An Index for Two Dimensional String Matching Allowing Rotations

Kimmo Fredriksson[1⋆], Gonzalo Navarro[2⋆⋆], and Esko Ukkonen[1⋆⋆⋆]

[1] University of Helsinki, Department of Computer Science,
P.O.Box 26, FIN-00014 Helsinki, Finland,
Fax: +358 9 1914 4441, Kimmo.Fredriksson@cs.Helsinki.FI.
[2] Department of Computer Science, University of Chile.

Abstract. We present an index to search a two-dimensional pattern of size $m \times m$ in a two-dimensional text of size $n \times n$, even when the pattern appears rotated in the text. The index is based on (path compressed) tries. By using $O(n^2)$ (i.e. linear) space the index can search the pattern in $O((\log_\sigma n)^{5/2})$ time on average, where σ is the alphabet size. We also consider various schemes for approximate matching, for which we obtain either $O(\text{polylog}_\sigma n)$ or $O(n^{2\lambda})$ search time, where $\lambda < 1$ in most useful cases. A larger index of size $O(n^2(\log_\sigma n)^{3/2})$ yields an average time of $O(\log_\sigma n)$ for the simplest matching model. The algorithms have applications e.g. in content based information retrieval from image databases.

1 Introduction

Two dimensional pattern (image) matching has important applications in many areas, ranging from science to multimedia. String matching is one of the most successful special areas of algorithmics. Its theory and potential applications in the case of one–dimensional data, that is, linear strings and sequences, is well understood. However, the string matching approach still has considerable unexplored potential when the data is more complicated than just a linear string. Two dimensional digital images are an example of such a data.

Examples of combinatorial pattern matching algorithms that work in two dimensions, but do not allow rotations are e.g. [1,7,11,12,13,14,15]. On the other hand, there are many non–combinatorial approaches to rotation invariant pattern matching, for a review, see e.g. [16]. The only combinatorial methods, that come close to us in some respects, are [12,14]. However, these do not address the pattern rotations. As stated in [2], a major open problem in two–dimensional (combinatorial) pattern matching is to find the occurrences of a two-dimensional

⋆ Work supported by ComBi.
⋆⋆ Work developed while the author was in a postdoctoral stay at the Dept. of Computer Science, Univ. of Helsinki. Partially supported by the Academy of Finland and Fundación Andes.
⋆⋆⋆ Work supported by the Academy of Finland.

J. van Leeuwen et al. (Eds.): IFIP TCS 2000, LNCS 1872, pp. 59–75, 2000.

pattern of size $m \times m$ in a two-dimensional text of size $n \times n$ when the pattern can appear in the text in rotated form. This was addressed from a combinatorial point of view first in [3], in which an online algorithm allowing pattern rotations was presented.

In this work we give the first algorithms for offline searching, that is, for building an index over the text that allows fast querying. The data structure we use is based on tries. Suffix trees for two-dimensional texts have been considered, e.g. in [8,9,10]. The idea of searching a rotated pattern using a "suffix" array of spiral like strings is mentioned in [10], but only for rotations of multiples of 90 degrees. The problem is much more complex if we want to allow any rotation.

In [3] the consideration was restricted to the matches of a pattern inside the text such that the geometric center of the pattern has been put exactly on top of the exact center point of some text cell. This is called the "center–to–center assumption". Under this assumption, there are $O(m^3)$ different relevant rotation angles to be examined for each text cell. In this paper we make this assumption, too, and consider the following four matching models:

Exact: the value of each text cell whose center is covered by some pattern cell must match the value of the covering pattern cell.

Hamming: an extension of the Exact model in which an error threshold $0 \leq k < m^2$ is given and one is required to report all text positions and rotation angles such that at most k text cells do not match the covering pattern cell.

Grays: an extension of the Exact model more suitable for gray level images: the value of each text cell involved in a match must be between the minimum and maximum value of the 9 neighboring cells surrounding the corresponding pattern cell [5].

Accumulated: an extension of the Hamming model, more suitable for gray levels. The sum of the absolute differences between the value of the text cells involved in a match and the values of the corresponding patterns cells must not exceed a given threshold k.

Our results are summarized in Table 1. For some algorithms we have two versions, one with pattern partitioning technique, and one without it. We denote by σ the alphabet size and assume in our average case results that the cell values are uniformly and independently distributed over those σ values. The times reported are average–case bounds for the search. In the Hamming and Accumulated models $\alpha = k/m^2$ (note that $\alpha < 1$ for Hamming and $\alpha < \sigma$ for Accumulated) and k^*_H and k^*_A denote the maximum k values up to where some techniques work: $k^*_H = k/(1 - e/\sigma)$ and $k^*_A = k/(\sigma/(2e) - 1)$. Moreover, $H^H_\sigma(\alpha) = -\alpha \log_\sigma(\alpha) - (1-\alpha)\log_\sigma(1-\alpha)$ and $H^A_\sigma(\alpha) = -\alpha \log_\sigma(\alpha) + (1+\alpha)\log_\sigma(1+\alpha)$. According to Table 1, the search times are sublinear on average when the conditions are met, which implies in particular that $\alpha < 1 - e/\sigma$ for Hamming and $\alpha < \sigma/(2e) - 1$ for Accumulated. In all the cases the index needs $O(n^2)$ space and it can be constructed in average time $O(n^2 \log_\sigma n)$.

We have also considered the alternative model in which the pattern centers are used instead of the text centers. For this case we obtain an index that for the Exact model needs $O(n^2(\log_\sigma n)^{3/2})$ space and gives $O(\log_\sigma n)$ time.

