Interpretation Method of *Guqin* (Chinese Ancient Zither) Notation Based on Radical and Structural Analysis

NI En-zhi(倪恩志)[1*], JIANG Min-jun(蒋旻隽)[2], ZHOU Chang-le(周昌乐)[1]
1 School of Information Science and Technology, Xiamen University, Xiamen 361005, China
2 School of Computer Science and Information Engineering, Shanghai Institute of Technology, Shanghai 201418, China

Abstract: ***Guqin* music is a precious cultural heritage of China. The notation of *Guqin* is very special, which records its playing methods and techniques. For the purpose of preserving the *guqin* art, the digitalization of *guqin* notation and an interpretation method of *guqin* notation were conducted. By using this interpretation method, raw images of handwritten notations are transformed into structural data that can be processed and analyzed by computers easily. The method decomposes each single complex character of *guqin* notations into simple radicals and finds the structure of the character. According to the radicals and the structure, the character is interpreted into meaningful codes. The experimental results show our method is effective.**

Key words: *guqin notation*; *character interpretation*; *radical extraction*
CLC number: TP311 **Document code**: A
Article ID: 1672-5220(2013)01-0007-08

Introduction

Guqin music is an ancient art of China. It plays a key role in Chinese culture. Its instrument is a plucked seven-string musical instrument called "*guqin*" (see Fig. 1). Although it is very popular in ancient China, it is rarely known in China nowadays. In 2003, *guqin* and its music were proclaimed as one of the Masterpieces of the Oral and Intangible Heritage of Humanity by United Nations Educational Scientific and Cultural Organization. To preserve the precious cultural heritage, we focus on the digitalization of the *guqin* music. The *guqin* music has its own special notation (see Fig. 2), which is totally different from western musical notations. Instead of indicating note value, tempo or rhythm, *guqin* notation indicates the fingering of playing *guqin*. The paper focuses on automatic interpretation of *guqin* notation. In this way, we can transform a raw image of *guqin* notation into structural data which are easier to analyze and process.

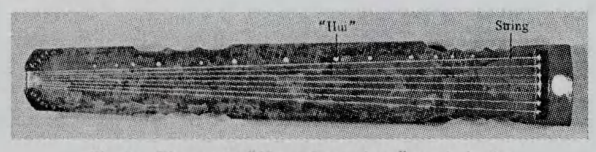

Fig. 1 The *guqin* "Hewu Longxiang" made in the Southern Song Dynasty (1127–1279 A. D.)

A *guqin* notation consists of many characters. These characters are called as reduced characters, since they are composed of radicals of Chinese characters. Most of *guqin* notations are handwritten. The method presented in the paper includes the following three steps. The flowchart of the method is shown in Fig. 3.

(1) Character decomposition

Decompose the reduced character into radicals.

(2) Layout matching

Determine the optimal layout template and the sections of radicals in the layout template.

(3) Character interpretation

Fig. 2 A page of ancient *guqin* notations

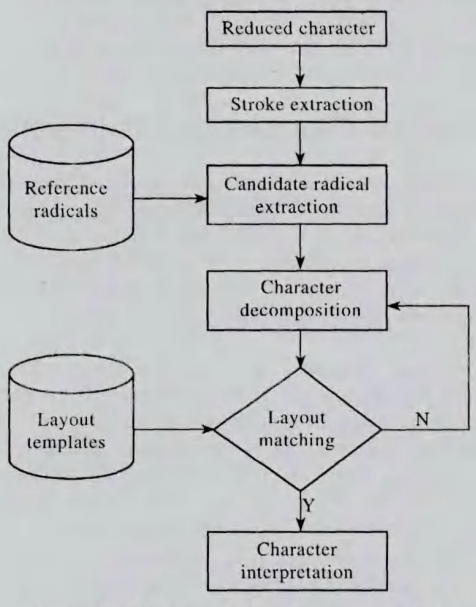

Fig. 3 The flowchart of the interpretation method

Interpret the character according to its radicals and layout template.

The first step is the key to the *guqin* notation interpretation. However character decomposition (or radical extraction, radical matching) is still a challenge so far. There are some related researches for character decomposition. In the field of offline character recognition, Wang *et al.* [1] proposed a recursive hierarchical scheme for radical extraction. Chung *et al.* [2] proposed a radical segmentation algorithm based on snakes. These two methods utilized gaps within a character to decompose the character. Shi *et al.* [3-4] used active shape

Received date: 2012-12-12
Foundation item: National Natural Science Foundation of China (No. 6097507)
* Correspondence should be addressed to NI En-zhi, E-mail: nienzhi@gmail.com

models to extract radicals from Chinese characters, but this method only extracted the peripheral radicals. Chellapilla *et al.* [5] designed a radical-at-location radical recognizer which could recognize radicals at a specific location in a character. Ni *et al.* [6] presented a radical cascade classifier, and each radical cascade classifier detected a specific radical within the whole character image. In the field of online character recognition, Ma *et al.* presented an appearance-based radical recognition[7] which recognized radicals from left-right and top-bottom structural characters, and a statistical-classification-based method[8] for detecting special radicals from special-structural characters. Lü *et al.* [9] segmented radicals by stroke projection. Chou *et al.* (tree search)[10] and Xiao *et al.* (dynamic programming)[11] exploited stroke order, but stroke order was not available for offline character and often varied with different persons. The studies in Refs. [10-12] are limited to extract the front radical and the rear radical in a character, because these two kinds of radicals are relative stable.

