3	2	1	0	1	2	3
5	3	4	3	2	1	0	1	2
6	2	3	4	3	2	1	0	1
7	1	2	3	4	3	2	1	0

Using above formula, the distance, $g_k(I,J)$, is calculated. After examining the distances with all templates, we find the symbol with the nearest distance and output the symbol as the recognized result.

After examining the distances with all templates, we find the symbol with the nearest distance and output the symbol as the recognized result.

3.5 Notation Recognition

After a stroke is recognized as a symbol, the notation recognizer will be conducted by combining symbols. The output symbol will be combined with previously recognized and unused symbols to form a notation based on the semantic information. There are 3 levels in the notation recognition. The other part of notation recognition is modification operation.

3.6.1 Bar Level

In music theory, a bar is a container containing notes, and a bar line is used to separate bars. In our system, the bar would be constructed automatically to hint the user. We combine unused symbols to form a bar line in the bar level. In each bar, the pseudo borders of the bar are pre-drawn in our system. We define the head and the end of the bar as the reactive area for combining symbols separately. By the shape of bar lines, we define the components in Table 3.3 to describe how to form a bar line.

Table 3.3 List of bar line with the set of components forming them.

Bar line name	Component
Single bar line ()	1 VLine
Double bar line ()	2 VLines
End bar line ()	3 VLines
Repeat sign line(:)	1 Dot, 0 or more VLines

3.6.2 Note Level

Notes are used to represent the relative duration and pitch of a sound in the music score. Symbols are combined to form a note in this level. By the composition of a note in music theory, there are three types of notes: determinable, uncertain and incomplete. Determinable note means that the numbers of symbols in it are fixed. The uncertain note means there are innumerable dots, heads, or flags in it. The incomplete note is a part of a certain note and recorded as a temporary note in this system.

When a new symbol is coming to this level, we would search the nearest uncertain or incomplete note. We do not have to search the determinable note, because it is impossible to add more symbols to it. If the distance to the nearest note is too large, we would construct a new empty incomplete note, and add the new symbol to it. Then check the symbols with rules in Table 3.4 [25] , which consists of three cases as follows:

1. If we find a match in the table, then update the note and set its type.
2. If we could not find a match in the table, and the set of symbols is a subset of a note, then we add the symbol to the note and set its type to be incomplete.
3. If we could not find a match and the set of symbols is not a subset of a note, then the new symbol would be discarded.

Table 3.4 List of notes with the set of symbols forming them .

Note name	Type	Components
FClef	Determinable	2 Dots, 1 FClefArc
	Determinable	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
GClef	Determinable	1 GClef
Natural	Determinable	1 LCheck, 1 NaturalRt
		1 LCheck, 1 8Rest
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, 0 or more Slash(es)
		1 or more BHead(s), 0 or more Dot(s), 0 or more Slash(es), 1 VLine, 0 or more UHook (s)
	Uncertain	1 or more BHead(s), 0 or more Dot(s), 1 StUHook, 0 or more Uhook(s), 0 or more Slash(es)
	Uncertain	1 or more BHead(s), 0 or more Dot(s), 1 Lcheck, 0 or more Slash(es), 0 or more UHook(s)
Whole rest	Uncertain	0 or more Dot(s), 1 WRest
Half rest	Uncertain	0 or more Dot(s), 1 HRest
Eight rest	Uncertain	1 8Rest, 0 or more Dot(s), 0 or more HLine(s)
Quarter rest	Uncertain	0 or more Dot(s), 1 QRest

3.6.3 Group Level

In music theory, when two or more notes with filled head and flags appear successively, we could group them using a beam to replace the flags. When playing the music score, the notes with beam should be more connected than non-beamed notes. As writing, users always draw a horizontal line across the notes to represent the grouping action. In the group level, we group the notes to form a beamed note.

3.6.4 Modification Operation

In this stage, we introduce the modification operations for editing the music score.. Instead of buttons, we take the advantage of pen based input method and provide some gestures for the modification operations.

We define two horizontal lines which are higher and lower than the music score, called “border lines.” The border lines are the writing borders in the system. The area between two border lines is called “writing area,” and the other areas are called “deleting area.” Writing in the writing area is valid, or it is an illegal operation. The concept of the modification operations contains two points: (1) if we want to move the location or pitch of a note, we could drag parts of a notation or whole notation to the destination directly. (2) If we want to delete some parts of the notation or the whole one, just drag it to the deleting area.