Model	Search time	Condition
Exact	$(\log_\sigma n)^{5/2}$	$\pi m^2/4 \geq \log_\sigma n^2$
Hamming	$(2\log_\sigma n)^{k+3/2}(\sigma/k)^k$	$\pi m^2/4 \geq \log_\sigma n^2 > k^*_H$
Hamming (pattern partitioning)	$n^{2(\alpha+H^H_\sigma(\alpha))}m^5/\log_\sigma n$	$\pi m^2/4 \geq \log_\sigma n^2 > k^*_H$
Grays	$n^{2(1-\log_\sigma(5/4))}(\log_\sigma n)^{3/2}$	$\pi m^2/4 \geq \log_\sigma n^2$
Accumulated	$n^{\log_\sigma 4}(1+2(\log_\sigma n)/k)^k(\log_\sigma n)^{3/2}$	$\pi m^2/4 \geq \log_\sigma n^2 > k^*_A$
Accumulated (pattern partitioning)	$n^{2(\log_\sigma 2+H^A_\sigma(\alpha))}m^5/\log_\sigma n$	$\pi m^2/4 \geq \log_\sigma n^2 > k^*_A$

Table 1. Time complexities achieved under different models.

The algorithms are easily generalized for handling large databases of images. That is, we may store any number of images in the index, and search the query pattern simultaneously from all the images. The time complexities remain the same, if we now consider that n^2 denotes the size of the whole image library.

2 The Data Structures

Let $T = T[1..n, 1..n]$ and $P = P[1..m, 1..m]$ be two dimensional arrays of *point samples*, such that $m < n$. Each sample has a *color* in a finite ordered *alphabet* Σ. The size $|\Sigma|$ of Σ is denoted by σ. The arrays P and T are point samples of colors of some "natural" image. There are several possibilities to define a mapping between T and P, that is, how to compare the colors of P to colors of T. Our approach to the problem is combinatorial. Assume that P has been put on top of T, in some arbitrary position. Then we will compare each color sample of T against the color of the closest sample of P. The distance between the samples is simply the Euclidean distance. This is also technically convenient. The Voronoi diagram for the samples is a regular array of unit squares.

Hence we may define that the array T consists of n^2 unit squares called *cells*, in the real plane $\mathbf{R}^2$ (the (x, y)–plane). The corners of the cell for $T[i, j]$ are $(i-1, j-1), (i, j-1), (i-1, j)$ and (i, j). Each cell has a *center* which is the geometric center point of the cell, i.e., the center of the cell for $T[i, j]$ is $(i-\frac{1}{2}, j-\frac{1}{2})$. The array of cells for pattern P is defined similarly. The *center* of the whole pattern P is the center of the cell in the middle of P. Precisely, assuming for simplicity that m is odd, the center of P is the center of cell $P[\frac{m+1}{2}, \frac{m+1}{2}]$. For images, the cells are usually called *pixels*.

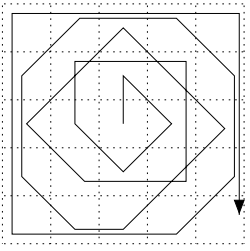
Assume now that P has been moved on top of T using a rigid motion (translation and rotation), such that the center of P coincides exactly with the center of some cell of T. The location of P with respect to T can be uniquely given as $((i-\frac{1}{2}, j-\frac{1}{2}), \theta)$, where $(i-\frac{1}{2}, j-\frac{1}{2})$ is the location of the center of P in T, and θ is the angle between the x–axis of T and the x–axis of P. The occurrence (or more generally, distance) between T and P at some location, is determined by comparing the colors of the cells of T and P that overlap. We will use the centers of the cells of T for selecting the comparison points. That is, for the pattern at location $((i-\frac{1}{2}, j-\frac{1}{2}), \theta)$, we look which cells of the pattern cover the centers of the cells of the text, and compare the corresponding colors of those cells. As

the pattern rotates, the centers of the cells of the text move from one cell of P to another. In [3] it is shown that this happens $O(m^3)$ times, so there are $O(m^3)$ relevant orientations of P to be checked. The actual comparison result of two colors depends on the matching model.

We propose to use a trie based index of the text, defined as follows. Trie is a well–known tree structure for storing strings in which each edge is labeled by a character. Each cell of the text defines a string which is obtained by reading text positions at increasing distances from the center of the cell. The first character is that of the cell, then come the 4 closest centers (from the cells above, below, left and right of the central cell), then the other 4 neighbors, and so on. The cells at the same distance are read in some predefined order, the only important thing is to read the cells in the order of increasing distance from the center cell. This effectively utilizes the $O(m^3)$ result to restrict the number of rotations our algorithms must consider on average, see Sec. 3. If such a string hits the border of the text it is considered finished there. We will call *sistrings* (for "semi–infinite strings") [10] the strings obtained in this way. Figure 1 shows a possible reading order.

22	16	10	17	23
15	5	1	6	18
9	4	0	2	11
14	8	3	7	19
21	13	12	20	24

(a)



(b)

2	56	12	1	3
1	7	2	5	9
3	6	3	19	1
31	28	2	5	23
2	13	12	22	4

(c)

Fig. 1. A possible reading order ("spiral") for the sistring that starts in the middle of a text of size 5×5. Figure *(a)* shows the reading order by enumerating the cells, and figure *(b)* shows the enumeration graphically. Figure *(c)* shows the color values of the cells of the image, and for that image the sistring corresponding to the reading order is $< 3, 2, 19, 2, 6, 7, 5, 5, 28, 3, 12, 1, 12, 13, 31, 1, 56, 1, 9, 23, 22, 2, 2, 3, 4 >$

Therefore each text cell defines a sistring of length $O(n^2)$. A trie on those strings (called the *sistring trie*) can be built, which has average size $O(n^2)$ and average depth $O(\log_\sigma n^2)$. Alternatively, the unary paths of such a trie can be compressed, in similar manner used to compress the suffix trees. In such a tree each new string adds at most one leaf and one internal node, so the worst case size is $O(n^2)$.

