[19]中华人民共和国专利局

[11] 公开号 CN 1102487A

[12] 发明专利申请公开说明书

[21]申请号 94108937.1

[51]Int.Cl[5]

[43]公开日 1995 年 5 月 10 日

G06F 3/00

[22]申请日 94.8.4

[30]优先权

[32]93.10.22[33]US[31]141,137

[71]申请人 财团法人工业技术研究院

地址 中国台湾

[72]发明人 周胜邻 余孝先 赖源正

[74]专利代理机构 中国专利代理(香港)有限公司

代理人 王 岳 叶恺东

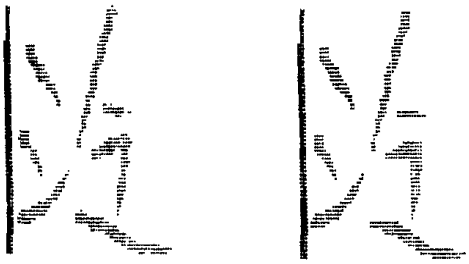
G06F 15/20

说明书页数: 附图页数:

[54]发明名称 汉字图像的圆滑处理方法

[57]摘要

汉字图像的圆滑处理方法，该方法包括步骤：（a）建立所有像素在水平及垂直四个方向上 run 的信息；（b）空白 run 填补，检测每一个空白 run 看是否满足填补的条件，如果满足则将该空白 run 做部分或全部填补；（c）小空白三角填补，检测每一个由横、直黑 run 夹住而形成的空白三角区域，检查看是否满足填补条件，若是则将该三角区域填补成黑点；及（d）黑 run 删除，检测每一个黑 run，检测是否满足删除条件，若是则将该黑 run 及周边的短小黑 run 删除。



1.汉字图像的圆滑处理方法，该汉字已经数字化且变成二次元像素，各像素被定义成为一空白或黑色像素，其特征在于该方法包括下列步骤:

(a)建立所有像素在水平及垂直四个方向上run的信息，其中，该run为像素点在四个方向上相同像素值的延伸长度;

(b)空白run填补，检测每一个空白run 看是否满足填补的条件，如果满足则将该空白run做部分或全部填补;

(c)小空白三角填补，检测每一个由横、直黑run夹住而形成的空白三角区域，检查看是否满足填补条件，若是则将该三角区域填补成黑点；及

(d)黑run删除，检测每一个黑run,检测是否满足删除条件，若是则将该黑run及周边的短小黑run删除。

2.根据权利要求1 所述的汉字图像的圆滑处理方法，其特征在于还包括:

(e)在删除或填补同时更新相关像素点的run的信息；及

(f)重复(a)-(e)步骤若干次，或直到没有删除与填补操作发生。

3.根据权利要求1 所述的汉字图像的圆滑处理方法，其中该空白run填补步骤包含以下的步骤:

(a)利用该run信息找到欲处理的横向空白run;

(b)针对该横向空白run,检查以下的条件:

(i)RUN_H < MAX_RUN_TO_BE_FILLED_H,其中RUN_H表示正检测run的长度，而MAX_RUN_TO_BE_FILLED_H则为一预设常数，用以限制欲填补的横向空白run的长度，所有长度均是以像素(pixel)为单位;

(ii) RUN_H < p_RUN_H 或 RUN_H < n_RUN_H,其中 p_RUN_H与n_RUN_H分别为夹住该横向空白run前后的黑run的长度; 及

(iii) image [r-1][c-1] = 1且

hf_run [r-1][c-1] > (RUN_H + 2),

或 image [r+1][c-1] = 1且

hf_run [r+1][c-1] > (RUN_H + 2),

(c)如果(b)的条件满足, 则将检测的横向空白run 中被黑点像素上、下包住的像素点填补成黑色像素;

(d)更新与填补像素点相邻的像素点的run信息;

(e)利用run信息找到直向空白run;

(f)针对直向空白run,检查以下的条件:

(i)RUN_V < MAX_RUN_TO_BE_FILLED_V,其中RUN_V表示正检测run的长度, 而MAX_RUN_TO_BE_FILLED_V则为一预设常数, 用以限制欲填补的直向空白run的长度, 所有长度均是以像素为单位;