In this paper, a two-stage heuristic search is proposed for reduced character decomposition. A reference radical is described with strokes and stroke relations. We propose a novel distance measure to calculate the similarity between a reference radical and a real radical. All similar radicals are extracted as candidate radicals. The first-stage heuristic search is to search for candidate radicals among a character. The second-stage heuristic search is to find a combination of radicals that best matches the input character among candidate radicals. After character decomposition, the layout matching will discover the structure of the character and the correspondences between radicals and sections. At last, the character is interpreted.

1 Reduced Character and Its Interpretation

A typical reduced character is composed of five types of radicals which correspond to different types of *guqin* fingerings: *finger* indicates which finger of left hand stops the string; *position* indicates the position of finger of left hand; *left_move* indicates the movement of left hand; *pluck* indicates the plucking technique of right hand; *string* indicates the string number.

A reduced character is given in Fig. 4 (a). The *pluck* determines the structure of a reduced character. Every radical of this kind corresponds to at least one layout template, and the other radicals are arranged in the character according to the layout template. Figure 4 (b) is the layout template of the reduced character in Fig. 4 (a). This layout template is determined by the *pluck* "勾" of the reduced character in Fig. 4 (a). There are seven sections in the template. And *finger* lies in section 1; *position* lies in section 2 and section 3; *left_move* lies in section 4 and section 5; *pluck* "勾" lies in section 6; and string lies in section 7.

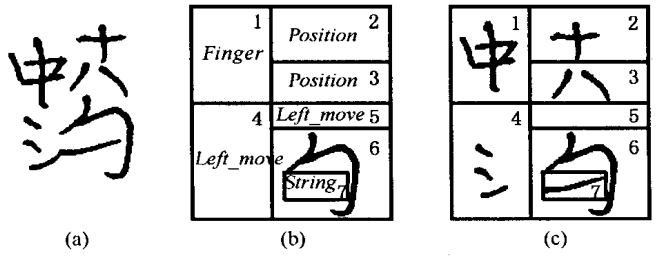

Fig. 4 Interpretation of the reduced character: (a) a reduced character; (b) the layout template of the pluch "勾"; (c) the correspondences between the radicals and sections of the layout template

The correspondences between the radicals of the reduced character in Fig. 4(a) and the sections of the layout template in Fig. 4(b) are shown in Fig. 4(c). We can interpret the reduced character according to the radicals and the sections after their correspondences are established. The reduced character in Fig. 4 (a) is interpreted as moving the middle finger of the left hand down to the position of the 10th "hui" and the 8th "fen" while pulling the first string inward with the middle finger of the right hand.

2 Reference Radical

Before radical matching, the reference radicals of all radical categories must be defined. The reference radicals are described with strokes and stroke relations. A stroke is depicted with its type and attributes. In terms of the types of feature points of strokes, 10 stroke relations are presented in this section.

2.1 Stroke types and attributes

The concept of "stroke" has various definitions in literatures. "Stroke" in the paper refers to the line stroke. We define 7 stroke types. They are dot, horizontal stroke, vertical stroke, slash, back slash, tick, and hook[11] (see Fig. 5). Each type has different tolerances of orientation and bend angles. There is much distortion on handwritten characters, so the tolerance is useful for stroke extraction. The chosen attributes are length, orientation, and the positions of start, end, and center points.

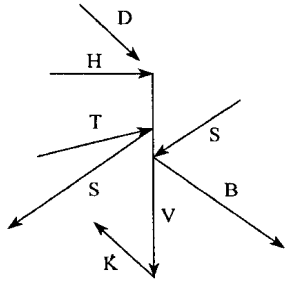

Fig. 5 Stroke types (D = dot, H = horizontal, V = vertical, S = slash, B = back slash, T = tick, K = hook)

2.2 Stroke relations

We define 10 stroke relations in light of the types of feature points of strokes. They are 7 basic relations (see Fig. 6) and 3 compound relations. R_1 and R_2 describe the situation in which the two strokes have a turn point, while R_3 describes the two strokes having a crossover point. R_4-R_7 represent the situation that the two strokes have a fork point (the fork point could not actually exist). The criterions of judging R_1-R_7 are listed as below (S_1RS_2 represents stroke S_1 has relation R to S_2).

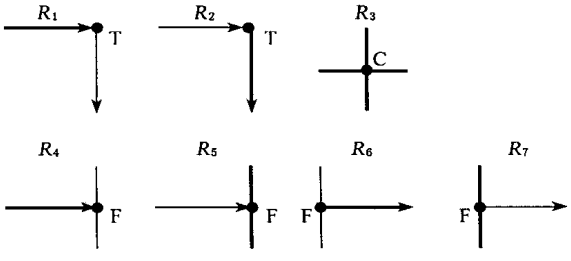

Fig. 6 Basic stroke relations (the target strokes are drawn with thick line, T = turn point, C = crossover point, F = fork point)

S_1 R_1 S_2: the end point of S_1 is (or is close to) the start point of S_2.

S_1 R_2 S_2: the start point of S_1 is (or is close to) the end point of S_2.

S_1 R_3 S_2: S_1 and S_2 have one same crossover point.

S_1 R_4 S_2: the fork point can not actually exist. There are two non-standard situations of R_4 in Fig. 7. In Fig. 7(a), the fort point does not exist. And in Fig. 7(b), the fork point is replaced by a crossover point. The criteria for R_4 are $(AC + AD) > (BC + BD)$, and BC and BD are in different sides of AB.