4. Experiment Result

Experiments are conducted to evaluate the performance of the proposed method. 13801 strokes, collected from 14 distinct writers, are used to test our algorithm. 6509 out of 13801 are taken as the training data. The remaining 7292 strokes are the testing data. Every stroke in the testing data is examined by symbol recognition. Finally, we could get the most similar symbol of the stroke as the output. In our experiments, a notebook (Intel T2300 CPU; only single cpu used; 1.66GHz; 1GB memory) and a digital tablet are used.

In order to measure the performance, we define the “precision” as follows:

$$\text{Precision} = \frac{\text{Correct}}{\text{Correct} + \text{Incorrect}}, \quad (5)$$

The precision for each symbol is shown in Table 4.1. The average precision for the symbols of our method is 98.35%, which is better than 97.54% of Miyao- Maruyama’s method [25].

Table 4.1 Precision of each symbol (*continued*).

Symbol name	Our method (%)	Miyao- Maruyama’s method(%)
WHead	98.46	97.49
BHead	96.70	99.85
StUHook	96.90	99.78
WRest	99.72	99.72
HRest	100.00	100.00
QRest	96.41	99.70
8Rest	95.88	100.00
Average	98.35	97.54

Table 4.1 Precision of each symbol. (*continued*).

Symbol name	Our method (%)	Miyao- Maruyama's method(%)
Dot	100.00	99.73
HLine	97.73	87.31
VLine	100.00	100.00
Slash	96.52	96.52
UHook	100.00	93.85
GClef	98.80	99.71
FClefArc	98.55	93.68
LCheck	99.71	90.81
NatureRt	97.87	100.00
Flat	98.69	100.00
WHead	98.46	97.49
BHead	96.70	99.85
StUHook	96.90	99.78
WRest	99.72	99.72
HRest	100.00	100.00
QRest	96.41	99.70
8Rest	95.88	100.00
Average	98.35	97.54

From the misclassified strokes, we find that the misclassification is due to that some users do not have any domain knowledge about the music theory, and they are not familiar with writing music notations. Sometimes they ignore the detail about the difference between symbols, like the curvature or the corners in a stroke. It makes some strokes ambiguous as trying to recognize. For example, if the user ignores the curvature between the slash and circle in BHead, the stroke is easily to be recognized as a WHead.

For the misclassified strokes, we provide the semantic correction to correct the mistakes. There are two rules defined in note level of notation recognition. First, while a WHead is misclassified to BHead and combine with a Half note, the system would convert BHead to WHead and do the combination. Second, while a BHead is misclassified to WHead and combine with Note with filled head, the system would convert WHead to BHead and do the combination. By the semantic correction, the precisions of WHead and BHead raise to 99.48% and 99.38%.

The total time of processing the 7292 testing data is about 157.38 seconds. Thus, the average processing time is about 0.0216 seconds per stroke. This is faster than Miyao-Maruyama's method which takes 0.0731 seconds per stroke by a PC (Pentium 4 CPU; 1.8GHz; 512MB memory). Thus, a user takes less waiting time while writing. Furthermore, our method is more suitable to migrate to the handheld devices with touched screen which have low computing power, and the user could compose a music score everywhere.