(ii) RUN_V < p_RUN_V 或 RUN_V < n_RUN_V,其中 p_RUN_V与n_RUN_V分别为夹住空白run上下的黑run的长度;

(iii) image [r-1][c-1] = 1且

vd_run [r-1][c-1] > (RUN_V + 2),

或 image [r+1][c-1] = 1且

vd_run [r+1][c-1] > (RUN_V + 2),

其中image代表字符影像阵列, image = 1表示黑色像素点;

(g)如果(f)的条件满足, 将检测的直向空白run 中被黑像素点左右包住的像素填补成黑点; 及

(h)更新相关像素点的run信息。

4.根据权利要求1 所述的汉字图像的圆滑处理方法, 其中该空白run 填补的步骤包括横向空白run的平底V字深谷填补, 其特征在于还

包括如下步骤:

(a)对于横向空白run，检测其左上，与右上(或左下与右下) 的向左与向右黑run数行，对于每一行，分别求出左上与右上黑run的长度run0与run1;

(b)累计sum_run0与sum_run1,其中sum_run0与sum_run1分别表示run0与run1的累加值;

(c)检测以下的条件限制:

(i)run0＞ V_RUN且run1＞ V_RUN,

或(ii)sum_run0＞ SUM_V_RUN,

且 sum_run1＞SUM_V_RUN,

其中V_RUN与SUM_V_RUN为预设常数;

(d)若(c)所述的条件满足，则整个V 字深谷中的空白点均填补成黑点，否则继续检测上一行左上与右上黑run，重复上述步骤; 及

(e)若需要，更新相关像素的run信息。

5.根据权利要求1 所述的汉字图像的圆滑处理方法，其中该空白run 填补的步骤包括横向空白run的平底V字形深谷填补，其特征在于还包括如下步骤:

(a)对于直向空白run，检测其左上，与右上(或左下与右下) 的向上与向下黑run数行，对于每一行，分别求出左上与右上黑run的长度run0与run1;

(b)累计sum_run0与sum_run1,两者分别表示run0与run1的累加值;

(c)检测以下的条件限制:

(i)run0＞ V_RUN且run1＞ V_RUN,

或(ii)sum_run0＞ SUM_V_RUN,

且 sum_run1＞SUM_V_RUN,

其中V_RUN与SUM_V_RUN为二预设常数;

(d)若(c)所述的条件满足，则整个开口向右V 字深谷内的空白点均填补成黑点，否则继续检测上一行的run0与run1，重复上述步骤;及

(e)若需要，更新相关像素的run信息。

6.根据权利要求1 所述的汉字图像的圆滑处理方法，其特征在于该小空白三角填补步骤包含步骤如下:

(a)利用run信息找出被直向与横向黑run 夹角而形成的三角空白区域，假设OH与OV分别表示形成三角区域的横向与直向的黑run;

(b)检测小空白三角填补条件，如下:

(i)小空白三角两边外围紧邻的黑run，其长度必须分别大于构成小空白三角的黑的长度，此条件主要是希望处理的笔划为斜笔划;

(ii)小空白三角的边长必须小于某一预设常数TRIANGLE_SIZE;及

(iii)OH′与OV′的夹角大于135°，其中H′，与V′分别为0点沿边缘路径往H与V点方向m点外的像素点，m为一常数，此条件限制OH与OV同属一笔划直线上;

(c)若(b)的条件满足，则OH与OV夹角所形成的小空白三角区域中的空白点均填补成黑点;

(d)若有填补动作，则更改相关点的run信息。

7.根据权利要求1所述的汉字图像的圆滑处理方法，其中该黑run删除，步骤包含对横向黑run的删除，且黑run的起点为(r,c)，长度为RUN,其特征在于包含步骤如下:

(a)针对被检测的横向黑run，定义其上的黑run长度u_RUN 如下:

$$u\_RUN = \begin{cases} 0 & \text{若 image [r-1][C] =0 或 image [r-1][C+RUN-1] =0} \\ & \text{否则} \\ hf\_run[r-1][C]+hb\_run[r-1][C]-1 & \end{cases}$$

其中hf_run为横向向前run,hb_run为横向向后run;