S_1 R_5 S_2, S_1 R_6 S_2, and S_1 R_7 S_2: refer to S_1 R_4 S_2.

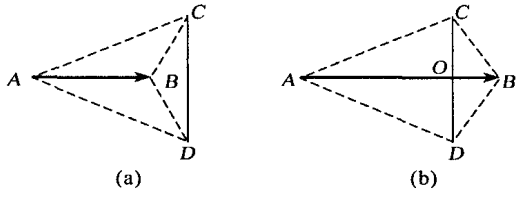

Fig. 7 R_4 in two non-standard situations

The compound relations are defined as $R_8 = \overline{R_1 + R_2 + R_3 + R_4 + R_5 + R_6 + R_7}$, $R_9 = \overline{R_1 + R_2 + R_3}$, and $R_{10} = \overline{R_3}$. R_8 is a parallel relation (see Figs. 8(a) and (b)) and R_9 is a side relation (see Figs. 8(c) and (d)). R_{10} is a loose constraint against R_3. In this paper, we use $R(i, j)$ ($R(i, j) \in \{R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, R_9, R_{10}\}$) to denote the relation between stroke i and stroke j.

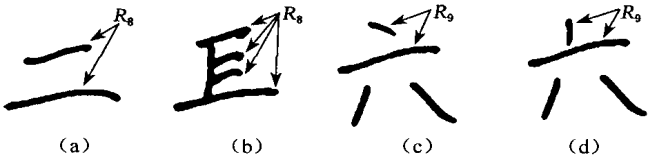

Fig. 8 Examples of stroke relations R_8 and R_9

2.3 Radical distance measure

A good distance measure is essential to radical matching. Although there are many distance measures for character matching, they are unavailable for radical matching. Most of distance measures for character matching take the lengths and positions of strokes into consideration. However, as the sizes and positions of radicals in different characters can be different (see Fig. 9), their lengths and positions of strokes can be different too. Figure 9(a) is radical "七", and Figs. 9(b) and (c) are two reduced characters containing radical "七". Therefore, the lengths and positions of strokes can not be used for radical distance measure.

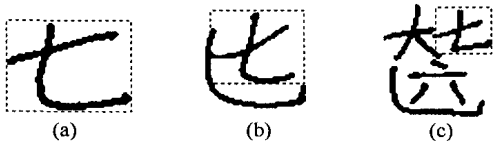

Fig. 9 Different sizes and positions of radicals in different characters

As mentioned above, radical distance measure will be difficult. To solve the problem, we propose three new distances for radical matching.

2.3.1 The distance of relative direction

Denote the two strokes in the same radical by S_1 and S_2. The relative direction is the direction from a point on S_1 (S_2) to a point on S_2 (S_1). The relative direction is approximately invariant to the change of the size and position of a radical. So it can be used as a measure for radical matching. For convenience, we only consider the relative directions from the start, center, and end points of one stroke to the start, center, and end points of another stroke. Figure 10 shows these relative directions.

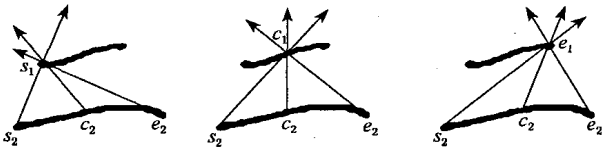

Fig. 10 Relative direction distance measure

Denote the two strokes in the reference radical by S_1^r and S_2^r (the superscript r denotes the reference stroke), and denote the two input strokes by S_1^i and S_2^i (the superscript i denotes the input stroke). Suppose that S_1^i is already determined to be the corresponding stroke of S_1^r. The distance d_θ between the relative direction of S_2^i to S_1^i and the relative direction of S_2^r to S_1^r is calculated by

$$d_\theta((S_2^i, S_1^i), (S_2^r, S_1^r)) = \frac{\sum_{p\in\{s_2, c_2, e_2\}} \sum_{q\in\{s_1, c_1, e_1\}} |\theta_{pq}^r - \theta_{pq}^i| \cdot D(p^r, q^r) \cdot D(p^i, q^i)}{\pi \sum_{p\in\{s_2, c_2, e_2\}} \sum_{q\in\{s_1, c_1, e_1\}} D(p^r, q^r) \cdot D(p^i, q^i)}, \quad (1)$$

$$D(p, q) = \begin{cases} 1, & d_{pq} \geqslant d_{\min}, \\ 0, & d_{pq} < d_{\min}, \end{cases} \quad (2)$$

where s, c, and e represent the start point, center point, and end point of the stroke respectively; θ_{pq} is the direction from p to q; d_{pq} is the distance between p and q; and $d_{\min}$ is the minimum effective distance. Because of the distortion of handwritten characters, the direction does not make sense when the distance of two points is less than $d_{\min}$. Therefore $D(p, q)$ is equal to 0 if $d_{pq} < d_{\min}$. As a result, the related part of p and q is ignored in this case.

2.3.2 The distance of relative length

The lengths of the strokes are possibly different when the same radicals lie in different characters, but the ratios of the lengths of two strokes in the same radicals are usually unchanged. For example, the second stroke of the radical "三" is usually shorter than the others, and the third stroke is usually longer than the others. So we propose relative length measure in light of this observation. The relative length is the length of one stroke relative to the other stroke. The changes of the width and height of a radical have different influence on different types of strokes. For example, the change of width has greater influence on horizontal strokes than vertical strokes. Therefore, the distance of relative length is separated to two parts: one is the distance of relative length on x-axis; the other is the distance of relative length on y-axis. Every type of strokes has different impact coefficients on these two distances of relative length.

