第 2 卷 第 12 期
1997 年 12 月

中国图象图形学报
Journal of Image and Graphics

Vol. 2, No. 12
Dec. 1997

二维模式近似匹配的快速算法

桑梓勤　丁明跃　张天序

(华中理工大学图象识别与人工智能研究所 图象信息处理与智能控制国家教委开放实验室,武汉　430074)

摘　要　给定一个大小为 $n \times n$ 的文本 $\mathscr{T}$ 和一个大小为 $m \times m$ 的模板 $\mathscr{P}$,如果文本 $\mathscr{T}$ 中存在一个 $m \times m$ 的子块与模板 $\mathscr{P}$ 能够逐点匹配,称为精确匹配。如果最多有 k 个元素不同,称为带有最多 k 个误差的近似匹配。对于精确匹配,本文给出了一个时间复杂性为 $O(n^2\log|\sum|)$ 的算法,$\sum = \{a_1, a_2, \cdots, a_{|\sum|}\}$,是模板的字符集。对于近似匹配,快速算法分为两步:(1) 预选。利用精确匹配算法找出能精确匹配的 $s \times s (0 \leqslant s \leqslant m)$ 子块,得到 h 个候选的对准点;(2) 验证。把模板对准候选点,逐点比较,以确定不相同的元素是否不超过 k 个。近似匹配的时间复杂性为 $O(n^2\log|\sum| + hm^2)$。

关键词　精确匹配,近似匹配,快速算法

1　引　言

所谓二维模式匹配,就是在一个 $n \times m$ 二维矩阵 $\mathscr{T}$(称为文本)中寻找另一个 $p \times q$ 二维矩阵 $\mathscr{P}$(称为模板)。为方便起见,假定文本 $\mathscr{T}$ 的大小为 $n \times n$,模板 $\mathscr{P}$ 的大小为 $m \times m, m < n$,如果在文本 $\mathscr{T}$ 中我们能够找到某一个起始点 $(i_0, j_0), 1 \leqslant i_0, j_0 \leqslant n - m + 1m$,使得 $t[i_0 + i][j_0 + j] = p[i + 1][j + 1], 0 \leqslant i, j \leqslant m - 1$,则我们称模板能够精确地匹配文本。如果我们允许模板和文本之间最多有 k 个元素不同,则称模板和文本是带有最多 k 个误差的近似匹配。

在模式匹配中,近年来对于一维的字符串匹配研究较多。从最早的 Boyer-Moore 算法[1] 和 Knuth-Morris-Pratt 算法[2],到 Karp-Rabin 的随机算法[3] 以及 Vishkin 的决定性抽样[4],都是精确的字符串匹配,时间复杂性为 $O(n)$,n 为模式串的长度。关于字符串匹配的有限自动机模型,可参看[5]。对于带 $k(k > 0)$ 个字符误差的字符串匹配,如果字符集的大小固定,Landau-Vishkin 算法的时间性能是 $O(kn)$[6]。Grossi 和 Luccio[7]、Baeza-Yates 和 Gonnet[8]、Wu 和 Manber[9] 都提出了过滤(filtration)算法。该算法不仅在最坏情况下有很好的性能,而且实际运行起来也非常快。Ukkonen 则提出了另外一类过滤算法[10],Pevzner 和 Waterman 采用多重过滤(multiple filtration),使实际运行速度更快。一维的字符串匹配可以用在文献检索、文本编辑、分子生物学中核苷酸与氨基酸的匹配等方面[11]。

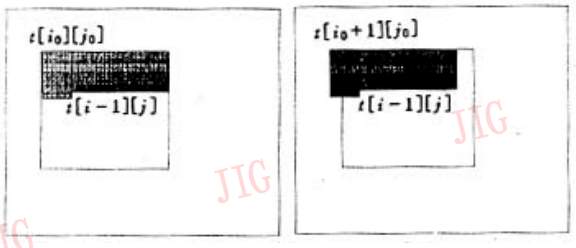
对于二维的模式匹配,Baker 最早把一维的精确匹配算法推广到二维[12],其时间复杂性是 $O(n^2)$,所需的空间为 $O(n^2)$。可以看出,Baker 的方法需要与原文本同样大小的存贮空间。Karp 和 Rabin 采用随机方法求出文本和模式的"指纹函数",如果文本和模式的"指纹"相符,则再进行精确匹配,时间复杂性是 $O(n^2)$,所需的空间为 $O(1)$[3]。但无论是 Baker 算法还是 Karp-Rabin 算法,都无法直接推广到带有 k 个误差的二维模式近似匹配。彭嘉雄和张天序通过对文本元素在空间的自相关性进行统计分析,提出了自适应搜索法,使得模板能够变步长移动,加快了匹配速度。但是在最坏情况下,所需的时间仍然是 $O(n^2m^2)$[13]。