Still another possibility is to construct an array of n^2 pointers to T, sorted in the lexicographic order of the n^2 sistrings in T. Such an array, called the *sistring array*, can be formed by reading the leaves of a sistring trie in the lexicographic order, or directly, by sorting the sistrings. The array needs $O(n^2)$ space, but is

much smaller that the sistring trie or tree in practice. Hence the sistring array is the most attractive alternative from the practical point of view and will therefore be used in the experiments.

The sistring trie can be built in $O(n^2 \log_\sigma n)$ average time, by level–wise construction. The sistring array can be built in $O(n^2 \log n)$ string comparisons, which has to be multiplied by $O(\log_\sigma n^2)$ to obtain the average number of character comparisons. The sistring array is very similar to the suffix array, which in turn is a compact representation of a suffix tree.

For simplicity, we describe the algorithms for the sistring trie, although they run with the same complexity over sistring trees. For sistring arrays one needs to multiply the search time results by $O(\log n)$ as well, because searching the array uses binary search.

We consider now a property of the sistring trie that is important for all the results that follow. We show that under a uniform model, the number of sistring trie nodes at depth ℓ is $\Theta(\min(\sigma^\ell, n^2))$. This roughly is to say that in levels $\ell \leq h$, for $h = \log_\sigma(n^2) = 2\log_\sigma n$ all the different strings of length ℓ exist, while from that level on the $\Theta(n^2)$ sistrings are already different. In particular this means that nodes deeper than h have $O(1)$ children because there exists only one sistring in the text with that prefix of length h (note that a sistring prefix is graphically seen as a spiral inside the text, around the corresponding text cell).

To prove this property we consider that there are n^2 sistrings uniformly distributed across σ^ℓ different prefixes, of length ℓ, for any ℓ. The probability of a prefix not being "hit" after n^2 attempts is $(1-1/\sigma^\ell)^{n^2}$, so the average number of different prefixes hit (i.e. existing sistring trie nodes) is

$$\sigma^\ell(1-(1-1/\sigma^\ell)^{n^2}) \;=\; \sigma^\ell(1-e^{-\Theta(n^2/\sigma^\ell)}) \;=\; \sigma^\ell(1-e^{-x})$$

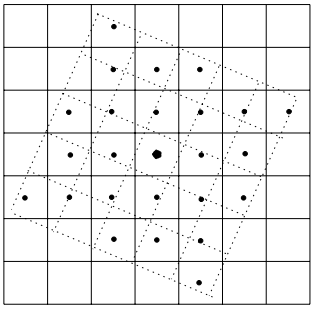
for $x = \Theta(n^2/\sigma^\ell)$. Now, if $n^2 = o(\sigma^\ell)$ then $x = o(1)$ and $1 - e^{-x} = 1 - (1 - x + O(x^2)) = \Theta(x) = \Theta(n^2/\sigma^\ell)$, which gives the result $\Theta(n^2)$. On the other hand, if $n^2 = \Omega(\sigma^\ell)$ then $x = \Omega(1)$ and the result is $\Theta(\sigma^\ell)$. Hence the number of sistring trie nodes at depth ℓ is on average $\Theta(\min(\sigma^\ell, n^2))$, which is the same as the worst case. Indeed, in the worst case the constant is 1, i.e. the number of different strings is at most $\min(\sigma^\ell, n)$, while on average the constant is smaller. We need this result for giving bounds for the maximum number of sistring trie nodes inspected by our algorithms.

3 The Exact Model

We first describe the algorithm for the exact matching model. The other algorithms are relatively straight–forward extensions of it. As shown in [3], there are $O(m^3)$ relevant orientations in which the pattern can occur in a given text position. A brute force approach is to consider the $O(m^3)$ pattern orientations in turn and search each one in the sistring trie. To check the pattern in a given orientation we have to see in which order the pattern cells have to be read so that they match the reading order of the sistring trie construction.

Figure 2 shows the reading order induced in the pattern by a rotated occurrence, using the spiral like reading order given in Figure 1. For each possible rotation we compute the induced reading order, build the string obtained by reading the pattern in that order from its center, and search that string in the sistring trie. Note in particular that some pattern cells may be read twice and others may not be considered at all. Observe that in our example the cells numbered 30, 32, 34, and 36 are outside the maximum circle contained in the pattern, and are therefore ignored in the sistring trie search. This is because those values cannot be used unless some levels are skipped in the search, which would mean entering into all the branches after reading cell number 20. Text cells 21–29, 31, 33, 35, and 37– all fall outside the pattern.

The algorithm first considers the sistring of P for angle $\theta = 0$. The sistring is searched from the trie until some cell of P mismatches, at depth ℓ. Now the pattern must be rotated a little in order to read the next sistring. The next rotation to try is such that any of the first ℓ cells of the previous sistring changes, that is, any of the centers of the cells of T hits some border of the first ℓ sistring cells of P.



(a)

34	10	17	18	*36*
16	5	1	6	11
15	4	0	2	19
9	8	3	7	20
32	14	13	12	*30*

(b)

2	19	7	1	3
1	7	2	5	9
3	6	3	4	1
9	12	2	5	3
2	7	6	5	4

(c)

Fig. 2. Reading order induced in the pattern by a rotated occurrence. Figure *(a)* shows the pattern superimposed on the image, Figure *(b)* shows the enumeration of the induced reading order, and Figure *(c)* shows the color values for the pattern cells. The corresponding sistring is $< 3, 2, 4, 2, 6, 7, 5, 5, 12, 9, 19, 9, 5, 6, 7, 3, 1, 7, 1, 1, 3 >$. Cells numbered 30, 32, 34, and 36 are ignored in the trie search.

The number of rotations to try depends on how far we are from the center. That is, the number of the text centers that any cell of P may cover depends on how far the cell is from the rotation center. If the distance of some cell of P from the rotation center id d, then it may cover $O(d)$ center of T. In general, there are $O(m^3)$ rotations for a pattern of size $m \times m$ cells. The number of rotations grows as we get farther from the center, and they are tried only on the existing branches of the sistring trie. As the pattern is read in a spiral form, when we are at depth ℓ in the sistring trie we are considering a pattern cell which is at distance $O(\sqrt{\ell})$ from the center. This means that we need to consider $O(\ell^{3/2})$

different rotations if the search has reached depth ℓ. For each rotation we assume in the analysis that the sistring is read and compared from the beginning.