(b)定义下面黑run的长度d_RUN如下:

$$d\_RUN = \begin{cases} 0 & \text{若 image [r-1][C] =0 或 image [r-1][C+RUN-1] =0} \\ & \text{否则} \\ hf\_run[r+1][C]+hb\_run[r+1][C]-1 \end{cases}$$

(c)检测以下删除的条件:

(i) RUN < MAX_RUN_TO_BE_DELETED

其中MAX_RUN_TO_BE_DELETED为一预设常数,

(ii) u_RUN≥ 2 * run 且u_RUN > LEN_TO_DELETE_NGHB

d_RUN≥ 2 * run 且d_RUN > LEN_TO_DELETE_NGHB

其中LEN_TO_DELETE_NGHB为预设常数;

(d) w > W,其中w为欲删除黑run以下(或上) 的短小黑run行数, W为预设常数, 通常小于5;

(e) w 行的黑run长度必须均在RUN的两端之内;

(f) 若(c)的条件满足, 则将检测的黑run连同其下的W行黑run删除, 及

(g)更新相关像素的run信息。

8.根据权利要求1所述的汉字图像的圆滑处理方法, 其中该黑run删除, 步骤包括直向黑run 的删除, 其特征在于步骤与条件与权利要求6的横向黑run删除步骤基本相同。

汉字图像的圆滑处理方法

本发明涉及手写汉字图像边缘圆滑处理，以改善字符图像质量，以便在随后的诸如字符特征分析、特征提取、字符识别等步骤的处理中能大幅度地提高其效率。更具体的说，本发明涉及经数字化的手写汉字图像，处理的目标是消除掺杂信号，使字符边缘圆滑，而处理的目的则是希望提高字符质量，以利于随后的处理。本发明所采用的方法是RUN(即图像中连续的黑点或白点)为基础，而所根据的则是汉字构成的横、直、斜笔划特征。

随着汉化电脑的普及，便捷的汉字输入需求日益迫切。由于近年来手写汉字识别技术进展颇快，使得自然的汉字输入方法燃起了一线生机。不过，为了达到普遍化的阶段，识别率仍需不断地改善。在追求更高识别率的过程中，除了识别技术的进展外，改善输入的字符图像的质量也是很重要的一环。而对于汉字图像的前处理，尤其是字缘圆滑处理，还没有一令人满意的结果。本发明就是针对上述不足之处提出的一种解决的方案。

本发明是以RUN 为基础手写的汉字边缘圆滑技术，而此技术的形成则是依赖于汉字的构成特征，即汉字主要由横、直与斜笔划组成。所谓一个 RUN是指字符图像中具有相同值(黑或白) 的连续像素。因此一个RUN 可以是直线或横线。本发明的第一个步骤就是找出字符图像中所有的直RUN与横RUN，然后再依据其中的每一个像素，求出该点在四个方向的延伸长度(具有与该点相同值的长度) 。有了这些run信息之后，就可以很轻易地找出所有的空白run与黑run，针对空白run

与黑run,分别检测填满或删除条件，以便做填补或删除的操作。同时，由横直向黑run 的夹角所形成的小空白三角亦应被检测以确定是否填补以弥补斜笔划因数字化而造成的锯齿现象。填补与删除操作构成一个处理回合，而整个处理过程则可重覆(iteration) 若干回合，或直到没有填补或删除操作为止。在处理的每一回合中，填补与删除的顺序可以随意为之，并无限制，同时横run与直run的处理顺序也无限制，由于本发明在处理过程中，均是以RUN 为单位，无需检测每一个像素，因此时间上相当迅速，而圆滑的效果亦显而易见。

附图说明:

图1所示为一横向空白run(即RUN)夹在两个黑run之间欲被填补的情形;

图2(a)与2(b)所示为横向空白run部分填补前后的情形;

图3(a)与3(b)所示为横向空白run全部填补前后的情形;

图4示出横向空白run,平底v字凹槽填补;

图5示出小空白三角填补;

图6示出横向黑run 的删除;

图7(a)与7(b)所示为'泛'字在圆滑处理前后的字符图像;

图8(a)与8(b)所示为'柱'字在圆滑处理前后的字符图像;

图9(a)与9(b)所示为'怕'字在圆滑处理前后的字符图像;