參考文獻

- [1] L. H. Chen and Y. P. Yin, "A System for On-line Recognition of Handwritten Mathematical Expressions," *Computer Processing of Chinese and Oriental Languages*, Vol. 6, No. 1, pp. 19-39, 1992.
- [2] G. Hutton, M. Cripps, D. Elliman, and C. Higgins, "A Strategy for On-line Interpretation of Sketching Engineering Drawings," *Fourth Intl. Conf. on Document Analysis and Recognition*, pp. 771-775, 1997.
- [3] Z. Lin, J. He, Z. Zhong, R. Wang, and H. Shum, "Table Detection in Online Ink Notes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, No. 8, pp. 1341-1346, 2006.
- [4] T. Kimura, A. Apte, and V. Vo, "Recognizing Multistroke Geometric Shapes : an Experimental Evaluation," *Proceedings of the ACM conference on User Interface and Software Technology (UIST'93)*, pp. 121-128, 1993.
- [5] M. Fonseca and J. Jorge, "Using Fuzzy Logic to Recognize Geometric Shapes Interactively," *The Ninth IEEE International Conference on Fuzzy Systems*, Vol. 1, pp. 291-296, 2000.
- [6] UML in Wikipedia : http://en.wikipedia.org/wiki/Unified_Modeling_Language
- [7] Object Management Group : <http://www.uml.org/>
- [8] C. Damm, K. Hansen, M. Thomsen, and M. Tyrsted, "Creative Object-Oriented Modeling : Support for Intuition, Flexibility and Collaboration in CASE Tools," *ECOOP 2000-Object-Oriented Programming : 14th European Conference*, Vol. 1850, pp. 27-43, 2000.
- [9] D. Rubine, "Specifying Gestures by Example," *Proceedings of SIGGRAPH'91*, pp. 329-337, 1991.
- [10] E. Lank, J. Thorley, S. Chen, and D. Blostein, "On-line Recognition of UML Diagrams," *The Sixth IEEE International Conference on Document Analysis and Recognition*, pp. 356-360, 2001.
- [11] Q. Chen, J. Grundy, and J. Hosking, "An E-whiteboard Application to Support Early Design-Stage Sketching of UML Diagrams," *IEEE Symposium on Human Centric Computing Language and Environments*, pp. 219-226, 2003.
- [12] G. Costagliola, V. Deufemia, and M. Risi, "A Multi-layer Parsing Strategy for On-line Recognition of Hand-drawn Diagrams," *IEEE Symposium on Visual Languages and Human-Centric Computing (VL-HCC'06)*, pp. 103-110, 2006.
- [13] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, 2006.
- [14] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, 2001.
- [15] J. Anstice, T. Bell, A. Cockburn and M. Setchell, "The Design of a Pen-Based Musical Input System," In *Proceedings of the 6th Australian Conference on Computer-Human Interaction (OZCHI 1996)*, Hamilton, New Zealand, pp. 260-267, Nov. 1996.
- [16] MagicScore Maestro software, DG software. (<http://www.dgalaxy.net/>)
- [17] Allegro, finale software. (<http://www.finalemusic.com/>)
- [18] S. E. George, "Online Pen-Based Recognition of Music Notation with Artificial Neural Networks," *Computer Music Journal*, vol. 27, no. 2, pp. 70-79, Jun. 2003.
- [19] A. Forsberg, M. Dieterich, and R. Zeleznik, "The music notepad," In *Proceedings of the 11th annual ACM symposium on User interface software and technology*, San Francisco, CA, USA, pp. 203-210, Nov. 1998.
- [20] Calligrapher, ParaGraph International, Inc. (<http://www.paragraph.com/>)
- [21] D. Rubine, "Specifying Gestures by Example," In *Proceedings of ACM SIGGRAPH '91*, New York,

USA, pp. 329-337, Jul. 1991.

- [22] E. Ng, T. Bell and A. Cockburn, "Improvements to a Pen-Based Musical Input System," OzCHI'98: The Australian Conference on Computer-Human Interaction, Adelaide, South Australia, pp. 178-185, Dec. 1998.
- [23] G. Taubman, "MusicHand: A Handwritten Music Recognition System," Honor thesis, Brown University, 2005.
- [24] S. Macé, E. Anquetil and B. Couïasnon, "A generic method to design pen-based systems for structured document composition : Development of a musical score editor," In Proceedings of the 1st Workshop on Improving and Assessing Pen-Based Input Techniques, Edinburgh, Scotland, pp. 15-22, Sep. 2005.
- [25] H. Miyao and M. Maruyama, "An Online Handwritten Music Score Recognition System," In Proceedings of the 17th International Conference on Pattern Recognition (ICPR 2004), Cambridge, United Kingdom, pp. 461-464, Aug. 2004.
- [26] S. Connell and A.K. Jain, "Template-based Online Character Recognition," Pattern Recognition 34(1), pp. 1-13. 2001.
- [27] X. Li and N. S. Hall, "Corner detection and shape classification of on-line handprinted Kanji strokes," Pattern Recognition 26(9), pp. 1315-1334. 1993.

國科會補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估

達成目標

未達成目標（請說明，以 100 字為限）

實驗失敗

因故實驗中斷

其他原因

說明：

2. 研究成果在學術期刊發表或申請專利等情形：

論文： 已發表 未發表之文稿 撰寫中 無

Z. H. Ou and L. H. Chen, 2011, "Hiding Data in Tetris", International conference on Machine Learning and Cybernetics 2011, Guilin China, 10-13 July.

專利： 已獲得 申請中 無

技轉： 已技轉 洽談中 無

其他：（以 100 字為限）

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）（以500字為限）

在計畫中我們今年提出了 UML 線上手寫辨識系統以及線上的樂譜手寫辨識系統。在 UML 線上手寫辨識系統中，根據我們的觀察，UML 的圖形多半為類似方形或是菱形的圖形，因此在本系統中利用決策樹的方式，來達到辨識的效果。首先我們擷取使用者輸入圖形的幾何特徵，來進行第一階段的分類。接著從輸入圖形擷取我們需要的特徵，和其所屬分類中各個圖形的特徵向量進行比對，即可得到最後的辨識結果。本系統之優點在於可以接受使用者任意筆順的輸入，並且辨識的方法較之前更為簡單有效，正確結果出現在前三名的辨識率為 91.24%。