The fact that on average every different string up to length h exists in the sistring trie means that we enter always until depth h. The number of sistring trie nodes considered up to depth h is thus

$$\sum_{\ell=1}^{h} \ell^{3/2} \quad = \quad O(h^{5/2})$$

At this point we have $O(h^{3/2})$ candidates that are searched deeper in the sistring trie. Now, each such candidate corresponds to a node of the sistring trie at depth h, which has $O(1)$ children because there exist $O(1)$ text sistrings which share this prefix with the pattern (a "prefix" here means a circle around the pattern center).

Two alternative views for the same process are possible. First, consider all the $O(h^{3/2})$ candidates together as we move to deeper levels $\ell > h$ in the sistring trie. There are on average n^2/σ^ℓ sistrings of length ℓ matching a given string, so the total work done when traversing the deeper levels of the sistring trie until the candidates get eliminated is

$$\sum_{\ell \geq h+1} \frac{n^2}{\sigma^\ell} \ell^{3/2} \quad = \quad \sum_{\ell \geq 1} \frac{(\ell+h)^{3/2}}{\sigma^\ell} \quad = \quad O(h^{3/2})$$

An alternative view is that we directly check in the text each candidate that arrived to depth h, instead of using the sistring trie. There are $O(h^{3/2})$ candidates and each one can be checked in $O(1)$ time: if we perform the comparison in a spiral way, we can add the finer rotations as they become relevant. The ℓ-th pattern cell in spiral order (at distance $\sqrt{\ell}$ from the center) is compared (i.e. the comparison is not abandoned before) with probability $\ell^{3/2}/\sigma^\ell$. Summing up the probabilities of being compared over all the characters yields

$$\sum_{\ell \geq 1} \frac{\ell^{3/2}}{\sigma^\ell} \quad = \quad O(1)$$

where for simplicity we have not considered that we have compared already h characters and have an approximate idea of the orientations to try.

Therefore we have a total average search cost of $O((\log_\sigma n)^{5/2})$. This assumes that the pattern is large enough, i.e. that we can read h characters from the center in spiral form without hitting the border of the pattern. This is equivalent to the condition $m^2 \geq \frac{4}{\pi} \log_\sigma n^2$ which is a precondition for our analysis. Smaller patterns leave us without the help of the sistring trie long before we have eliminated enough candidates to guarantee low enough average time to check their occurrences in the text.

4 The Hamming Model

This model is an extension of the exact matching model. An additional parameter k is provided to the search algorithm, such that a mismatch occurs only when more than k characters have not matched. In this section we use $\alpha = k/m^2$. We require $0 \leq \alpha < 1$, as otherwise the pattern would match everywhere.

The problem is much more complicated now. Even for a fixed rotation, the number of sistring trie nodes to consider grows exponentially with k. To see this, note that at least k characters have to be read in all the sistrings, which gives a minimum of $O(\sigma^k)$ nodes. This means in particular that if $k \geq h$ then we will consider the n^2 sistrings and the index will be of no use, so we assume $k < h$; still stricter conditions will appear later. We first present a standard technique and then a pattern partitioning technique.

4.1 Standard Searching

Imagine that for each possible rotation we backtrack on the sistring trie, entering into all the possible branches and abandoning a path when more than k mismatches have occurred. As explained, up to depth k we enter into all the branches. Since $h > k$, we have to analyze which branches we enter at depths $k < \ell \leq h$. Since all those strings exist in the sistring trie, this is the same as to ask how many different strings of length ℓ match a pattern prefix of length ℓ with at most k mismatches.

A pessimistic model assumes that there are $\binom{\ell}{k}$ ways to choose the cells that will not match, and σ^k selections for them. As we can replace a cells by itself we are already counting the cases with less than k errors. The model is pessimistic because not all these choices lead to different strings. To all this we have to add the fact that we are searching $O(\ell^{3/2})$ different strings at depth ℓ. Hence, the total number of sistring trie nodes touched up to level h is

$$\sum_{\ell=1}^{k} \ell^{3/2}\sigma^\ell \;+\; \sum_{\ell=k+1}^{h} \ell^{3/2}\binom{\ell}{k}\sigma^k \;=\; O\left(h^{3/2}\sigma^k\binom{h}{k}\right)$$

For the second part of the search, we consider that there are on average n^2/σ^ℓ sistrings of length $\ell > h$ equal to a given string. So the total amount of nodes touched after level h is

$$\sum_{\ell \geq h+1} \ell^{3/2}\binom{\ell}{k}\sigma^k\frac{n^2}{\sigma^\ell} \;=\; \sum_{\ell \geq h+1} \ell^{3/2}n^2\binom{\ell}{k}\frac{1}{\sigma^{\ell-k}}$$

In the Appendix A we show that $\binom{\ell}{k}\frac{1}{\sigma^{\ell-k}}$ is exponentially decreasing with ℓ for

$$\ell \;>\; k^*_H \;=\; \frac{k}{1-e/\sigma}$$

while otherwise it is $\Omega(1/\sqrt{\ell})$.

Therefore, the result depends on whether or not $h > k_H^*$. If $h \leq k_H^*$, then the first term of the summation alone is $\Omega(hn^2)$ and there is no sublinearity. If, on the other hand, $h > k_H^*$, we have an exponentially decreasing series where the first term dominates the whole summation. That is, the cost of the search in levels deeper than h is

$$h^{3/2}n^2\binom{h}{k}\frac{1}{\sigma^{h-k}} \quad = \quad h^{3/2}\binom{h}{k}\sigma^k \quad = \quad O\left((\log_\sigma n)^{k+3/2}(\sigma/k)^k\right)$$

which matches the cost of the first part of the search as well. Therefore, the condition for a sublinear search time is $k_H^* < h < m^2$. This in particular implies that $\alpha < 1 - e/\sigma$.