收稿日期:1996-11-17;收到修改稿日期:1997-03-10

本文在第 2 节提出了一个时间为 $O(n^2\log|\sum|)$* 的二维模式的精确匹配算法，第 3 节对近似匹配进行了分析，导出了一个时间复杂性相近的算法，最后给出了我们的结论。

2 二维模式的精确匹配

最直观的方法，是把模板 $\mathscr{P}$ 与文本 $\mathscr{T}$ 在文本左上角 $t[1][1]$ 处对准(称为对准点)，进行逐个元素的比较，如果 $m\times m$ 个元素都相同，则得到了精确匹配点(1,1)。如果不能匹配，则把对准点移到 $t[2][1]$ 进行匹配，这样逐行逐个元素扫描移动模板，最坏情况下要穷举所有的对准点，进行 $(n-m+1)^2\times m^2$ 次比较。如图 1 所示，这个方法的问题在于没有考虑模板元素之间的相关性。事实上，当前一次的对准点为 $t[i_0][j_0]$ 时，假定匹配到时 $t[i][j]$ 时，第一次出现 $t[i][j]\neq p(i-i_0+1)[j-j_0+1]$，则要把对准点移到 $t[i_0+1][j_0]$，重新开始比较，而前一次所进行的 $(j-j_0)\times m+(i-i_0)$ 成功的比较，对于后一次对准，没有丝毫帮助。从图 1(b)可知，前一次 $(j-j_0)\times(m-1)+(i-i_0)$ 次成功的比较，落在后一次对准的模板中。我们下面的方法主要是发挥这些元素的作用，使得 $t[i][j]$ 只与模板中的某个元素比较一次，以减少总的匹配次数，提高匹配算法的性能。



(a) (b)

图 1 最直观的穷举搜索法

(a) 对准点为，阴影部分表示上层模板与下层文本的元素完全相同；(b) 对准点为

Fig. 1 The brute-force search

(a)the alignig point is $t[i_0+1][j_0]$, the shadow means that the elements of up pattern are of the same of down pattern (b) the alignig point is $t[i_0][j_0]$

我们将 Wu-Manber 的一维字符串匹配方法[9]在二维上进行推广。取 n 个 $m\times m$ 的布尔矩阵 $R_1,R_2,\cdots,R_n$，分别用于记录第 v 行文本元素 $t[1][v],t[2][v],\cdots,t[n][v]$ 与模板 $\mathscr{P}$ 中某个元素 $p[i][j]$ 的匹配情况。为了反映出扫描行的变化，我们把布尔矩阵记作 $R_1^{(v)},R_2^{(v)},\cdots,R_n^{(v)},1\leqslant v\leqslant n$，扫描行由第 v 行移到第 $v+1$ 行时 $R_1^{(v)},R_2^{(v)},\cdots,R_n^{(v)}$，更新为 $R_1^{(v+1)},R_2^{(v+1)},\cdots,R_n^{(v+1)}$，。布尔矩阵 $R_u^{(v)}$ 的元素取值为：

$$r_u^{(v)}[i][j]=\begin{cases}1 & \text{当 } t[u][v]=p[i][j] \text{ 且} \\ & r_u^{(v-1)}[i][j-1]=1 \text{ 且} \\ & r_{u-1}^{(v)}[i-1][j]=1 \text{ 时} \\ 0 & \text{否则}\end{cases} \tag{1}$$