图10(a)与10(b)所示为繁体'渊'字在圆滑处理前后的字符图像;

图11(a)与11(b)所示为繁体'态'字在圆滑处理前后的字符图像。

以下为本发明的详细说明。不过，以下的描述只是为了说明本发明的内涵，而非代表本发明的全部内容，当然也不表示本发明只限制在此叙述之内。

本发明涉及数字化的手写汉字图像，假设字符图像为image,其大小为R列XC行，而每一像素点只有黑白两个值。针对此RXC的字符图像，

首先要建run信息表。所谓run是指图像中连续的黑点或白点像素。run可以是直线或横线，也可以是黑 run或空白run。建立了。run信息表之后，接下去处理的每一回合包含了以下三个步骤：(A)空白run填补，(B)小空白三角填补，与(C)黑run 删除。在填补与删除的过程中，同时更新 run信息表，以用于下一回合处理之用。以下就针对填补与删除的步骤做详细说明。

run信息的记录是针对影像中的任一点，假定为(r,c)点，记录其4个方向run的长度，即向上，向下，向左，向右4 个方向的延伸长度(与该点相同的像素长度) 。因此，run 信息表可以表示如下:

vu_run[r][c]:(r,c)的向上的run的长度

vd_run[r][c]:(r,c)的向下的run的长度

hb_run[r][c]:(r,c)的向左的run的长度

hf_run[r][c]:(r,c)的向右的run的长度

其中1＜r＜R,1＜c＜C,表示字符图像所有的点。由于本发明只检测run,由此run信息，可以轻易地算出下一个紧邻run的位置，因此不需检测每一个像素。同时，在填补或删除时，只要更改相关像素点的run信息即可反应出填补或删除后影像run的信息。

(A)空白run 填补

图1所示为一横空白run(标示为RUN)夹在两个黑run(分别标示p_RUN与n_RUN)之间的情形。以下针对此横向空白run说明填补的条件。至于直向空白run的填补与此相似，不再重述。

如图1所示的空白RUN，如果要被填满的话，则必须满足以下的条件:

(1)长度限制: 欲被填补的空白run的长度必须小于一特定值，即

RUN < MAX_RUN_TO_BE_FILLED

其中RUN也表示长度，MAX_RUN_TO_BE_FILLED 则为一事先设定的

值。由于尚有其它条件限制。MAX_RUN_TO_BE_FILLED可以不必定得太小。通常如果图像大小是64×64，则其值可设有为15左右。

(2)欲填补的run 必须为非主干，亦即欲填补的run的长度与前后夹住的黑run 相对来说，应该属次要的。此叙述可以用下列式子来表示：RUN < p_RUN. 或RUN < n_RUN

(3)左，右夹住该空白run的p_RUN与n_RUN必须有连结，其连结可以透过RUN上面或下面的黑run，如图1所示即经过下面的黑run连结。此意表示成条件式子如下：

(i) image [r-1][c-1] = 1且
hf_run [r-1][c-1] > (RUN - 2)

(ii) image [r+1][c-1] = 1且
hf_run [r+1][c-1] > (RUN + 2)

其中image代表字符图像阵列，而像素值等于1代表黑点，0 则代表白点。上述(i),(ii)条件分别表示p_RUN与n_RUN是透过RUN 上面与下面的黑run相连接。

假如如图1所示的白 run(标示为RUN)满足以上三个条件，接着就可以决定是该部分填补或全部填补。不同的填补状况讨论如下：

(1)部分填补：图2(a) 所示为满足部分填补的情形，即白点上、下均为黑点所包住，图2(b)所示为填补后的情形。

(2)全部填补：图3(a) 所示为一简单全部填补的例子，即所有的白点上、下均有黑点包住。另外，对于没有被黑点上下包住的白run，如果其长度小于p_RUN或n_RUN较大长度的一半，则仍然全部填补，亦即

