

# 基于笔顺自由及连笔的联机手写汉字识别<sup>①</sup>

## On-Line Chinese Character Recognition Based on Free Ordering and Connective Strokes

崔景楠 邢长征 (辽宁工程技术大学 计算机应用技术专业 辽宁 葫芦岛 125105)

**摘要:** 提出了一种基于笔顺重排算法的手写汉字识别。将手写汉字的可见线段和不可见线段进行联合编码,并采用了一种基于单字切分及基本笔顺表的识别方法。首先将单字分解为部件,根据分解的结构,对字典进行粗略的过滤,将字典中不符合待识别汉字拆分结构的字排除,然后根据笔划编码进行识别,有效提高了笔划的匹配速度,较好地解决了联机手写汉字识别中连笔及笔顺自由问题。

**关键词:** 连笔 笔顺 联机手写汉字识别 笔顺重排 单字拆分

联机手写汉字识别技术涉及模式识别、图象处理、自然语言理解、人工智能等多种学科,是一门综合性技术,在中文信息处理、办公自动化、机器翻译、等高技术领域,都有着重要的实用价值和理论意义。

### 1 引言

为解决连笔书写汉字的轨迹编码问题,本文将落笔到提笔的可见线段称为 **stroke** 线,从提笔到下一轮的落笔所经过的不可见线段称为 **connection** 线,将输入汉字所有的 **stroke** 线和 **connection** 线都作为汉字书写轨迹而进行笔画编码。这样将 **stroke** 线和 **connection** 线一同编码,在进行汉字识别时等于匹配汉字的行笔方向和位置等信息,而避开了落笔和提笔等信息,能有效地解决汉字连笔书写的问题。

由于将 **stroke** 线和 **connection** 线的引入,匹配只依据整体的行笔方向、走势、位置等信息,而没有用到落笔、提笔的信息,很自然地解决了连笔的问题。甚至可以很好地识别完全连笔的汉字,即一笔书写而成的汉字<sup>[1]</sup>。

上述算法的明显缺点是它的连笔自由是建立在笔顺不自由的基础上的。虽然,通过向字典里添加笔顺变种的模版,可在一定程度上解决笔顺自由的问题,但是,会带来字典过于庞大、匹配速度减慢等问题,而且也不可能穷尽所有可能的笔顺变化。因此,必须

加以改进才能用于笔顺自由的识别系统上。

文献[1]提出:“为了实现笔顺自由,待识汉字同标准模式匹配时,可以首先找出它们之间的正确的笔画对应关系,把输入汉字的笔顺和标准模式的笔顺调整一致,就可以同时解决笔顺自由和连笔自由的问题”,将待识别汉字的与标准模版笔顺的匹配问题转换为指派问题。

这种方法的好处是“能较好地同时解决笔顺自由和连笔自由的问题”。缺点是:待识别汉字必须针对字典中的每一个标准模版,改变其笔顺以获得最小的匹配距离,这就大大增加了识别时的系统开销,降低了识别效率。

本文提出了一种基于单字切分及基本笔顺表的识别方法。首先将单字分解为部件,根据分解的结构,对字典进行粗略的过滤,将字典中不符合待识别汉字拆分结构的字排除,然后根据笔划编码进行识别,有效的提高了笔划的匹配速度。

### 2 局部笔顺重排

如果字典中只存放一种笔顺,或者不是存放所有的笔顺变化而只存放几种笔顺的模版,为了实现笔顺自由,需要对笔顺进行重新排列。但是由于汉字字型复杂,手写汉字更是千变万化,对整个汉字进行笔划重新排列是不现实的。因此,本文采用对汉字先分块,