$r_u^{(v)}[i][j]=1$ 表示文本 $\mathscr{T}$ 上以 $t[u-i+1][v-j+1]$ 为左上角、以 $t[u][v]$ 为右下角的一个 $i\times j$ 小块与模板 $\mathscr{P}$ 的一个 $i\times j$ 小块相匹配。如果 $i=j=m$，则表示存在一个精确匹配。从递归表达式(1)中可以看出，文本元素 $t[u][v]$ 与模板元素 $p[i][j]$ 比较时，$r_u^{(v-1)}[i][j-1]=1$ 表示上一行扫描时文本 $\mathscr{T}$ 的一个 $i\times(j-1)$ 小块与模板 $\mathscr{P}$ 相匹配，$r_{u-1}^{(v)}[i-1][j]=1$ 又保证了左边一个 $(i-1)\times j$ 小块与模板 $\mathscr{P}$ 是匹配的。为了使递归表达式(1)对于 $i=j=u=v=1$ 也适用，我们令 $r_u^{(0)}[i][0]=1,1\leqslant i\leqslant m$ 和 $r_0^{(v)}[0][j]=1,1\leqslant j\leqslant m$。这相当于在初始化时把记录矩阵 R_u 暂时由 $m\times m$ 扩大为 $m\times(m+1)$，及增加一个 $(m+1)\times m$ 的记录矩阵 R_0，如图 2 所示。

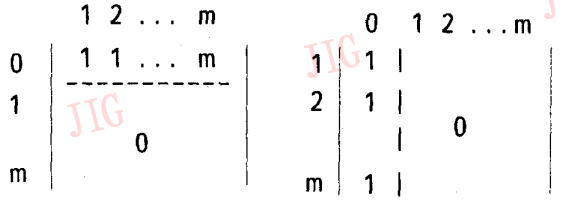


图 2 (a)R_u 由 $m\times m$ 扩大为 $m\times(m+1)$；(b)增加一个 $(m+1)\times m$ 的 R_0

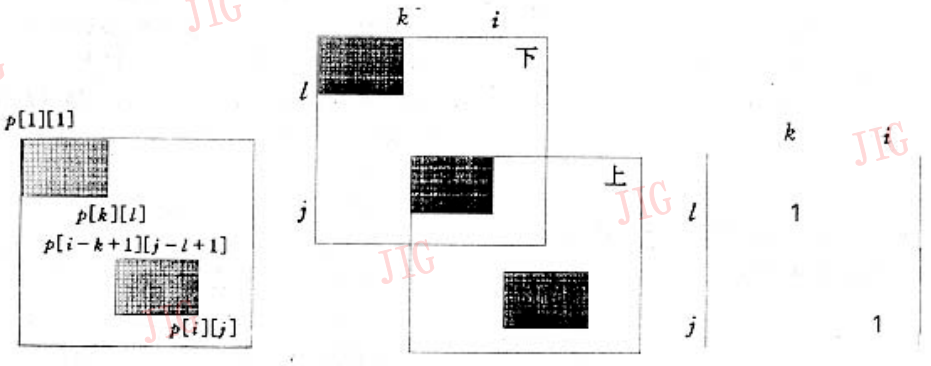
Fig. 2 (a) $R_u$expands from $m\times m$ to $m\times(m+1)$; (b) Add aR_0 of size $(m+1)\times m$

递归表达式(1)也可以理解为布尔矩阵 $R_u^{(v-1)}$ 的“1”由 $r_u^{(v-1)}[i][j-1]$ 下移到 $R_u^{(v)}$ 的，$r_u^{(v)}[i][j]$，且 $R_u^{(v-1)}$ 的“1”由 $r_{u-1}^{(v)}[i-1][j]$ 右移到的 $R_u^{(v)}$ 的 $r_u^{(v)}[i][j]$，且保证 $t[u][v]=p[i][j]$。我们的算法正是基于这种理解。下面首先给出一个引理。

* 以下对数都是以 2 为底的，即 $\log x=\log_2 x$。

引理 2.1　在模板 $\mathscr{P}$ 中，如果从 $p[1][1]$到 $p[k][l]$的一个 $k\times l$ 小块，重复出现在从 $p[i-k+1][j-l+1]$到 $p[i][j]$的 $k\times l$ 小块中，则在文本匹配时，对于记录矩阵 $R_u^{(v)}$，若 $r_u^{(v)}[i][j]=1$，一定有 $r_u^{(v)}[k][l]=1$。

证明：如图 3(a)所示，当模板中存在 $k\times l$ 的重复小块时，一定有 $p[1][1]=p[i-k+1][j-l+1]$，而布尔矩阵 $R_u^{(v)}$ 中，$r_u^{(v)}[i][j]=1$ 蕴含着 $t[u-k+1][v-l+1]=p[i-k+1][j-l+1]=p[1][1]$，根据表达式(1)，$r_{u-k+1}^{(v-l+1)}[i-k+1][j-l+1]=r_{u-k+1}^{(v-l+1)}[1][1]=1$，如图 3(b)，相当于下模板的 $p[i-k+1][j-l+1]$元素与文本匹配时，又叠加了一个上模板。则当上下模板同时右移 k 步、下移 1 步时，$r_u^{(v)}[i][j]=1$ 一定有 $r_u^{(v)}[k][l]=1$。



(a)　　(b)　　(c)

图 3 (a)模板 $\mathscr{P}$ 中存在 $k\times l$ 的重复小块；(b)模板 $\mathscr{P}$ 可以与自身对准，上下模板存在 $k\times l$ 的重复小块；(c)布尔矩阵 $R_u^{(v)}$ 中，$r_u^{(v)}[i][j]=r_u^{(v)}[k][l]=1$.