4.2 Pattern Partitioning

The above search time is still polylogarithmic in n, but exponential in k. We present now a *pattern partitioning* technique that obtains a cost of the form $O(n^{2\lambda})$ for $\lambda < 1$. The idea is to split the pattern in j^2 pieces (j divisions across each coordinate). If there are at most k mismatches in a match, then at least one of the pieces must have at most $\lfloor k/j^2 \rfloor$ errors. So the technique is to search for each of the j^2 pieces (of size $(m/j) \times (m/j)$) separately allowing k/j^2 errors, and for each (rotated) match of a piece in the text, go to the text directly and check if the match can be extended to a complete occurrence with k errors. Note that the α for the pieces is the same as for the whole pattern.

The center–to–center assumption does not hold when searching for the pieces. However, for each possible rotation of the whole pattern that matches with the center–to–center assumption, it is possible to fix some position of the center of each piece inside its text cell. (The center of the piece is ambiguous, as there are infinitely many angles for the matching pattern: there are $O(m^3)$ different relevant rotations of the pattern, and between the corresponding angles, there are infinitely many angles where the occurrence status does not change. However, any of the possible positions for the center of the pieces can be chosen). The techniques developed to read the text in rotated form can be easily adapted to introduce a fixed offset at the center of the matching subpattern. Therefore we search each of the j^2 pieces in every of the $O(m^3)$ different rotations.

The search cost for this technique becomes j^2m^3 times the cost to search a piece (with a fixed rotation and center offset) in the sistring trie and the cost to check for a complete occurrence if the piece is found.

If we consider that $(m/j)^2 \leq h$, then all the strings exist when a piece is searched. Therefore the cost to traverse the sistring trie for a piece at a fixed rotation is equivalent to the number of strings that can be obtained with k mismatches from it, i.e.

$$U = \binom{(m/j)^2}{k/j^2}\sigma^{k/j^2}$$

while the cost to check all the U candidates is $Ukn^2/\sigma^{(m/j)^2}$, i.e. k times per generated string times the average number of times such a string appears in the

text. Therefore the overall cost is

$$j^2 m^3 \left(U \ + \ Uk \frac{n^2}{\sigma^{(m/j)^2}} \right)$$

where (after distributing the initial factor) the first term decreases and the second term increases as a function of j. The optimum is found when both terms meet, i.e. $j = m/\sqrt{2 \log_\sigma n}$ which is in the limit of our condition $(m/j)^2 \leq h$. In fact, the second term is decreasing only for $\alpha < 1 - e/\sigma$, otherwise the optimum is $j = 1$, i.e. no pattern partitioning.

For this optimal j, the overall time bound becomes

$$O\left(\frac{m^5}{\log_\sigma n} \ n^{2(\alpha + H_\sigma^H(\alpha))} \right)$$

where we have written $H_\sigma^H(\alpha) = -\alpha \log_\sigma(\alpha) - (1-\alpha) \log_\sigma(1-\alpha)$.

This bound is sublinear as long as $\alpha < 1 - e/\sigma$. On the other hand, we can consider to use a larger j, violating the assumed condition $(m/j)^2 \leq h$ in order to reduce the verification time. However, the search time will not be reduced and therefore the time bound cannot decrease.

5 The Grays Model

In the Grays model a match requires that the color of text cell must be between the minimum and maximum pattern colors in a neighborhood of the pattern cell that corresponds to the text cell. In this case, we do not enter into a single branch of the sistring trie, but for each pattern cell we follow all branches where color is between the minimum and maximum neighbor of that pattern cell. The number of characters qualifying for the next pattern character is a random variable that we call Δ, where $1 \leq \Delta \leq \sigma$.

Since there are now $O(\Delta^\ell)$ possible strings that match the pattern prefix of length ℓ, we touch

$$\sum_{\ell=1}^{h} \ell^{3/2} \Delta^\ell \ = \ O(h^{3/2} \Delta^h)$$

sistring trie nodes up to depth h, because all those qualifying strings exist up to that depth. From that depth on, there are on average $O(n^2/\sigma^\ell)$ sistrings in the text matching a given string of length ℓ. Therefore, the work in the deeper part of the sistring trie is

$$\sum_{\ell > h} \ell^{3/2} \ \frac{n^2}{\sigma^\ell} \ \Delta^\ell \ = \ O\left(h^{3/2} \ \frac{n^2}{\sigma^h} \ \Delta^h \right) \ = \ O(h^{3/2} \Delta^h)$$

since the first term of the summation dominates the rest. Therefore, the total complexity is

$$O(h^{3/2} \Delta^h) \ = \ O\left((\log_\sigma n)^{3/2} n^{2 \log_\sigma \Delta} \right) \ = \ O\left((\log_\sigma n)^{3/2} n^{2(1 - \log_\sigma 5/4)} \right).$$

Here the last step is based on that $\overline{\Delta}$, the average value of Δ, equals $(4/5)\sigma$ as the difference between the maximum and minimum of 9 values uniformly distributed over σ On the other hand, the cost function is concave in terms of Δ, and hence $\overline{f(\Delta)} \leq f(\overline{\Delta})$. In practice $\overline{\Delta}$ is much less than $(4/5)\sigma$, see [5].

6 The Accumulated Model

Even more powerful model is the Accumulated model, which provides a Hamming-like matching capability for gray-level images. Here, the sum of the absolute differences between text colors and the color of the corresponding pattern cell must not exceed k.

As for the Hamming model, we have to enter, for each relevant rotation, into all the branches of the sistring trie until we obtain an accumulated difference larger than k. We present first a standard approach and then a pattern partitioning technique.