<sup>①</sup> 收稿时间:2008-10-11

再排序的方法，将一个汉字分成若干 **Block**，先再 **Block** 中对笔划进行排序，然后将多个 **Blocks** 按照位置排序。

根据对手写汉字书写习惯的研究，通常书写汉字时的首笔都是正确的，所以在排序时，被识别汉字的首笔即是排序后的首笔。这样可以准确、高效重新排列笔顺。

### 2.1 整字分块

在预处理阶段，我们通过归一化，滤波，平滑，笔划提取等步骤，将手写的自由曲线处理成易于识别的直线段的集合。如图 1 所示：

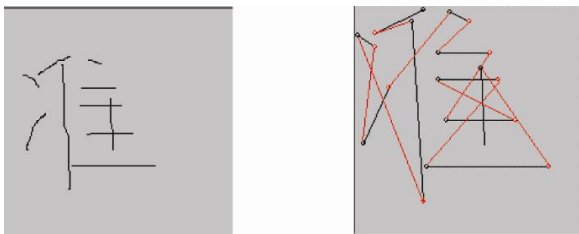


图 1 手写输入的笔划和预处理之后用于识别的笔划

图 1 中，黑色表示 **stroke** 线，红色表示 **connection** 线。预处理之后，待识别的汉字即变成一系列首尾相接的直线。再对直线集合中的 **stroke** 线在坐标轴水平方向和垂直方向上进行投影。根据汉字的结构特征，切分的位置通常不会降笔划切断，在对笔划进行投影时，我们忽略与切分方向垂直的信息。因此，我们做如下处理：

水平方向乘以  $width / height \times a$

垂直方向乘以  $height / width \times a$

其中，**a** 为常数，**width**、**height** 分别为笔划包罗矩形的一半的高度和宽度。得出汉字在水平和垂直方向上的曲线分布情况，如图 2 所示：

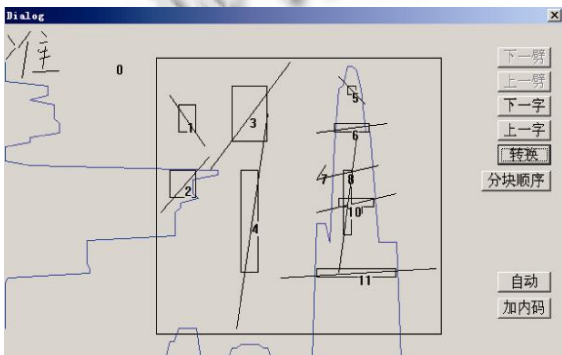


图 2 水平方向和垂直方向上的投影曲线

由于笔划间的长短影响笔划与笔划之间的位置关系，但是笔划之间的实际位置关系通常与笔划的首尾段无关，所以在投影时，是以笔划的中点为中心，笔划的包罗矩形的长宽一半为长和宽的矩形在水平、垂直方向投影，这样就能去处在书写时由于笔划的长短对笔划间的相对位置产生的影响。得出分布曲线后，根据曲线的分布，在曲线的波谷位置，将汉字分为几个部分。

根据对汉字手写习惯的统计，大部分的连笔发生在部件之内，如图 3。因此，部件与部件之间的连接笔划只有一笔。而且我们在投影时，以笔划的中点为中心，笔划的包罗矩形的长宽一半为长和宽的矩形在水平、垂直方向投影，所以对于连接部件之间的笔划，投影到坐标轴时仅仅保留了中间一部分信息，对汉字部件的拆分并无实质的影响。例如“识”字，左边的“讠”和“只”中间有一笔连接，如图 4 所示。我们根据曲线的分别情况，在曲线波谷的位置将汉字拆分为几个 **Block**。

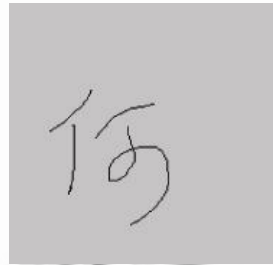


图 3 部件内的连笔情况

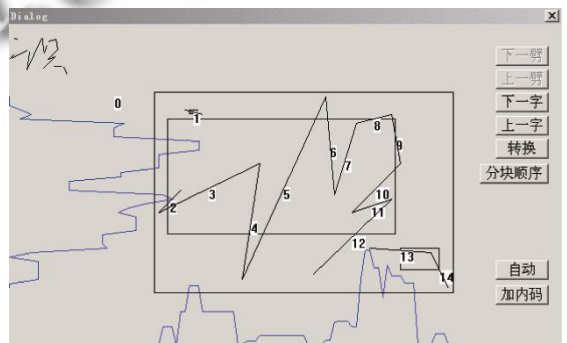


图 4 连笔汉字的投影图

根据待识别汉字被分解的结构，我们便可以将字典中与之相差较大的汉字排除。例如处理“识”字时，便可以将上下结构的汉字全部排除。减少了标准模版的数量，提高了识别效率。

## 2.2 笔顺排列基本规则

这里我们介绍如何将块内的笔划排序。我们列出了所有的笔划，并且根据他们之间的位置关系决定他们的先后顺序。

### 2.2.1 笔划的分类

表 1 列出了汉字笔顺的基本规则<sup>[6]</sup>。笔划集合中与表 1 的顺序不一致时，便按照汉字基本笔顺表将之纠正过来。对于可能存在的一些例外情况，我们可以在程序中用 case 语句特殊处理。