Fig. 3　(a) There is a duplicate subblock of size $k\times l$ in the patten $\mathscr{P}$; (b) the pattern $\mathscr{P}$ can self-align, there are duplicate subblocks in the up and down patterns; (c) in $R_u^{(v)}$, $r_u^{(v)}[i][j]=r_u^{(v)}[k][l]=1$

引理 2.1 解决了图 1 所示的最直观方法的“元素重复比较”问题，使得文本 $\mathscr{T}$ 中的每个元素只与模板比较一次。为了操作方便，我们把模板也变为布尔矩阵。令模板的字符集 $\sum=\{a_1,a_2,\cdots,a_{|\sum|}\}$，$|\sum|\leqslant m^2$，则对于字符集 $\sum$ 中的每个字符 a_k，$1\leqslant k\leqslant|\sum|$，我们构造一个与模板同样大小的布尔“掩膜”$M_k$，它表示该字符在模板中的位置，即对于字符 a_k，它的掩膜

$$m_k[i][j]=\begin{cases}1 & \text{当 } p[i][j]=a_k \text{ 时}\\ 0 & \text{否则}\end{cases}\tag{2}$$

我们根据文本元素 $t[u][v]$在字符集中找到 a_k（假定字符集已排序，使用折半查找，时间为 $O(\log|\sum|)$），就可以确定掩膜 M_k。则表达式(1)使用矩阵可以表述为：

$$R_u^{(v)}=M_k \quad \text{AND} \quad (R_u^{(v-1)}\text{ 下移一行}) \quad \text{AND} \quad (R_{(u-1)}^{(v)}\text{ 右移一列})\tag{3}$$

由于记录距阵和掩膜的元素取值为 0 或 1，因此可用一个机器“位”表示。如果机器一个字长为 w 位，当以行（或列）为主存储时，每个 $m\times m$ 矩阵需要 $m\times[m/w]$个“字”表示。以下我们以行为主存贮记录矩阵和掩膜，矩阵的 AND 操作可以用汇编语言实现，(3)在常数时间内完成。根据表达式(1)，$R_u^{(v-1)}$ 下移一行时，第一行要补充 m 个二进制“1”，$R_{u-1}^{(v)}$右移一列时，第一列要完成移位(SHIFT)指令，最左边位为二进制“1”，即：

$$\begin{bmatrix}r_u^{(v)}(1)\\ r_u^{(v)}(2)\\ \vdots\\ r_u^{(v)}(m)\end{bmatrix}=\begin{bmatrix}m_k(1)\\ m_k(2)\\ \vdots\\ m_k(m)\end{bmatrix}\text{AND}\begin{bmatrix}11\cdots1\\ r_u^{(v-1)}(1)\\ \vdots\\ r_u^{(v-1)}(m-1)\end{bmatrix}\text{AND}\begin{bmatrix}SHIFT(r_{u-1}^{(v)}(1))\\ SHIFT(r_{u-1}^{(v)}(2))\\ \vdots\\ SHIFT(r_{u-1}^{(v)}(m))\end{bmatrix}\tag{4}$$

利用汇编语言实现时，(4)需要的时间为 $O(1)$。下面是精确匹配的具体算法：

PROCEDURE exact-matching;
BEGIN
{初始化掩膜和记录矩阵}
for i:=1 to m do
for j:=1 to m do
BEGIN
找出模板 $\mathscr{P}$ 的字符 a_k，排序；
构造掩膜 M_k；
END;
初始化记录矩阵 $R_0,R_1,R_2,\cdots,R_n$;
{文本元素与模板元素匹配}
for v:=1 to n do
for u:=1 to n do
BEGIN
读取文本元素 $r[u][v]$;
利用折半查找，找出模板 $\mathscr{P}$ 的字符 a_k;
if $t[u][v]=a_k$ then
BEGIN