$$RUN < \frac{1}{2} \cdot \max\ (p\_RUN, n\_RUN)$$

(3)平底V字型全部填补：图4所示为平底V字型全部填补的范例。所谓平底V字型，乃指一白run两边被黑run包围在V字型的深谷中，而

此深谷深度通常不是很大，造成原因很可能是掺杂信号所致，故若以下所述条件满足，则可全部填满。图4中，空白run为正检测欲填补的情况，run0则表示其左上k行，向左的黑run，其起点为[r-n，Cn]（此类run即为hb_run[r-n，Cn]，其中n为1到k的整数，而Cn 则为此向左的黑run的起点。C_o=C，Cn可轻易由C及run信息逐次求得。相同的，run1则表示右上角k行，向右的黑run，即hf_run[r-n，Cn]，其中n=1，2，…k；Cn则为此向右黑run的起点，Cn可由Cn-1及run信息轻易求出。C=c+RUN+1。此处的K值为在2至5之间，其值表示V 字的深谷。上述的叙述可以用较数学式的方式叙述如下。

首先，定义C_o，C'_o，run0与run1，

$C_o = C$

$C'_o = C' + RUN - 1$

run0 = run1 =0

sum_run0 = sum_run1 =0

此处run0，run1，sum_run0，sum_run1，实际应写成run0，run1，sum_run0，sum_run1，不过为了方便起见，下标在此省略。

接着，逐一检测欲填补run上面的n行，向左，向右的黑run，n=1，2，…，k。针对第n行，先做如下的运算

$C_n = C_{n-1} - run0 - 1$

$C'_n = C'_{n-1} + run1 + 1$

$$run0 = \begin{cases} 0 & \text{若 image}[r-n, C'n] = 1 \\ hb\_run[r-n, Cn] & \text{否则} \end{cases}$$

$$run1 = \begin{cases} 0 & \text{若 image}[r-n, Cn] = 1 \\ hf\_run[r-n, C'n] & \text{否则} \end{cases}$$

sum_run0 = sum_run0 + run0

sum_run1 = sum_run1 + run1

在此所有变量的下标均省略。执行上述的计算之后就可以检查以下的两个条件:

(i) run0 > V_RUN 且 run1 > V_RUN; 或者

(ii) sum_run0 > SUM_V_RUN 且

sum_run1 > SUM_V_RUN

其中V_RUN与SUM_V_RUN为两个预设常数。

如果以上两个条件之一满足，那么从底部的白RUN上到此第n行的V字型深谷均被填补，否则上述的步骤一直重复至n > k 为止。

以上所述为一横向空白run被填补的情况。如果空白run被填补，则同时更新run信息。对于直向的空白run，填补的处理方式与横向run相似，在此不再赘述。

(B)小空白三角填补

小空白三角填补的操作，如图5 所示，是希望将两个黑run(分别为横直) 所夹住的小三角空白区域填补，以弥补斜线边缘在被数字化时造成的锯齿状。

所谓小空白三角是由一个横向，一个直向的黑run 所构成的直角三角形空白区域。如图5的小空白三角△HOV为例，其要被填补的条件如下(假设△点的位置为(r,c)):

(1) hf_run[r-1][c-1] < hf_run[r][c] 且

vd_run[r-1][c-1] < vd_run[r][c] ;

(2) H_RUN < TRIANGLE_SIZE 或

V_RUN < TRIANGLE_SIZE

其中 H_RUN = min(hf_run[r-1][c-1], TRIANGLE_SIZE)

V_RUN = min(vd_run[r-1][c-1], TRIANGLE_SIZE)

(3) OH′ 与OV′ 的夹角大于135°，其中H′ 与V′ 为O点沿H与V路径上m点之外的像素点，其中m为一常数。此条件是保证H,O,V三点是在同一斜笔划线上。

以上所述的小空白三角填补条件若满足，则△OHV 中所有的白点均被填补成黑点。

以上所述以仅位于第四象限的小空白三角为例，至于小空白三角第一、二、三象限的情形，填补条件相似。

(C) 黑run删除

考虑如图6所示的横向黑run(标示为RUN)，假设其左边起始位置为(r,c),而u_RUN与d_RUN分别为其上、下的黑run的长度。u_RUN与d_RUN的定义如下:

$$u\_RUN = \begin{cases} 0 & \text{若 image [r-1][C] =0 或image [r-1][C+RUN-1] =0} \\ hf\_run[r-1][C]-hb\_run[r-1][C]-1 & \text{否则} \end{cases}$$

$$d\_RUN = \begin{cases} 0 & \text{若 image [r+1][C] =0 或image [r+1][C+RUN-1] =0} \\ hf\_run[r+1][C]-hb\_run[r+1][C]-1 & \text{否则} \end{cases}$$