6.1 Standard Searching

We enter into all the branches of the sistring trie until we can report a match or the sum of the differences exceeds k. As we show in Appendix B, the number of strings matching a given string of length ℓ under this model is at most $2^\ell \binom{k+\ell}{k}$. Since up to length h all them exist, we traverse

$$\sum_{\ell=1}^{h} \ell^{3/2} 2^\ell \binom{k+\ell}{k} \;=\; O\left(h^{3/2} 2^h \binom{k+h}{k}\right)$$

nodes in the trie. For the deeper parts of the trie there are $O(n^2/\sigma^\ell)$ strings matching a given one on average, so the rest of the search takes

$$\sum_{\ell>h} \ell^{3/2}\ \frac{n^2}{\sigma^\ell}\ 2^\ell \binom{k+\ell}{k} \;=\; \sum_{\ell>h} \ell^{3/2}\ n^2\ \frac{2^\ell}{\sigma^\ell} \binom{k+\ell}{k}$$

In Appendix B we show that $(2/\sigma)^\ell \binom{k+\ell}{k}$ is exponentially decreasing with ℓ for $k/\ell < \sigma/(2e) - 1$, otherwise it is $\Omega(1/\sqrt{\ell})$. Therefore, we define

$$k_A^* \;=\; \frac{k}{\sigma/(2e) - 1}$$

and if $h \leq k_A^*$ the summation is at least $O(hn^2)$ and therefore not sublinear. If, on the other hand, $h > k_A^*$, then the first term of the summation dominates the rest, for a total search cost of

$$O\left(h^{3/2} 2^h \binom{k+h}{k}\right) \;=\; O\left((\log_\sigma n)^{3/2} n^{2\log_\sigma 2} \left(1 + \frac{2\log_\sigma n}{k}\right)^k\right)$$

which is sublinear in n for $\sigma > 2$. On the other hand, $\sigma = 2$ means a bilevel image, where the Hamming model is the adequate choice. Hence we obtain sublinear complexity (albeit exponential on k) for $k_A^* < 2\log_\sigma n$.

6.2 Pattern Partitioning

As for the Hamming model, we can partition the pattern to j^2 subpatterns that are searched exhaustively in the sistring trie. Again considering $(m/j)^2 \leq h$ we have a total search cost of

$$j^2 m^3 \left(U \ + \ Uk \frac{n^2}{\sigma^{(m/j)^2}} \right)$$

where this time

$$U = 2^{(m/j)^2} \binom{(m/j)^2 + k/j^2}{k/j^2}$$

After distributing the initial factor of the cost formula, we see that the first term decreases and the second term increases as a function of j. The optimum is found when both terms meet, which is again $j \ = \ m/\sqrt{2 \log_\sigma n}$ which is consistent with our condition $(m/j)^2 \leq h$. In fact, the second term is decreasing only for $\alpha < \sigma/(2e) - 1$, otherwise the optimum is $j = 1$, i.e. no pattern partitioning.

For this optimal j, the overall complexity is

$$O\left(\frac{m^5}{\log_\sigma n} \ n^{2(\log_\sigma 2 + H_\sigma^A(\alpha))} \right)$$

where we have defined $H_\sigma^A(\alpha) = -\alpha \log_\sigma(\alpha) + (1+\alpha) \log_\sigma(1+\alpha)$.

This complexity is sublinear as long as $\alpha < \sigma/(2e) - 1$. Again, we can consider to use a larger j value but the complexity does not improve.

7 An Alternative Matching Model

We have considered up to now that text centers match the value of the pattern cells they lie in. This has been done for technical convenience, although an equally reasonable alternative model is that the pattern cells must match the text color where their centers lie in the text.

Except for the Grays model, all the algorithms considered can be adapted to this case. The algorithms are more complex in practice now, because there may be more than one pattern center lying at the same text cell, and even no pattern center at all. This means that in some branches of the sistring trie we may have more than one condition to fit (which may be incompatible and then the branch can be abandoned under some models) and there may be no condition at all, in which case we have to follow all the branches at that level of the trie.

On average, however, we still have $\Theta(\ell)$ conditions when entering in the sistring trie with a pattern string of length ℓ, and therefore all the time bounds remain the same. However, in the Exact matching model, we can do better using the pattern centers.

We consider now indexing the rotated versions of the text sistrings, instead of considering the rotated versions of the pattern at search time. Hence, the pattern is simply searched with no rotations. Imagine that we index all the rotations of

the text up to depth H. This means that there will be $O(n^2H^{3/2})$ sistrings, and the sizes of the sistring trie and array will grow accordingly.

The benefit comes at search time: in the first part of the search we do not need to consider rotations of the pattern, since all the rotated ways to read the text are already indexed. Since we index $O(n^2H^{3/2})$ strings now, all the different sistrings will exist until depth $h' = \log_\sigma(n^2H^{3/2}) = 2\log_\sigma n + 3/2\log_\sigma H$. We first assume that $H \geq h'$. This means that until depth H we pay $O(H)$. After that depth all the surviving rotations are considered. Since $H \geq h'$, they all yield different strings, and the summation, as in Section 3, yields $O(n^2H^{3/2}/\sigma^H)$. Therefore the total search time is

$$O\left(H + \frac{n^2H^{3/2}}{\sigma^H}\right)$$

which is optimized for $H = 2\log_\sigma n + (1/2)\log_\sigma H$. Since this is smaller than h' we take the minimal $H = h'$. For instance $H = x\log_\sigma n$ works for any $x > 2$.

This makes the total search time $O(\log_\sigma n)$ on average. The space complexity becomes now $O(n^2(\log_\sigma n)^{3/2})$. Trying to use $H < h'$ worsens the complexity.

The matching model has changed, however. In the normal index the text sistrings are indexed once at a fixed rotation (zero). When a given pattern rotation is tried, the pattern is read in rotated form, in an order driven by the text centers. Now the text sistrings are read in all the rotated forms, and the pattern will be read once. The way to index a text sistring in rotated form is to assume that a rotated pattern is superimposed onto it and read the text cells where the pattern cells, read in order, fall. This effectively corresponds to the model we are considering in this section.