$$\begin{bmatrix} r_u^{(v)}(1) \\ r_u^{(v)}(2) \\ \vdots \\ r_u^{(v)}(m) \end{bmatrix} = \begin{bmatrix} m_k(1) \\ m_k(2) \\ \vdots \\ m_k(m) \end{bmatrix} \text{AND} \begin{bmatrix} 11\cdots1 \\ r_u^{(v-1)}(1) \\ \vdots \\ r_u^{(v-1)}(m-1) \end{bmatrix} \text{AND}$$

$$\begin{bmatrix} SHIFT(r_{u-1}^{(v)}(1)) \\ SHIFT(r_{u-1}^{(v)}(2)) \\ \vdots \\ SHIFT(r_{u-1}^{(v)}(m)) \end{bmatrix} \text{END}$$

$$\text{else} \begin{bmatrix} r_u^{(v)}(1) \\ r_u^{(v)}(2) \\ \vdots \\ r_u^{(v)}(m) \end{bmatrix} = [0]$$

$$\text{if} \begin{bmatrix} r_u^{(v)}(1) \\ r_u^{(v)}(2) \\ \vdots \\ r_u^{(v)}(m) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \text{then 输出}(u-m+1,v-m+1);$$

END
END;

定理 2.2 精确匹配的匹配时间为 $O(n^2\log|\sum|)$，空间为 $O(m\lceil m/w\rceil|\sum| + nm\lceil m/w\rceil)$。

证明：根据算法可知，匹配的时间之所以不是 n^2，是因为在算法的循环体中，需要折半查找模板 $\mathscr{P}$ 的字符 a_k，因此要乘以因子 $\log|\sum|$，为 $n^2\log|\sum|$。而在初始化时，需要 $m\lceil m/w\rceil|\sum|$ 个存贮空间保存 $|\Sigma|$ 个字符的掩膜，而 $n+1$ 个记录矩阵需要 $(n+1)m\lceil m/w\rceil$ 个字，因此所需空间为 $m\lceil m/w\rceil|\sum| + (n+1)m\lceil m/w\rceil$。

3 近似匹配

根据前面的定义，如果文本 $\mathscr{T}$ 中有一个 $m\times m$ 的块与模板 $\mathscr{P}$ 匹配，它们之间不同的元素个数不超过 k，$k<m^2$。那么这个块与模板 $\mathscr{P}$ 至少有一个 $s\times s$ 子块能够精确匹配，$1\leqslant s\leqslant m$。我们的近似匹配算法就是基于这种观察。首先在文本 $\mathscr{T}$ 上找到一个 $s\times s$ 的块与模板 $\mathscr{P}$ 精确匹配，我们称为预选阶段，然后把这个 $s\times s$ 块向上下左右扩大为 $m\times m$ 块，与模板 $\mathscr{P}$ 再次匹配，如果不同元素不超过 k，则这个 $m\times m$ 块即为所求，这个阶段我们称为验证阶段。

现在的问题在于：1）$s\times s$ 块多大？2）$s\times s$ 块相对于 $m\times m$ 的模板，定位在哪里？

仿照 Pevzner 和 Waterman 处理一维字符串近似匹配的方法[11]，对于二维的模式近似匹配，我们得到了以下引理：

引理 3.1 在一个 $m\times m$ 的布尔矩阵 $\boldsymbol{B}$ 中，如果有最多 k 个“0”，则 $\boldsymbol{B}$ 至少包含 $(m-s+1)^2-k\times s^2$ 个 $s\times s$ 的全“1”子布尔矩阵 $\boldsymbol{S}$。

证明：布尔矩阵 $\boldsymbol{B}$ 包含 $(m-s+1)^2$ 个大小为 $s\times s$ 的子布尔矩阵 $\boldsymbol{S}$，$\boldsymbol{B}$ 中的每个“0”最多属于 s^2 个子布尔矩阵，这样 $\boldsymbol{B}$ 中的所有“0”（最多 k 个），最多属于 $k\times s^2$ 个子布尔矩阵。则全“1”的子布尔矩阵至少有 $(m-s+1)^2-k\times s^2$ 个。

例子 如图 4 所示，$m=12$ 且 $k=8$ 时，选取 $s=3$，则 $\boldsymbol{B}$ 包含 $(12-3+1)^2=100$ 个子布尔矩阵 $\boldsymbol{S}$，其中至少有 $100-8\times3^2=28$ 个全“1”的 3×3 子布尔矩阵 $\boldsymbol{S}$。