從線上的樂譜手寫辨識系統樂譜是用來記錄樂曲的工具，作曲家常用其於創作與交流音樂。論文中，我們使用與在紙上相同的書寫方法，利用多筆劃組合出音樂符號。而筆劃的特徵有三種，高度，組成之基本圖形以及方向，可用來辨識出此筆劃所屬類型，再將其組合成所需要的音樂符號。本系統支援基本創作需要之全部音樂符號，辨識率為 98.35%，並且提供方便及完善的樂譜修改功能。

最後我們所發展的系統，可供學校於教學使用，另外所研發的辨識方法，也可供用於手持裝置上利用。

國科會補助專題研究計畫項下出席國際學術會議心得報告

日期：100年 8 月 1 日

計畫編號	NSC98-2221-E-009-117-MY2		
計畫名稱	應用手寫辨識於 UML 與樂譜作曲之系統		
出國人員姓名	歐占和	服務機構及職稱	國立交通大學博士生
會議時間	100年07月10日 至 100年07月13日	會議地點	Guilin, Guangxi, China
會議名稱	(中文)機器學習與控制國際研討會 (英文)International Conference on Machine Learning and Cybernetics		
發表論文題目	(中文)於俄羅斯方塊中的資訊隱藏 (英文)Hiding Data in Tetris		

一、參加會議經過

第一天：7月9日上午出發前往桃園國際機場。由於是第一次獨自一人跟團出國，慎重起見，提早到了集合的地點。在等待的過程中，有不少相互認識的教授們，在候機時就已經開始討論彼此研究的內容。雖然只是日常的閒聊，但從教授們的對話中，其實就能獲得不少資訊，也算是個好的開始。在飛機上認識了坐在旁邊的幾個學生，很恰巧地，這些人跟我剛好是同一個場次報告，彼此在飛機上就已經交換了不少心得，也讓我更加期待會議的到來。在旅館分配房間時，被分到與建國科技大學的程守雄教授同寢。程老師在這五天的行程中給予了我許多建議與鼓勵，讓我十分感激，也很慶幸這五天能與這樣優秀的老師相處。

第二天：早上的研習會(Tutorio)講的是研究所需的必要技巧以及如何有效率地發表研究成果，演講者是 IEEE Fellow、Witold Pedrycz 教授。能夠看到這樣了不起的研究人員，心中還是有些微的感動，也再次提醒自己研究的路途還很長，自己不了解的事情還非常的多。演講結束之後，由於研討會本日並沒有其他的活動，於是便回到房間準備隔日的報告。雖然與其他優秀的學者們互動也是很重要的事情，但還是希望能藉由充分的準備，讓更多人能了解我這次想要發表的內容。

第三天：今天中午 11 點半，在 OMB1 的場次中發表了自己的論文，題目是 Hiding data in Tetris。該場次的主席是台灣科技大學的蔡明忠教授，也是第一天在飛機上認識的兩位學生的指導教授。由於前兩天就已經有過不少互動，在報告的過程中少了一些緊張的感覺，很順利地將報告完成。該場次結束後，蔡教授私底下也提了不少問題與建議，非常感謝蔡教授的鼓勵。

第四天：昨天遇到了母校的辛華昀老師，聽辛老師說他目前正在海洋大學擔任助理教授。昨天因為還得要準備報告的關係，所以時間比較匆忙，也就沒有甚麼機會與辛老師敘舊。今天的議程結束之後，晚上跟辛老師一起吃飯，聊了一下彼此目前的研究方向，也認識了辛老師的學生。畢業很多年之後，在研討會這樣的場合裡，再次遇到母校的老師，心裡其實有非常多的感觸。

第五天：今早與室友程守雄教授一起去吃早餐時，在餐廳巧遇了陳錫明教授，在程老師的介紹下，與陳錫明老師一起用餐。席間、我有提及我的指導教授是陳玲慧老師，陳錫明教授便很開心地說他跟陳玲慧老師是好朋友。其實在出發前，我也已經聽陳玲慧老師多次提起陳錫明教授，原本在整個研討會的過程中都沒有甚麼機會認識，能在最後與陳錫明教授一起吃飯，並且接受教授的建議與鼓勵，真的是臨別前最棒的經驗了。

二、與會心得

能夠參與這種大型的研討會，最大的衝擊是感受到自己的渺小。看到這個世界上有這麼多的人發表了這麼多的論文，真的會讓人覺得學海無涯、唯勤是岸。但是與此同時，有這麼多人都在學術研究上頭努力，在心中其實也受到了許多的鼓舞與激勵。