Denote the two strokes in the reference radical by S_1^r and S_2^r, and denote the two input strokes by S_1^i and S_2^i. Suppose that S_1^i is already determined to be the corresponding stroke of S_1^r. The distance d_l between the length of S_2^i relative to S_1^i and the length of S_2^r relative to S_1^r is calculated by

$$d_l((S_2^i, S_1^i), (S_2^r, S_1^r)) = \frac{1}{2}(d_{lx}((S_2^i, S_1^i), (S_2^r, S_1^r)) + d_{ly}((S_2^i, S_1^i), (S_2^r, S_1^r))), \quad (3)$$

$$d_{lx}((S_2^i, S_1^i), (S_2^r, S_1^r)) = \lambda_{lx}\frac{\left|\Delta x_2^i - \widehat{\Delta x_2^i}\right| + \varepsilon}{\widehat{\Delta x_2^i} + \varepsilon},$$

$$\text{if } d_{lx} > 1,\ d_{lx} = 1, \tag{4}$$

$$d_{ly}((S_2^i, S_1^i), (S_2^r, S_1^r)) = \lambda_{ly}\frac{\left|\widehat{\Delta y_2^i} - \Delta y_2^i\right| + \varepsilon}{\widehat{\Delta y_2^i} + \varepsilon},$$

$$\text{if } d_{ly} > 1,\ d_{ly} = 1, \tag{5}$$

$$\widehat{\Delta x_2^i} = \frac{\Delta x_2^r + \varepsilon}{\Delta x_1^r + \varepsilon} \cdot \Delta x_1^i, \tag{6}$$

$$\widehat{\Delta y_2^i} = \frac{\Delta y_2^r + \varepsilon}{\Delta y_1^r + \varepsilon} \cdot \Delta y_1^i, \tag{7}$$

where Δx and Δy are the lengths of the projections of the stroke on x-axis and y-axis respectively; ε is a small positive constant; λ_{lx} is the product of two impact coefficients on x-axis of the types of S_1^r and S_2^r; and λ_{ly} is the product of two impact coefficients on y-axis of the types of S_1^r and S_2^r. The impact coefficients are defined as: dot $\{x:0, y:0\}$; horizontal stroke $\{x:1, y:0\}$; vertical stroke $\{x:0, y:1\}$; slash $\{x:0.8, y:0.8\}$; back slash $\{x:0.8, y:0.8\}$; tick $\{x:0.4, y:0.4\}$; and hook $\{x:0.4, y:0.4\}$.

2.3.3 The distance of relative distance

The relative distance is the distance between two strokes relative to the distances between the other strokes. The relative distance is stable. For example, the distance between the first stroke and the second stroke of the radical "三" is stable relative to the distance between the first stroke and the third stroke. The distance of two strokes is calculated by the distance between two points on these two strokes. For convenience, we just consider the distances from the start, center, and end points of one stroke to the start, center, and end points of another stroke.

Let $I_{i,j}$ be the distance between S_i and S_j. Denote the strokes in the reference radical by $S_1^r, S_2^r, \cdots, S_n^r$, n is the number of strokes in the radical and $n \geqslant 2$. And denote the input strokes by $S_1^i, S_2^i, \cdots, S_m^i$, $2 \leqslant m \leqslant n$. Suppose that $S_1^i, S_2^i, \cdots, S_{m-1}^i$ are already determined to be the corresponding strokes of $S_1^r, S_2^r, \cdots, S_{m-1}^r$. The distance d_d of the relative distance between S_m^i and S_k^i and the relative distance between S_m^r and S_k^r is calculated by

$$d_d((I^i_{m,k}, I^i_{1,2,\cdots,k-1}), (I^r_{m,k}, I^r_{1,2,\cdots,k-1})) = \begin{cases} \varepsilon_d, & k = 1,\ m = 2, \\ \dfrac{\sum\limits_{p \in \{sm, cm, em\} q \in \{sk, ck, ek\}} \dfrac{\left| r_{m-1} \cdot d^r_{pq} - d^i_{pq} \right|}{r_{m-1} \cdot d^r_{pq}} \cdot D(p^r, q^r) \cdot D(p^i, q^i)}{\sum\limits_{p \in \{sm, cm, em\} q \in \{sk, ck, ek\}} D(p^r, q^r) \cdot D(p^i, q^i)}, & 1 \leqslant k < m \leqslant n,\ m > 2, \end{cases} \tag{8}$$

$$r_m = \frac{1}{m-1} \cdot \sum_{1 \leqslant k < m} \frac{\sum\limits_{p \in \{sm, cm, em\} q \in \{sk, ck, ek\}} \dfrac{d^i_{pq}}{d^r_{pq}} \cdot D(p^r, q^r) \cdot D(p^i, q^i)}{\sum\limits_{p \in \{sm, cm, em\} q \in \{sk, ck, ek\}} D(p^r, q^r) \cdot D(p^i, q^i)}, \quad m > 1, \tag{9}$$

$$D(p, q) = \begin{cases} 1, & d_{pq} \geqslant d_{\min}, \\ 0, & d_{pq} < d_{\min}, \end{cases} \tag{10}$$

where ε_d is a small nonnegative constant. It is hard to calculate the relative distance accurately when the distance of two points is less than $d_{\min}$. Therefore $D(p, q)$ is equal to 0 if $d_{pq} < d_{\min}$. As a result, the related part of p and q is ignored in this case.