其中的黑RUN,必须连RUN以下的短黑run一起考虑下述的条件:

(1) RUN < MAX_RUN_TO_BE_DELETED

其中MAX_RUN_TO_BE_DELETED为一预设常数，限制被删除黑run的长度。

(2)(i) u_RUN≥ 2 * run 且u_RUN > LEN_TO_DELETE_NGHB

或(ii) d_RUN≥ 2 * run 且d_RUN > LEN_TO_DELETE_NGHB

(3)假设RUN之下的短黑run，两端点均在run的两端之内，且其行数为w,则w必须小于一预设常数w。一般来说w值应在5以下。

如果上述三个条件均满足，则所有w 个短黑run连同RUN本身一起

删除。直向黑run的删除与横向黑run删除相似。

以上所述即为空白run填补，小空白三角填补与黑run删除的条件。此填补、删除操作构成圆滑处理的一个处理回合。整个圆滑处理过程可包含一致数个填补与删除回合，或者直到没有填补或删除操作发生之后。

说 明 书 附 图 CFEL/045412

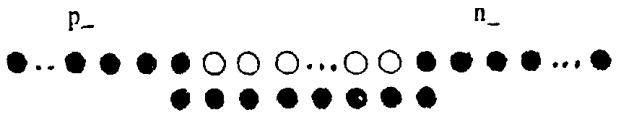


图 1

图 2(a)

图 2(b)

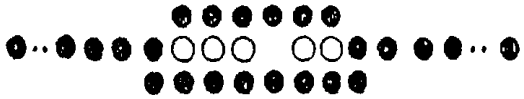


图 3(a)

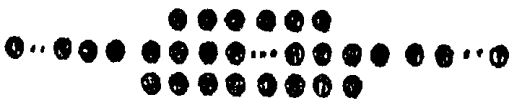


图 3(b)

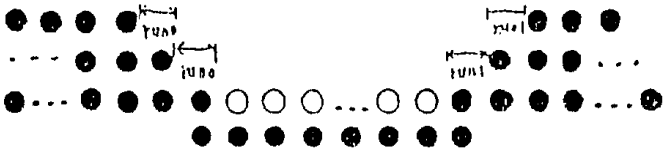


图 4

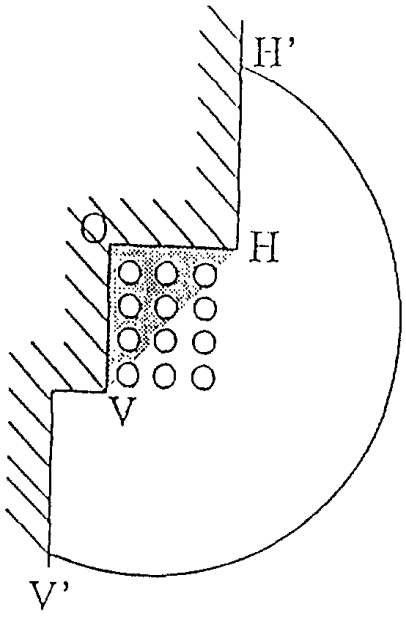


图 5

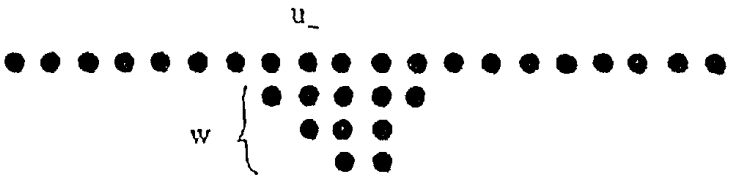


图 6

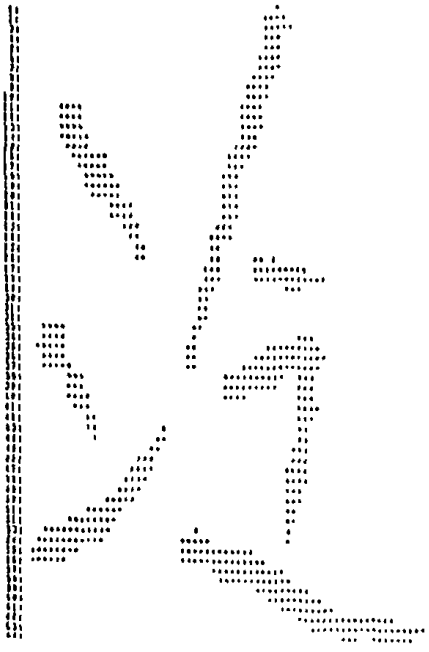


图 7(a)