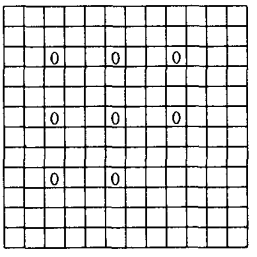


图 4 $m=12,k=8$ 时，“0”接近均匀地分布在 $\boldsymbol{B}$ 中（非“0”的格为“1”，图中没有标出）。

Fig. 4 $m=12,k=8$, “0” distribute uniformly in $\boldsymbol{B}$ (*the mon “0” lattice is “1”*)

引理 3.2 在一个 $m\times m$ 的布尔矩阵 $\boldsymbol{B}$ 中，如果有最多 k 个“0”，且至少有一个全“1”的子布尔矩阵 $\boldsymbol{S}$，则它的边长为 $s\leqslant\lceil m/(k_2+1)\rceil$，$k_2=\lceil\sqrt{k}\rceil$。

证明：由于 $\boldsymbol{B}$ 中至少有一个全“1”的子布尔矩

阵 S，根据引理 3.1，令 $(m-s+1)^2-k\times s^2\geqslant 1$。即 $k\times s^2\leqslant(m-s+1)^2-1$，则 $k\times s^2<(m-s+1)^2$，即 $\sqrt{k}\cdot s<m-s+1$。为了求得 s 的最小上界，取 $k_2=[\sqrt{k}]$，则 $k_2\cdot s<m-s+1$，即 $(k_2+1)\cdot s<m+1$，也就是 $(k_2+1)\cdot s\leqslant m$，得到 $s\leqslant[m/(k_2+1)]$。

我们注意到文本 $\mathscr{T}$ 的一个 $m\times m$ 块 T' 与模板 $\mathscr{P}$ 进行最多 k 个误差的匹配时，按照下列规则可以得到一个 $m\times m$ 的布尔矩阵 $\boldsymbol{B}$：

$$b[i][j]=\begin{cases}1 & \text{当 } t'[i][j]=p[i][j] \text{ 时}\\ 0 & \text{否则}\end{cases}\quad(5)$$

这样，由引理 3.1 可得到引理 3.3，由引理 3.2 可得到引理 3.4。

引理 3.3　如果文本 $\mathscr{T}$ 的一个 $m\times m$ 块 T' 与模板 $\mathscr{P}$ 能够进行最多 k 个误差的近似匹配，则当 $s\leqslant[m/(k_2+1)]$ 时，T' 和 $\mathscr{P}$ 共有 $(m-s+1)^2-k\times s^2$ 个 $s\times s$ 子块能精确匹配。

证明：略。

引理 3.4　如果文本 $\mathscr{T}$ 的一个 $m\times m$ 块 T' 与模板 $\mathscr{P}$ 能够进行最多 k 个误差的近似匹配，则它们精确匹配的 $s\times s$ 子块边长可取 $[m/(k_2+1)]$。

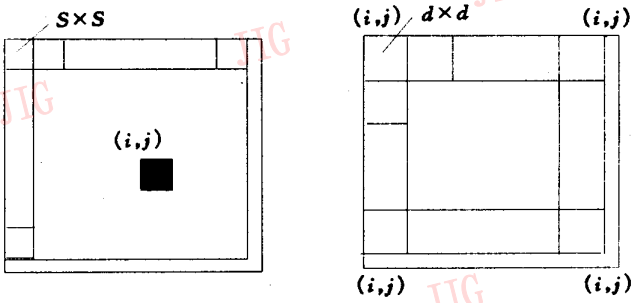
证明：略。

引理 3.5　令 $d=k_2+1$，$s=[m/d]$。如果文本 $\mathscr{T}$ 的一个 $m\times m$ 块 T' 与模板 $\mathscr{P}$ 能够进行最多 k 个误差的近似匹配，把 $m\times m$ 的模板 $\mathscr{P}$ 分割成 d^2 个 $s\times s$ 的子块，则至少有一个 $s\times s$ 的子块与模板 $\mathscr{P}$ 精确匹配。

证明：利用反证法。假设对于 d^2 个 $s\times s$ 的子块，每个子块都至少有一个元素不匹配，则对于 $m\times m$ 的模板 $\mathscr{P}$ 来说，不匹配的元素不小于 d^2，而 $d^2=(k_2+1)^2>k$，矛盾！命题得证。