The three distances above are based on the relations among the strokes in the same radical. We call the weighted sum of the three distances "relative distance". The stroke relative distance is defined as

$$d_{sr}(S_m^i) = \sum_{1 \leqslant k < m} [\lambda_\theta d_\theta(S_m^i, S_k^i) + \lambda_l d_l(S_m^i, S_k^i) + \lambda_d d_d(S_m^i, S_k^i)], \tag{11}$$

where λ_θ, λ_l, and λ_d are the coefficients of the three distances. Although the length and position of the stroke can not use for radical matching, the orientation of the stroke is still available. The stroke orientation distance is defined as

$$d_{so}(S^i) = \frac{2}{\pi}\left|\theta^i - \theta^r\right|. \tag{12}$$

Finally, the radical distance is defined as

$$d_r = \sum_{1 \leqslant k \leqslant n}\left[\frac{2}{n-1}\lambda_{rr} d_{sr}(S_k^i) + \lambda_{ro} d_{so}(S_k^i)\right], \tag{13}$$

where n is the number of strokes of the radical; λ_{rr} is the coefficient of the stroke relative distance; and λ_{ro} is the coefficient of the stroke orientation distance.

3 Two-Stage Heuristic Search

3.1 Radical matching

3.1.1 Candidate stroke extraction

The characters must be thinned and decomposed to line segments, and meanwhile feature points are extracted. A line segment is a piece of stroke between two feature points. After that, we get the characters described with line segments and feature points.

The method of candidate stroke extraction is the same to Ref. [13], except the used attributes of strokes. In Ref. [13], stroke length, stroke orientation, bend angle, and the position of center point are considered while selecting candidate strokes from line segments. However, as to radical matching, the attributes are not all available. A radical is more flexible than a character. The positions and sizes of the same radical in different characters can be different. There are only two attributes that are stable, namely stroke orientation and bend angle. Therefore, stroke orientation and bend angle are the only used attributes in candidate stroke extraction.

3.1.2 Heuristic search for radical matching

Depth-first tree search algorithm with heuristic rules is

applied to radical matching. The depth of the search tree is equal to the number of stokes in the reference radical. Each level of the search tree corresponds to each stroke of the reference radical. Each node represents a candidate stroke. The node is expanded with candidate strokes of next reference stroke. The relations between the new expanded node and all its ancestors and possible descendants are checked. The nodes satisfying the stroke relation constraints are appended to *OPEN* list. The nodes that have been expanded are moved to *CLOSED* list. All goal nodes (every reference strokes have been matched) in *CLOSED* list are back-tracked so that the paths are generated. Every path represents a matched radical. The radical distances are calculated by Eq. (13). Finally output the several paths with the smallest radical distances.

The radicals of reduced characters can be divided into surrounding radicals and non-surrounding radicals. As to surrounding radicals, other radicals can lie in their rectangular areas. As to non-surrounding radicals, no other strokes can lie in their areas. For example, the radical "勹" in section 6 in Fig. 4 (b) is a surrounding radical and the radical "十" in section 2 is a non-surrounding radical. Therefore, a goal node should meet two conditions: the depth of the node must be equal to the number of strokes of the reference radical; there should be no other strokes within the area of the radical represented by the node if the reference radical is a non-surrounding radical.

The algorithm is described as below.

Algorithm 1 Heuristic search for radical matching

n: the number of strokes of the reference radical;

$R(i, j)$: the relation between the ith stroke and the jth stroke in the reference radical;

$C_s(i)$: the set of candidate input strokes of the ith reference stroke;

$\max(d_\theta)$: the maximal allowed relative direction distance;

p: the number of output candidate radicals.

Method:

Step 1 Generate a search tree with a root node N_0. N_0 is stored in *OPEN* list, and the *CLOSED* list is empty.

Step 2 If *OPEN* is empty, go to **Step 7**.

Step 3 Select the last node in *OPEN*. Denote the node by N_k, k is the depth of the node. Move N_k to *CLOSED*.

Step 4 If $k=n$, go to **Step 2**.

Step 5 Expand N_k with the strokes in $C_s(k+1)$. Denote a new node generated in expansion by N'. If the following three conditions are satisfied, append N' to *OPEN*.

(1) The relations between N' and N_k and all ancestors of N_k are consistent with the corresponding relations in the reference radical.

(2) The relative direction distance of N' with every ancestor is less than $\max(d_\theta)$.

(3) For each i $(k+1<i<n)$, there exists at least one stroke in $C_s(i)$ whose relation to N' is consistent with $R(i, k+1)$.

Step 6 Go to **Step 2**.

Step 7 Search for the goal node in *CLOSED*. Back-track the paths from those nodes to N_0. Calculate the radical distances of those paths by Eq. (13). Output the p paths with the smallest radical distances.

3.2 Reduced character decomposition

Reduced character decomposition is to find a radical combination that best matches the input character among the candidate radicals. A* algorithm is adopted for this search. The depth of the search tree is equal to the number of line segments in the input character. Each node represents a candidate radical. And the depth of a node is equal to the number of line segments which have been covered by the node and its ancestors. According to the characteristics of the reduced character, we give some constraints in this search. A reduced character has one and only one *pluck* (there are several compound radicals which are composed of two *plucks*, and we treat them as single *pluck*). So the nodes at the first level of the search tree are limited to the candidate *pluck* and the nodes below the first level can not be *pluck*. To avoid the paths which have the same radicals being searched and a radical being appended to a path repeatedly, we sort the candidate radicals in the descending order of the numbers of the strokes of the radicals. When expanding a radical (node) which is not *pluck* in the search tree, only those radicals which are behind of the radical can be appended. In addition, the numbers of radical categories occur in the character are limited. And most of them can occur only once. So we limit the times every radical category can occur. Only the radicals which have not reached the limit can be appended to the search paths.