表 1 笔顺的基本规则

Rule	Example	Order of stroke
First horizontal, then vertical	十	一 十
First left-falling, then right-falling	人	丿 人
From top to bottom	三	一 二 三
From left to right	州	丨 丿 ㇇ 州 州
First outside, then inside	月	丿 月 月 月
Finish inside, then close	四	丨 冂 四 四 四
Middle, then the two sides	小	丨 小 小

在定制笔顺表之前，应该先定制基本笔划表。表 2 列举了 7 种笔划，所有的笔划兜被映射成这 7 种笔划。笔划的类型根据笔划长度，笔划的斜率以及笔划和笔划之间的向量角来决定。确定笔划类型的算法如图 1 所示：

```
/*变量定义 T 为笔划长度门限 T var 两个笔划之间的向量夹角 T var 笔划最大斜率 T var 笔划最小斜率*/
```

```
T : threshold of stroke length (100 pixel)
T var : the maximum angle between two vectors in a stroke
T var: maximum slope of stroke
T var : minimum slope of stroke
for all strokes in the set of strokes Q = {s1,s2... sn}{
/*定义笔划 “点” */
if ( length of si < T L ) label of si = DOT
/*定义笔划 “钩” */
```

```
else if (vector angle of si > T var )
label of si = HOOK
/*定义笔划 “竖” */
else if(slope of si < T min )
label of si = VERTICAL
/*定义笔划 “横” */
else if(slope of si > T max )
label of si = HORIZONTAL
/*定义笔划 “捺” */
else if( x-value of si > 0)
label of si = RIGHTFALLING
/*定义笔划 “撇” */
else label of si = LEFTFALLING
for all strokes in the set of strokes Q={s1,s2... sn }
/*定义笔划 “折” */
if ( label of si = VERTICAL &&
label of sj = HORIZONTAL &&
Intersect( si , sj ) = TRUE )
Label of si + sj = TURNING
}
```

图 1 笔划分类算法

表 2 基本笔划表

Label	Stroke
DOT	丶
HORIZONTAL	一
VERTICAL	丨
LEFTFALLING	丿
RIGHTFALLING	㇇
TURNING	冂

### 2.2.2 笔划重排算法

根据笔划之间的位置关系对笔划做出了更细致的划分。我们定义了笔划结合表(表 3)，并且定义了他们之间的相对位置关系。对于一个笔划集合  $Q = \{s_1, s_2, \dots, s_n\}$ ，其中  $s_i$  将根据与其他笔划的位置关系在表中定位。算法如图 2 所示：

```
/*定义 Q 为笔划集合 L 为表序号集合*/
Q = {s1,s2... sn } : the set of strokes
L = {l1,l2... ln } : the set of labels
/*确定笔划在基本表中的序号*/
1. For all strokes in Q = {s1,s2... sn } , determine
```

the label of  $s_i$ .

/\*确定笔划在子表中的序号\*/

2. Subdivide the label of  $s_i$  ( $s_i$  may have several labels).

/\*删除在基本表和子表中不一致的笔划\*/























3. Delete the labels of  $s_i$  which do not keep consistency.

/\*循环, 直到笔划序号  $s_i$  唯一\*/

4. Repeat 3, until the label of  $s_i$  is unique.

图 2 笔划细分类算法

表 3 笔划细分类

Base	Label	Ex.	Description
DOT			Not divided
HORIZONTAL	NONE		General
	CROSS		Intersection
	LEFT		Left of other strokes
	RIGHT		Right of other strokes
	CLOSE		口 or 日
VERTICAL	NONE		General
	T		Vertical of 王
	CROSS		Intersection
	LEFT		Vertical of 口
HOOK	NONE		General
	CROSS		Intersection
	CROSS 2		Hook of 子
LEFTFALLING	NONE		General
	SYM		Symmetry
	SYM2		Above the —
	RIGHT		Intersection
RIGHTFALLING	NONE		General
	SYM		Symmetry
	SYM2		Above the —
TURNING	NONE		General
	F		Turning of 民