引理 3.4 回答了我们的第一个问题。引理 3.3 和引理 3.5 保证了 $s\times s$ 子块存在。特别是引理 3.5，使我们得出了以下寻找候选的 $s\times s$ 子块的方法。

我们把第 2 节的精确匹配算法进行下列修改：首先把模板 $\mathscr{P}$ 分割成 d^2 个 $s\times s$ 子块（会留下 $m-d\times s$ 行和 $m-d\times s$ 列的“边角料”），然后取出每个 $s\times s$ 子块的 (1,1) 元素，组合成一个 $d\times d$ 子块，同样 d^2 个 $s\times s$ 子块的 (2,1) 元素，组合成第二个 $d\times d$ 子块，依次类推，得到 s^2 个 $d\times d$ 的子块，与留下的“边角料”再组成一个 $m\times m$ 的新模板 $\mathscr{P}'$。假如模板 $\mathscr{P}$ 的第 (i,j) 个子块 $s\times s$ 精确匹配，$1\leqslant i,j\leqslant d$，则对于交错模板 $\mathscr{P}'$，相匹配的 s^2 个元素分别分布在 s^2 个 $d\times d$ 子块 (i,j) 位置，如图 5 所示。因此，文本 $\mathscr{T}$ 与交错模板 $\mathscr{P}'$ 匹配时，表达式 (3) 中记录矩阵 $R_u^{(v-1)}$、$R_{u-1}^{(v)}$ 不是分别下移一行和右移一列，而是下移 d 行和右移 d 列，匹配的标志是 $r_u^{(v)}[(s-1)\times d+i][(s-1)\times d+j]=1$。



(a)　　(b)

图 5　(a) 模板 $\mathscr{P}$ 分割成个子块，第 (i,j) 个子块能精确匹配；(b) 交错模板 $\mathscr{P}'$ 由个子块组成，每个子块的 (i,j) 元素都能精确匹配.

Fig. 5　(a) The pattern $\mathscr{P}'$ is cut to d^2 subblocks of size $s\times s$, the (i,j) Subblock can exact-match; (b) The across pattern $\mathscr{P}'$ consists of s^2 subblocks of size $d\times d$, the (i,j) element of every subblock can exact-match.

于是，寻找子块 $s\times s$ 的精确匹配算法为：

```
PROCEDURE exact-matching2;
BEGIN
    {初始化交错模板的掩膜及记录矩阵}
    k2=[√k];
    d=k2+1;S=[m/d];
    构造交错模板 P';
    for i:=1 to m do
    for j:=1 to m do
      BEGIN
        找出交错模板 P' 的字符 ak，排序;
        构造交错模板 P' 的掩膜 Mk;
      END
    初始化记录矩阵 R0,R1,R2,…,Rn;
    {文本元素和交错模板元素匹配}
    for v:=1 to n do
    for u:=1 to n do
      BEGIN
        读取文本元素 t[u][v];
    利用折半查找，找出交错模板 P' 的字符 ak;
    if t[u][v]=ak then
      BEGIN
        R_u^(v)=Mk AND (R_u^(v-1) 下移 d 行) AND (R_(u-1)^(v)
            右移 d 列)
```

```
        END
      else R_u^(v) ≔[0];
      if r_u^(v)[(s-1)×d+i][(s-1)×d+j]=1 then
      输出对准点(u-i×s+1,v-j×s+1);
      END
  END;
```

近似匹配的验证阶段，可以采用最简单的的方法，把模板 $\mathscr{P}$ 与预选出来的文本对准点对准，逐点比较。

```
PROCEDURE approximate-matching;
  BEGIN
    exact-matching2;
    文本 𝒯 从对准点(u-i×s+1,v-j×s+1)开始
    与模板 𝒫 逐点比较;
    if 误差<k then 输出对准点(u-i×s+1;v-j×s
+1);
  END;
```

定理 3.6 如果候选的文本对准点有 h 个，则近似匹配时间为 $O(n^2\log|\sum| + hm^2)$，空间为 $O(m\lceil m/w\rceil|\sum| + nm\lceil m/w\rceil)$。

证明：从上述近似匹配算法中可知，总的时间包括预选时间和验证时间。算法 exact-matching2 和 exact-matching 的时间复杂性相同，即预选时间为 $O(n^2\log|\sum|)$。而验证时间为 $h\cdot m^2$，则总的时间复杂性为 $O(n^2\log|\sum| + hm^2)$。空间复杂性不变，仍然是 $O(m\lceil m/w\rceil|\sum| + nm\lceil m/w\rceil)$。