Denote the node by N, the path from root node to N by $P(N)$, the cost of N by $c_n(N)$, the cost of $P(N)$ by $c_p(N)$, the estimated remaining cost by $c_h(N)$. We think the radicals which have more strokes should be given higher priorities. The more strokes in the radical, the more constraints the radical has. The radical with more strokes has higher confidence if those constraints are satisfied. Therefore we use the weighted radical distance to be the cost of a node. The cost of a node c_n is calculated by

$$c_n(N) = w_k d_r(N), \tag{14}$$

where k is the number of the strokes of the radical. The values of w_k are determined by the curve $y=x^a$ ($a=\log_{15}0.1$) (see Fig. 11). $c_p(N)$ is the sum of c_n of the nodes in the same path.

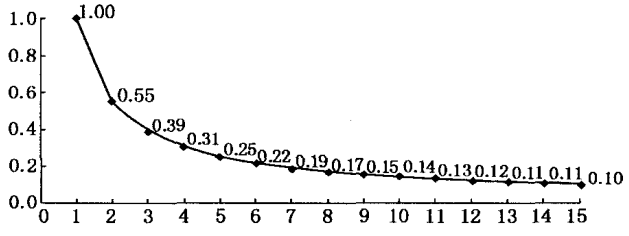

Fig. 11 The curve of the values of w_k

If the estimated remaining cost is a lower bound of the true remaining cost, the A* algorithm is guaranteed to find the global optimal solution. The algorithm for estimating $c_h(N)$ is presented as follows. It guarantees that $c_h(N)$ is not greater than the true estimated remaining cost.

Algorithm 2 Estimation of $c_h(N)$

m_c: the number of line segment of the character;

$S(N)$: the set of line segments covered by $P(N)$;

$C_r(N)$: the set of candidate radicals covering line segments which don't belong to $S(N)$. $C_r(N)$ is sorted in ascending order of the values of $wc_n(N)/|S(N)|$.

Method:

Step 1 $m=|S(N)|$, $c_h(N)=0$.

Step 2 While $m<m_c$, let the first radical in $C_r(N)$ be N', and the number of line segments covered by N' be m'. If $m'>m_c-m$, $c_h(N)=c_h(N)+w(m_c-m)c_n(N')/m'$ and return $c_h(N)$; otherwise $c_h(N)=c_h(N)+wc_n(N')$, $C_r(N)=C_r(N)-\{N'\}$, $m=m+m'$.

Based on the discussion above, we propose the heuristic search algorithm for character decomposition as below.

Algorithm 3 Heuristic search for character decomposition

C_r: the candidate radical set which is sorted in descending

order of the numbers of the strokes of the radicals.

Method:

Step 1 Generate a search tree with a root node N_0. N_0 is stored in *OPEN* list, and the *CLOSED* list is empty.

Step 2 If *OPEN* is empty, search for the node which and its ancestors cover the most line segments of the character and has a smallest cost in *OPEN* and *CLOSED*. Back-track the path from the node to N_0. Return the path.

Step 3 Select the last node in *OPEN*. Denote the node by N. Move N to *CLOSED*.

Step 4 If N and its ancestors cover all line segments of the character, go to **Step 7**.

Step 5 Expand N with the radicals in C_r. Denote a new node generated in expansion by N'. If the following four conditions are satisfied, append N' to *OPEN*.

(1) If N is N_0, N' should be *pluck*; otherwise, N' should not be *pluck*.

(2) If N is not *pluck*, N' should be behind of N in C_r.

(3) Every line segment which covered by N' is not covered by the ancestors of N'.

(4) In the path from N_0 to N', the number of radicals which belong to the same category of N' does not exceed the limit of the occurrence of this radical category in a character.

Step 6 Sort the *OPEN* list in descending order of the sums of the costs of the paths c_p and the estimated remaining costs c_h. Go to **Step 2**.

Step 7 Back-track the path from N to N_0. Return the path.

4 Layout Template Matching and Character Interpretation

After character decomposition, we have got radicals of a character. As mentioned above, the same radicals in different sections of a layout template indicate different meanings. As a result, the correspondences between its radicals and the sections of its layout template need to be established before interpreting a reduced character. Layout template matching is to find the best matching layout template of the input character and to establish the correspondences between the radicals and the sections of the template.

As mentioned in section 2, the layout template of a reduced character is determined by its *pluck*. So we collected the layout templates in terms of *plucks*. A layout template can be denoted by

$$Tem = \{Sec_1, Sec_2, \cdots, Sec_n\},$$

where *Sec* is the section of the template. A section can be denoted by

$$Sec = \{x_{Sec}, y_{Sec}, C_{Sec}, R_{Sec}\},$$

where (x_{Sec}, y_{Sec}) is the center point of the section; C_{Sec} is the set of radical categories which are allowed to lie in the section; and R_{Sec} is the set of position relations between the section and the other sections. We define 8 position relations: top, bottom, left, right, top left, top right, bottom left, and bottom right. A reduced character can be denoted by

$$Char = \{Rad_1, Rad_2, \cdots, Rad_k\},$$

where *Rad* is the radical which is contained in *Char*. A radical can be denoted by

$$Rad = \{x_{Rad}, y_{Rad}, c_{Rad}\},$$

where (x_{Rad}, y_{Rad}) is the center point of the radical, and c_{Rad} is the category which the radical falls into.