8 Experimental Results (Preliminary)

We have implemented the algorithms for Exact and Accumulated models, without the pattern partitioning technique. For the index structure, we used sistring array. The array based implementation is much more space efficient, but the search cost is also higher (both asymptotically and by the constant factors).

The implementation is in C, compiled using `gcc 2.95.2` on Linux 2.0.38, running in 700MHz PentiumIII machine. The implementation is not very optimized, and much of the time is spent in trigonometry; for computing the next angle to try, and for computing the coordinates of the cells for the given angle. Our test text was an image of size 768×768 cells, with 35 colors (gray levels), and a pattern of size 41×41 was extracted from it.

Table 2 shows some timings for the search. The difference between the times of the Exact model and the Accumulated model with $k = 0$ reveals the more complex implementation of the Accumulated model. Our results show that the algorithms can be implemented, and although preliminary versions, they run reasonably fast. For comparison, our optimized $O(n^2(k/\sigma)^{3/2})$ expected time on–line algorithm [4] spends 0.81 seconds for $k = 0$, 1.67 seconds for $k = 8$, 3.62 seconds for $k = 192$, and 3.85 seconds for $k = 256$. With the pattern partitioning, the algorithms would be much faster for large k.

k	Exact	0	1	2	4	8	16	32	64	96	128	192	256
time	0.0055	0.0086	0.0088	0.0089	0.0090	0.0097	0.0110	0.0165	0.0567	0.1720	0.3930	2.0390	6.3910

Table 2. Experimental results for the Exact and Accumulated models. The times are given in seconds.

9 Conclusions and Future Work

We have proposed a sistring tree index to search two dimensional patterns in two dimensional texts allowing rotations. We have considered different matching models and obtained average time bounds that are sublinear for most reasonable cases.

It is possible to extend the model by removing the center–to–center assumption [4]. In this case the number of patterns grows as high as $O(m^7)$ and therefore there are $O(\ell^{7/2})$ sistrings to search at depth ℓ. The search time for the Exact model becomes $O(\log_\sigma n)^{9/2}$. By indexing all the rotations and center displacements we get $O(\log_\sigma n)$ time again, but at a space cost of $O(n^2(\log_\sigma n)^{7/2})$.

It is also possible to extend the methods to three dimensions [6]. With the center–to–center assumption we have $O(m^{11})$ rotations. This means $O(\ell^{11/3})$ sistrings at depth ℓ. Therefore, at $O(n^3)$ space the total search time becomes $O((\log_\sigma n)^{14/3})$ for exact searching. If we index all the rotations up to $H = x \log_\sigma n$ with $x > 3$ we will have a space requirement of $O(n^3(\log_\sigma n)^{11/3})$ and a search cost of $O(\log_\sigma n)$. For the Grays model we have $O((\log_\sigma n)^{11/3} n^{3(1-\log_\sigma (28/27))})$ time. All the other techniques can be extended as well.

References

1. A. Amir, G. Benson, and M. Farach. Alphabet independent two dimensional matching. In N. Alon, editor, *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing*, pages 59–68, Victoria, B.C., Canada, May 1992. ACM Press.
2. A. Amir. Multidimensional pattern matching: A survey. Technical Report GIT-CC-92/29, Georgia Institute of Technology, College of Computing, 1992.
3. K. Fredriksson and E. Ukkonen. A rotation invariant filter for two-dimensional string matching. In *Proceedings of the 9th Annual Symposium on Combinatorial Pattern Matching (CPM'98)*, LNCS 1448, pages 118–125, 1998.
4. K. Fredriksson and E. Ukkonen. Algorithms for 2–d hamming distance under rotations. Manuscript, 1999.
5. K. Fredriksson and E. Ukkonen. Combinatorial methods for approximate image matching under translations and rotations. *Pattern Recognition Letters*, 20(11–13):1249–1258, 1999.
6. K. Fredriksson and E. Ukkonen. Combinatorial methods for approximate pattern matching under rotations and translations in 3D arrays. Submitted, 2000.
7. Z. Galil and K. Park. Truly alphabet-independent two-dimensional pattern matching. In IEEE, editor, *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 247–257, Pittsburgh, PN, October 1992. IEEE Computer Society Press.

8. R. Giancarlo. A generalization of suffix trees to square matrices, with applications. *SIAM J. on Computing*, 24:520–562, 1995.
9. R. Giancarlo and R. Grossi. On the construction of classes of suffix trees for square matrices: Algorithms and applications. *Information and Computation*, 130:151–182, 1996.
10. G. H. Gonnet. Efficient searching of text and pictures. Report OED-88-02, University of Waterloo, 1988.
11. J. Kärkkäinen and E. Ukkonen. Two and higher dimensional pattern matching in optimal expected time. In Daniel D. Sleator, editor, *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 715–723, Arlington, VA, January 1994. ACM Press.
12. G. M. Landau and U. Vishkin. Pattern matching in a digitized image. *Algorithmica*, 12(4/5):375–408, October 1994.
13. G. Navarro and R. Baeza-Yates. Fast multi-dimensional approximate string matching. In *Proceedings of the 10th Annual Symposium on Combinatorial Pattern Matching (CPM'99)*, LNCS, pages 243–257, 1999.
14. T. Takaoka. Approximate pattern matching with grey scale values. In Michael E. Houle and Peter Eades, editors, *Proceedings of Conference on Computing: The Australian Theory Symposium*, pages 196–203, Townsville, January 29–30 1996. Australian Computer Science Communications.
15. J. Tarhio. A sublinear algorithm for two-dimensional string matching. *PRL: Pattern Recognition Letters*, 17, 1996.
16. J. Wood. Invariant pattern recognition: a review. *Pattern Recognition*, 29(1):1–17, 1996.