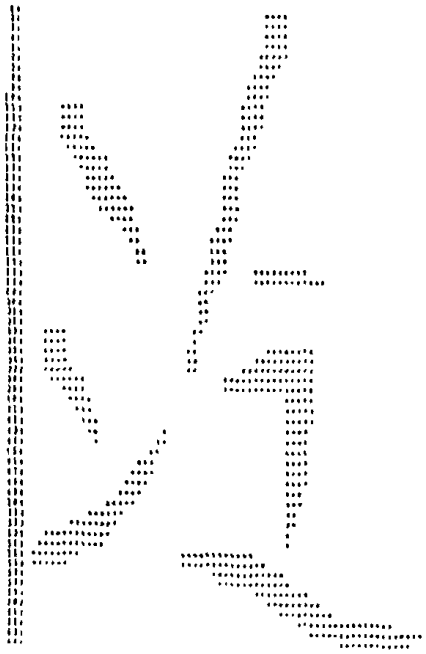


图 7(b)

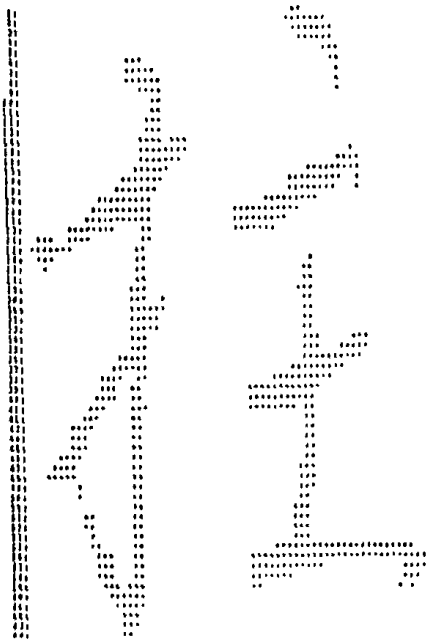


图 8(a)

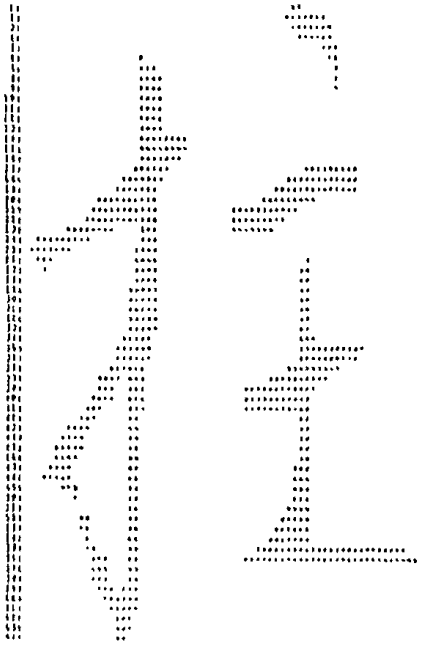


图 8(b)

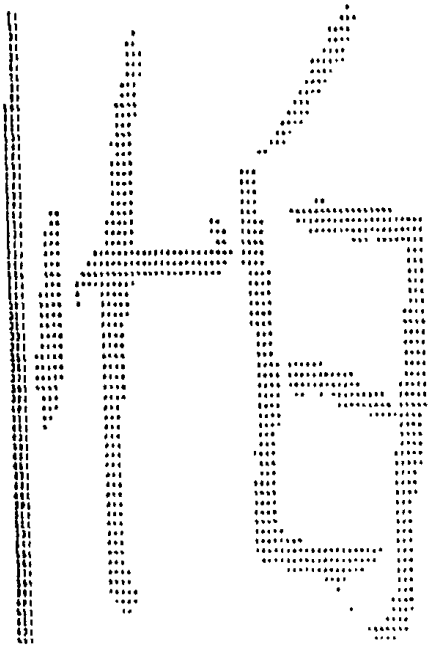


图 9(a)