Definition 1 *Char* is matched with *Tem* when the condition is satisfied, that for every *Sec* in *Tem*, there is exactly one *Rad* in *Char* such that $c_{Rad} \in C_{Sec}$ and the *Rad* satisfies every relations in R_{Sec} and no unmapped element remains in both *Char* and *Tem*.

In the phase of layout template matching, a reduced character is matched with every layout template. If there is no match, the procedure will return to the phase of character decomposition and find another combination among candidate radicals. If there are several matched layout templates, the one with the maximal similarity will output to next phase. Let f be the function from *Char* to *Tem*, and the similarity between *Char* and *Tem* is calculated by

$$\frac{1}{\sum_{Rad \in Char,\, Sec = f(Rad)} \sqrt{(x_{Rad} - x_{Sec})^2 + (y_{Rad} - y_{Sec})^2} + \varepsilon}. \quad (15)$$

An example of layout template matching is presented in Fig. 12.

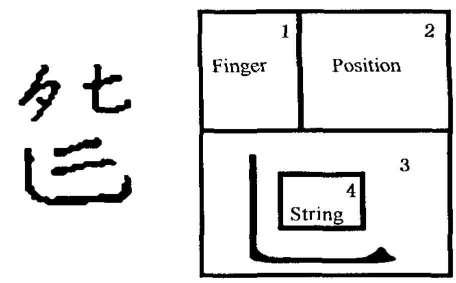

Fig. 12 An illustration of template matching (⅃ is a pluck)

Now we can interpret the reduced character. A reduced character can be interpreted as the playing method which it indicates. The playing method reads as a string of radical names sorted by section numbers (please refer to the example in section 2. 1). People can know what the character means by reading the string of names. Or a reduced character can be interpreted and encoded as a series of octal numbers[14], which is convenient for computer processing.

5 Experimental Results

We collect 500 reduced character categories for the experiment which are composed of 42 frequently used radical categories. These radical categories are listed in Table 1. The reference radicals and layout templates are defined manually. Five sets of these characters written by 5 writers respectively are used as the test set. The test set consists of 2 500 characters and 10 405 radicals. Some samples are shown in Fig. 13. Some experimental results are shown in Fig. 14 (the right-most column is the printed image generated after interpretation). The detailed results are shown in Tables 2 and 3.

Table 1 The radical categories used in the experiment

No. of strokes	Radicals
1	一
2	二 八 十 厂 卜 亻 [illegible] [illegible]
3	三 大 扌 丁 乚 厶 艹 氵
4	六 七 夕 木 勹 尸
5	五 九 中 乇 [illegible] [illegible] [illegible]
6	四 巛 北
7	马 早 [illegible] [illegible] [illegible] [illegible]
8	[illegible] [illegible]
10	[illegible]

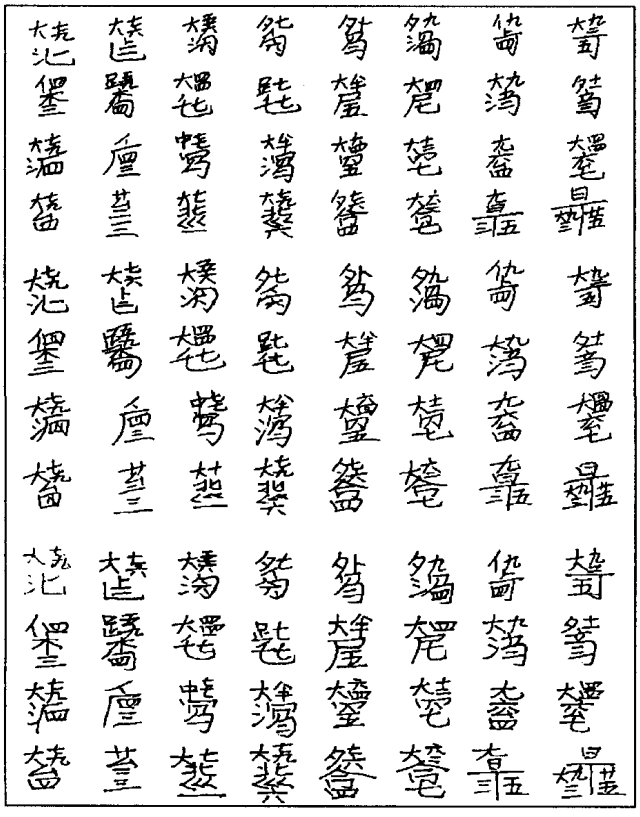

Fig. 13 Some reduced character samples

The number of strokes of a radical has great impact on radical extraction. When radicals have few strokes, few constraints of stroke relations exist. That will cause too many candidate radicals. The correct radical is hard to extract in such cases. On the other hand, the radicals with many strokes have tight constraints of stroke relations, but handwritten distortion often causes some of these constraints mismatched. Note that the reference radicals we choose are not just those with average number of strokes. The numbers of strokes of the selected radicals range from 1 to 10 (especially, the radicals having only 1 stroke is very difficult to deal with). Even though, our method yields good results as Table 2 showed.