此次研討會中，遇到了許多很好的老師，包括程守雄教授、辛華昀教授、蔡明忠教授、陳錫明教授，每一位老師都給予了我許多的教導與建議，我心中充滿了感激，也一定會將各位老師的忠告銘記在心。

這是我第一次單獨前往參加國際研討會，如果跟著同實驗室的其他同學一起參與研討會，大多數的時間都是跟同實驗室的人相處，就不可能有這麼豐富的體驗與收穫。回國以後那種在心中起伏的感動也很希望傳達給其他的學弟妹們，如果可以的話，自己一個人去參與這種研討會，想辦法主動認識一些研究領域中的先進，他們真的能教給我們很多很多的東西。

最後，感謝陳玲慧老師的鼓勵、指導，以及給我這個機會出席參加此次的機器學習與控制國際研討會。感謝國科會給予經費上的支持。感謝實驗室裡的同學，以及所有給予協助與幫忙過的人們，讓我得到了一個寶貴的經驗，非常謝謝大家。

三、考察參觀活動(無是項活動者略)

四、建議

如同與會心得中所提及的，單獨一人出席國際研討會的經驗非常寶貴，有勇氣獨自前往的學生應該受到鼓勵。此行認識了一位朝陽科技大學的碩一生，他一人獨自前往參與研討會，應該是個認真向學的學生。但是卻只有接受老師的補助，部分經費還要自行負擔，看起來其實還蠻委屈的。不知道貴會能否思考、對於獨行的學生、尤其是碩士生，在經費上給予較多的補助。以上建議，請貴單位參考。

五、攜回資料名稱及內容

Program for International Conference on 2011 Machine Learning and Cybernetics, International Conference on 2011 Wavelet Analysis and Pattern Recognition：研討會時程

表、專題演講與論文的摘要。

Proceedings of 2011 International Conference on Machine Learning and Cybernetics :
會議論文集。

六、其他



桂林機場外觀



與海洋大學辛華昀老師及其學生合影



論文報告



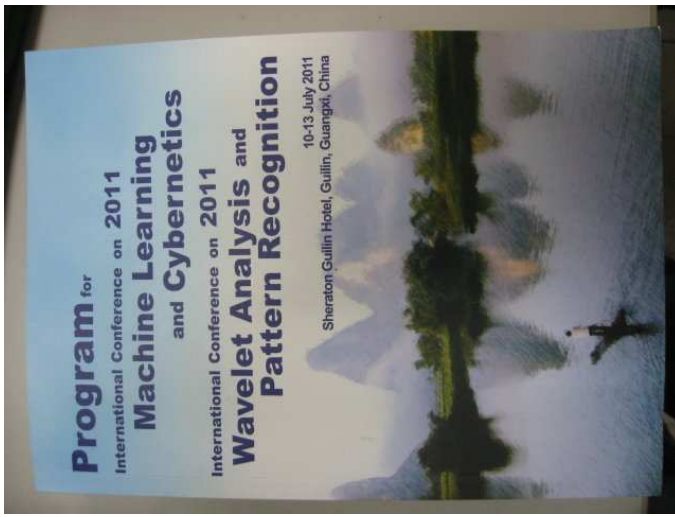
與同場次中的台灣學生及蔡老師合影



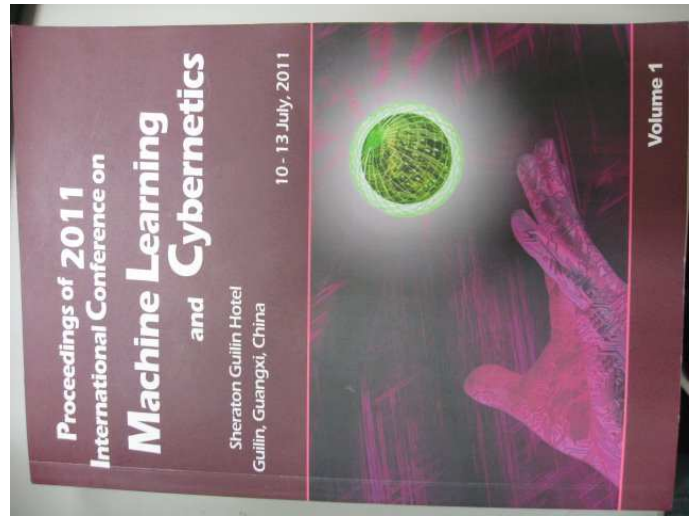
與台科大蔡明忠老師合影



與建國科技大學程守雄老師合影



攜回資料：會議時程



攜回資料：會議論文集