当所有笔划都在表中定位完成后, 我们定义笔划的顺序。我们在两个笔划之间定义" $<$ "。" $<$ "应该是反

身的, 反对称的和可传递的。我们在这里所考虑的笔划顺序如同是将笔划集合  $Q$  中的元素进行排序。" $<$ "的定义规则如表 4 所示。

表 4 笔顺信息表

Ahead	Later
DOT	HORIZONTAL_CLOSE
HORIZONTAL_CROSS	VERTICAL_CLOSE
HORIZONTAL_CROSS	HOOK_CROSS
VERTICAL_NONE	HORIZONTAL_LEFT
VERTICAL_NONE	HORIZONTAL_RIGHT
VERTICAL_T	HORIZONTAL_CROSS
VERTICAL_LEFT	TURNING_NONE
VERTICAL_NONE	LEFTFALLING_SYM
VERTICAL_NONE	RIGHTFALLING_SYM
HOOK_NONE	HORIZONTAL_LEFT
HOOK_NONE	HORIZONTAL_RIGHT
HOOK_CROSS2	HORIZONTAL_CROSS
HOOK_NONE	LEFFEFALLING_SYM
HOOK_NONE	RIGHTFALLING_SYM2
LEFFEFALLING_NONE	RIGHTFALLING_NONE
LEFFEFALLING_SYM	HORIZONTAL_NONE
LEFFEFALLING_SYM2	HORIZONTAL_CROSS
LEFFEFALLING_RIGHT	RIGHTFALLING_NONE
RIGHTFALLING_SYM2	HORIZONTAL_NONE
RIGHTFALLING_SYM2	HORIZONTAL_CROSS
TURNING_F	VERTICAL_NONE
TURNING_F	HOOK_NONE

### 2.3 块与块的排列

被识别汉字被切分为 Block 之后, Block 和 Block 之间的顺序依照汉字的书写习惯, 即: 从左到右, 从上到下, 先里后外排列即可。

## 3 编码与识别

按照表 4 的笔划顺序基本表以及块间排列顺序, 我们用排序算法得到了待识别汉字的笔划序列  $S$ , 则  $S = (s_1, s_2, \dots, s_n)$   $n$  为待识别汉字的笔划数目,  $T$  为标准模板的笔划序列  $T = (t_1, t_2, \dots, t_m)$   $m$  为标准模板的笔划数。 $S$  和  $T$  的匹配算法采用文献[1]提到的 DP 算法。即通过如下的 DP 迭代式计算出一条最佳的输入序列

和标准序列的匹配路径。

$$D(s_i, t_i) = \min \{ D(s_{i-1}, t_{i-1}) + 2 \cdot d(s_i, t_i), D(s_i, t_{i-1}) + d(s_i, t_i), D(s_{i-1}, t_i) + d(s_i, t_i) \} \quad (1)$$

待识别笔划序列与标准序列之间的最小距离即为识别结果。

#### 4 性能分析和测试实验

基于部件内的笔顺重排, 通过将单字拆分成若干部件, 分析待识别汉字的部件间分布机构, 缩小了标准字典中模版匹配的个数, 减少了匹配次数, 缩短了识别时间。

本文采用 **visual studio 2005** 编写联机手写汉字输入识别系统, 利用前面提出的方法从 **8 套 GB2312** 的 **6763** 个单字样本中, 将笔划进行随机打乱。用没有进行笔划排序的联机识别算法进行识别, 识别率为

**47.27 %**; 进行单字切分, 粗分类, 部件内部排序, 部件之间的排序后, 识别率为 **98.22 %**。在配置为 **Pentium 4 CPU, 3.2GHz, 512MB** 内存的 **pc** 机上对这些汉字进行识别, 平均耗时为 **203 字/秒**。

#### 参考文献

- 1 曹喆炯, 王永成. 笔顺连笔自由的联机手写汉字识别. 计算机工程与应用, 2005, 29: 167 - 169.
- 2 余楚中, 赵学军, 彭静, 等. 联机手写体汉字识别中的笔划分类及笔划识别. 重庆大学学报, 1998, 2(2): 131 - 134.
- 3 Sang Ok Koo Hyun Gyu Jang, Kwang Hee Won, Soon Ki Jung. Automatic Stroke Extraction and Stroke Ordering Based on TrueType Font EG UK Theory and Practice of Computer Graphics.