4 讨 论

由定理 2.2 和定理 3.6 可知，精确匹配的时间与文本的大小和模板的字符集的大小有关，与模板的大小无关，而近似匹配的时间还与预选出来的文本对准点个数有关。考虑近似匹配的一个应用——目标检测，二维图象（文本和模板）的字符即像元的取值，对于二值图象，$|\sum| \leqslant 2$，对于 0～255 级灰度的灰度图，$|\sum| \leqslant 256$，而彩色图象模板的字符集的大小为所取色彩的个数。

图象在采样、量化和传输时总会带来一些随机的噪声点，我们把噪声点控制在一定的范围内（$\leqslant k$），可以采用本文提出的近似匹配方法进行目标检测。特别是，当目标很稀少时，预选的文本对准点也很稀疏，近似匹配时间非常接近于精确匹配时间。而且，我们预先对模板进行了预处理，可以用于在线检测。

我们下一步的工作是另外一个类似的问题：如果近似匹配不仅仅是指文本与模板有 k 个元素不同，而且它们对应的元素之间的差距（如灰度差）控制在一定的范围之内，即：

$$\sum_{i=0}^{m-1}\sum_{j=0}^{m-1}|t[i_0+i][j_0+j]-p[i+1][j+1]|<\varepsilon,$$
$$1\leqslant i_0,j_0\leqslant n-m+1$$

怎样找出快速算法？

参 考 文 献

1 Boyer R S, Moore J S. A fast string searching algorithm, Comm. ACM, 1977, 20: 762～722.

2 Knuth D E, Morris J H, Pratt VR. Fast pattern matching in strings, SIAM J. Comput., 1977, 6: 323～350.

3 Karp R M, Rabin M D. Efficient randomized pattern-matching algorithm, IBM J. Res. Develop., 1987. 31: 249～260.

4 Vishkin U. Deterministic sampling - a new technique for fast pattern matching, SIAM J. Comput., 1991, 20: 22～40.

5 Aho A V, Hopcroft J E, Ullman J D. The Design and analysis of computer algorithms, Addison-Wesley, 1976.

6 Landau G M, Vishkin U. Fast parallel and serial appoximate string matching, J. Algorithms, 1989, 10: 157～169.

7 Galil Z, Luccio F. Simple and efficient string matching with k mismatches, Inform. Process. Lett. 1990, 33: 113～120.

8 Baeza-Yates R, Gonnet G H. A new approach to text searching, Comm. ACM, 1992, 35(10): 74～82.

9 Wu S, Manber U. Fast text searching allowing errors, Comm. ACM, 1992, 35(10): 83～90.

10 Ukkonen U. Finding approximate string-matching with q-grams and maximal matching, Theoret. Comput. Sci., 1992, 92: 191～211.

11 Pevzner P A, Waterman M S. Multiple filtration and approximate pattern matching, Algorithmica, 1995, 13: 135～154.

12 Baker T P. A technique for extending rapid exact-match string matching to arrays of more than one dimension, SIAM J. Comput., 1978, 7: 533～1541.

13 彭嘉雄，张天序. 模板匹配的自适应搜索法，自动化学报，1989，15(3)：201～208.

桑梓勤,1969 年 6 月出生,现为华中理工大学图象所博士研究生,主要研究方向为算法与计算复杂性、光照模型与图象合成。

A Fast Approach to Two-Dimensional Pattern Matching

Sang Ziqin, Ding Mingyue , Zhang Tianxu

(*Institute of Pattern Recognition and Artificial Intelligence, Huazhong University of Science and Technology, Wuhan* 430074)

Abstract Given a text of size $n\times n$ and a pattern of size $m\times m$, the exact-matching is to find all occurrence of the pattern in the text, while the approximate-matching is to find all the location of $m\times m$ blocks in the text that differ by at most k mismatches. We present a new approach for exact-matching, which runs in $O(n^2\log|\sum|)$, $\sum=\{a_1,a_2,\cdots,a_{|\sum|}\}$ is the alphabet of the pattern, and for approximate-matching , in $O(n^2\log|\sum|)+hm^2)$, which consists of two stages. The first stage is to preselect h subblocks of size $s\times s(0\leqslant s\leqslant m)$ that exactly match the pattern. And the second stage of verification compares the blocks containing these h subblocks to determine if the mismatches is no more than k.

Keywords Exact-matching, Approximate-matching, Fast algorithm