The number of radicals of a character also has great impact on character interpretation. The more radicals a character has, the more difficult it can be interpreted. More radicals increase the complexity of stroke extraction, radical extraction, and character interpretation. The results in Table 3 show the interpretation method is effective, although the correct interpretation rates are not that ideal. One major reason is that the result of radical extraction and character interpretation greatly depends on the result of stroke extraction. For the characters composed of many strokes, their strokes are more difficult to be extracted correctly. In this case, it is hard to interpret these characters accordingly. In despite of this deficiency, we implemented an effective method towards the interpretation of reduced character.

Fig. 14 Some experimental results

Table 2 The result of character decomposition

No. of strokes	Correct extracted radicals	Real radicals	Correct radical extraction rate/%
1	166	190	87.37
2	1 345	1 520	88.49
3	2 322	2 600	89.31
4	2 750	3 120	88.14
5	1 329	1 505	88.31
6	739	795	92.96
≥7	565	675	83.70

Table 3 The result of character interpretation

No. of radicals	Correct interpreted characters	Real characters	Correct character interpretation rate/%
2	203	235	86.38
3	322	430	74.88
4	609	835	72.93
5	447	710	62.96
≥6	138	290	47.59

6 Conclusions

Guqin music is a precious cultural heritage of China. To preserve and develop the *guqin* art, its digitalization is becoming more and more important. This paper presents an interpretation method of *guqin* notation. By this method, we are able to transform the *guqin* notation from the image to the structural data, which can be processed and analyzed by computers.

The interpretation of *guqin* notation is a challenge so far. We adopt a "divide and conquer" strategy for this purpose. The method proposed in this paper consists of three major steps, namely character decomposition, layout matching, and character interpretation. For reduced character decomposition, a novel radical distance measure and an algorithm of two-stage search are proposed in this paper. For layout matching, we propose a formula to calculate the similarity between a template and a character. Based on the extracted radicals and the layout template, the character is interpreted into its playing method or octal numbers. We perform experiments on handwritten *guqin* notations, and the experimental results show our method is effective.

Based on the accomplishment of this paper, more researches can be further carried out such as *guqin* notation database, *guqin* notation analysis and retrieval.

References

[1] Wang A B, Fan K C. Optical Recognition of Handwritten Chinese Characters by Hierarchical Radical Matching Method [J]. *Pattern Recognition*, 2001, **34**(1): 15-35.

[2] Chung F L, Ip W W S. Complex Character Decomposition Using Deformable Model [J]. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 2001, **31**(1): 126-132.

[3] Shi D M, Ng G S, Damper R I, *et al.* Radical Recognition of Handwritten Chinese Characters Using GA-Based Kernel Active Shape Modeling [J]. IEE Proceedings of Vision, Image & Signal Processing, 2005, **152**(5): 634-638.

[4] Shi D M, Damper R I, Gunn S R. Offline Handwritten Chinese Character Recognition by Radical Decomposition [J]. *ACM Transactions on Asian Language Information Processing*, 2003, **2**(1): 27-48.

[5] Chellapilla K, Simard P. A New Radical Based Approach to Offline Handwritten East-Asian Character Recognition [C]. Proceedings of the 10th International Workshop on Frontiers in Handwriting Recognition, La Baule, France, 2006: 261-266.

[6] Ni E Z, Zhou C L, Jiang M J. A Radical Cascade Classifier for Handwritten Chinese Character Recognition [J]. *Journal of Software*, 2012, 7(10): 2294-2300.

[7] Ma L L, Liu C L. On-line Handwritten Chinese Character Recognition Based on Nested Segmentation of Radicals [C]. Proceedings of Chinese Conference on Pattern Recognition, Nanjing, China, 2009: 1-5.

[8] Ma L L, Delaye A, Liu C L. Special Radical Detection by Statistical Classification for On-line Handwritten Chinese Character Recognition [C]. Proceedings of the 12th International Conference on Frontiers in Handwriting Recognition, Kolkata, India, 2010: 501-506.

[9] Lü X Q, Huang D S, Song E M, *et al.* One Radical-Based On-line Chinese Character Recognition (OLCCR) System Using Support Vector Machine for Recognition of Radicals [C]. Proceedings of the 1st International Conference on Bioinformatics and Biomedical Engineering, Wuhan, China, 2007: 558-561.

[10] Chou K S, Fan K C, Fan T I, *et al.* Knowledge Model Based Approach in Recognition of On-line Chinese Characters [J]. *IEEE Journal on Selected Areas in Communication*, 1994, **12**(9): 1566-1575.

[11] Xiao X H, Dai R W. On-line Handwritten Chinese Character Recognition Directed by Components with Dynamic Templates [J]. *International Journal of Pattern Recognition and Artificial Intelligence*, 1998, **12**(1): 143-158.

[12] Lay S R, Lee C H, Cheng N J, *et al.* On-line Chinese Character Recognition with Effective Candidate Radical and Candidate Character Selections [J]. *Pattern Recognition*, 1996, **29**(10): 1647-1659.

[13] Liu C L, Kim I J, Kim J H. Model-Based Stroke Extraction and Matching for Handwritten Chinese Character Recognition [J]. *Pattern Recognition*, 2001, **34**(12): 2339-2352.

[14] Zhou C L, Qi J F, Zhuang X X, *et al.* Method of Processing Jianzi Images on Windows [J]. *Computer Engineering*, 2010, **36**(5): 279-281. (in Chinese)