A Probability of Matching under the Hamming Model

We need to determine which is the probability of the search being active at a given node of depth ℓ in the sistring trie under the Hamming model. We are therefore interested in the probability of a pattern prefix of length ℓ matching a text substring of length ℓ. For this to hold, at least $\ell - k$ text characters text must match the pattern. Hence, the probability of matching is upper bounded by

$$\frac{1}{\sigma^{\ell-k}}\binom{\ell}{k}$$

where the combinatorial counts all the possible locations for the matching characters.

In the analysis that follows, we call $\beta = k/\ell$ and take it as a constant (which is our case of interest, as seen later). We will prove that, after some length ℓ, the matching probability is $O(\gamma(\beta)^\ell)$, for some $\gamma(\beta) < 1$. By using Stirling's approximation $x! = (x/e)^x\sqrt{2\pi x}(1 + O(1/x))$ over the matching probability we have

$$\frac{1}{\sigma^{\ell-k}}\left(\frac{\ell^\ell\sqrt{2\pi\ell}}{k^k(\ell-k)^{\ell-k}\sqrt{2\pi k}\sqrt{2\pi(\ell-k)}}\right)\left(1+O\left(\frac{1}{\ell}\right)\right)$$

which is

$$\left(\frac{1}{\sigma^{1-\beta}\beta^\beta(1-\beta)^{1-\beta}}\right)^\ell \ \ell^{-1/2}\ \left(\frac{1}{\sqrt{2\pi\beta(1-\beta)}}+O\left(\frac{1}{\ell}\right)\right)$$

This formula is of the form $\gamma(\beta)^\ell\ O(1/\sqrt{\ell})$, where we define

$$\gamma(x) \quad = \quad \frac{1}{\sigma^{1-x}x^x(1-x)^{1-x}}$$

Therefore the probability is exponentially decreasing with ℓ if and only if $\gamma(\beta) < 1$, that is,

$$\sigma \quad > \quad \left(\frac{1}{\beta^\beta(1-\beta)^{1-\beta}}\right)^{\frac{1}{1-\beta}} \quad = \quad \frac{1}{\beta^{\frac{\beta}{1-\beta}}(1-\beta)}$$

It is easy to show analytically that $e^{-1} \leq \beta^{\frac{\beta}{1-\beta}} \leq 1$ if $0 \leq \beta \leq 1$, so it suffices that $\sigma > e/(1-\beta)$, or equivalently, $\beta < 1 - e/\sigma$ is a sufficient condition for the probability to be exponentially decreasing with ℓ.

Hence, the result is that the matching probability is very high for $\beta = k/\ell \geq 1 - e/\sigma$, and that otherwise it is $O(\gamma(\beta)^\ell/\sqrt{\ell})$, where $\gamma(\beta) < 1$.

B Probability of Matching under the Accumulated Model

We need to determine what is the probability of the search being active at a given node of depth ℓ in the sistring trie under the Accumulated model. We are therefore interested in the probability of two random strings of length ℓ matching with at most k errors. Our model is as follows: we consider the sequence of ℓ absolute differences between both strings $\delta_1 \ldots \delta_\ell$. The matching condition states that $\sum_{i=1}^{\ell} \delta_i \leq k$.

The number of different sequences of differences satisfying this is $\binom{k+\ell}{\ell}$, what can be seen as the number of ways to insert ℓ divisions into a sequence of k elements. The ℓ divisions divide the sequence in $\ell + 1$ zones. The sizes of the first ℓ zones are the δ_i values and the last allows the sum to be $\leq k$ instead of exactly k. Note that we are pessimistically forgetting about the fact that indeed $\delta_i \leq \sigma$.

Finally, each difference δ_i can be obtained in two ways: $P_i + \delta_i$ and $P_i - \delta_i$ (we again pessimistically count twice the case $\delta_i = 0$). Therefore, the total matching probability is upper bounded by

$$\frac{2^\ell}{\sigma^\ell}\binom{\ell+k}{k}$$

In the analysis that follows, we call $\beta = k/\ell$ and take it as a constant (which is our case of interest, as seen later). We will prove that, after some length ℓ, the matching probability is $O(\gamma(\beta)^\ell)$, for some $\gamma(\beta) < 1$. By using Stirling's approximation $x! = (x/e)^x\sqrt{2\pi x}(1 + O(1/x))$ over the matching probability we have

$$\frac{2^\ell}{\sigma^\ell}\left(\frac{(k+\ell)^{k+\ell}\sqrt{2\pi(k+\ell)}}{k^k\ell^\ell\sqrt{2\pi k}\sqrt{2\pi\ell}}\right)\left(1 + O\left(\frac{1}{\ell}\right)\right)$$

which is

$$\left(\frac{2(1+\beta)^{1+\beta}}{\sigma\beta^{\beta}}\right)^{\ell} \ell^{-1/2} \left(\sqrt{\frac{1+\beta}{2\pi\beta}} + O\left(\frac{1}{\ell}\right)\right)$$

This formula is of the form $\gamma(\beta)^{\ell}\, O(1/\sqrt{\ell})$, where we define

$$\gamma(x) \quad = \quad \frac{2(1+x)^{1+x}}{\sigma x^{x}}$$

Therefore the probability is exponentially decreasing with ℓ if and only if $\gamma(\beta) < 1$, that is,

$$\frac{2(1+\beta)}{\sigma}\left(1+\frac{1}{\beta}\right)^{\beta} \quad < \quad 1$$

It can be easily seen analytically that $(1+1/\beta)^{\beta} \leq e$, so $\beta < \sigma/(2e) - 1$ is a sufficient condition for the probability to be exponentially decreasing with ℓ.

Hence, the result is that the matching probability is very high for $\beta = k/\ell \geq \sigma/(2e) - 1$, and that otherwise it is $O(\gamma(\beta)^{\ell}/\sqrt{\ell})$, where $\gamma(\beta) < 1$.