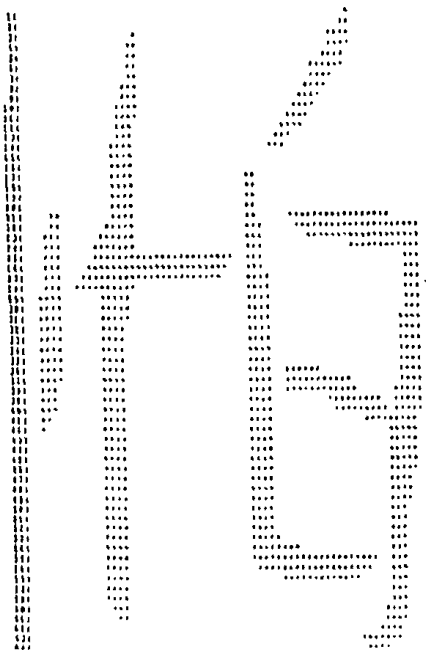


图 9(b)

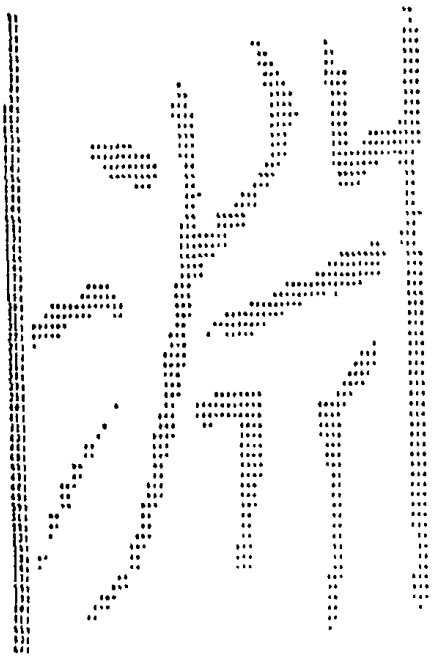


图 10(a)

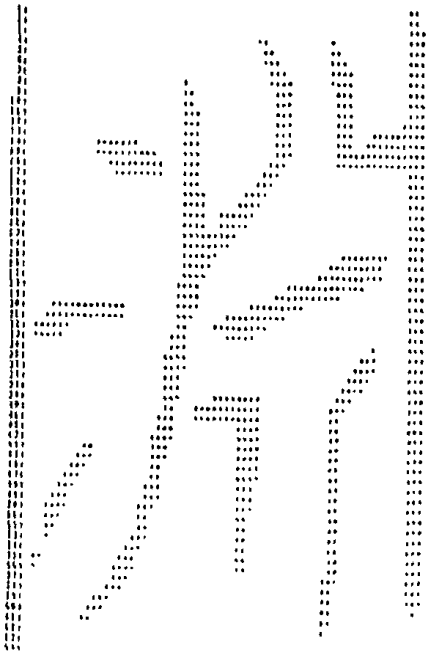


图 10(b)

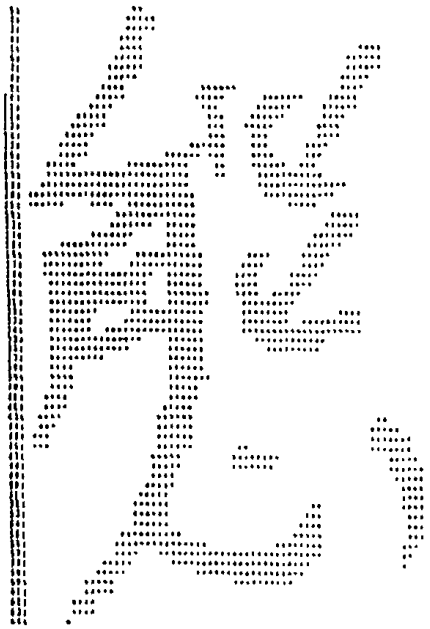


图 11(a)

图 11(b)