A Rotation Invariant Filter for Two–dimensional String Matching[1]

Kimmo Fredriksson and Esko Ukkonen

Department of Computer Science, University of Helsinki
PO Box 26, FIN–00014 Helsinki, Finland
kfredrik@cs.Helsinki.FI, ukkonen@cs.Helsinki.FI

Abstract. We consider the problem of finding the occurrences of two–dimensional pattern $P[1..m, 1..m]$ in two–dimensional text $T[1..n, 1..n]$ when also rotations of P are allowed. A fast filtration–type algorithm is developed that finds in T the locations where a rotated P can occur. The corresponding rotations are also found. The algorithm first reads from P a linear string of length m in all $O(m^2)$ orientations that are relevant. We also show that the number of different orientations which P can have is $O(m^3)$. The text T is scanned with Aho–Corasick string matching automaton to find the occurrences of any of these $O(m^2)$ linear strings of length m. Each such occurrence indicates a potential set of occurrences of whole P which are then checked. Some preliminary running times of a prototype implementation of the method are reported.

1 Introduction

The pattern matching problem with fixed orientation of the pattern is a well studied problem. The classical problem is to find the exact or approximate occurrences of pattern P from text T, when the orientation of P is known *a priori*. See e.g. [2, 6, 4, 8, 7, 11].

The idea of reducing the problem of finding fixed orientation occurrences of a two–dimensional pattern to the problem of finding one–dimensional string is not a new one [12, 13, 3]. However, in this paper we present a filtration method that reduces the problem of finding a rotated two–dimensional pattern to the well studied problem of finding the occurrences of a set of strings. The method is based on a careful study of the combinatorial structure of the problem.

Other methods for rotation invariant pattern matching are e.g. geometric hashing [9], template matching [5] and various methods of log–polar transformation and Fourier transform techniques; for a summary see e.g. [5].

[1] A work supported by the Academy of Finland under grant 8745.

An interesting variation of two dimensional pattern matching problem and techniques to solve it was presented in [10] by Landau and Vishkin. They did not consider the rotations of the pattern, but a pattern matching problem in which the pattern and the text are specified in terms of their "continuous" properties. The pattern matching problems arising from this point of view and the techniques used to solve the problem are somewhat similar to ours.

2 Problem Definition

Let *pattern* $P = P[1..m, 1..m]$ be a two–dimensional $m \times m$ array and *text* $T = T[1..n, 1..n]$ an $n \times n$ array over some finite alphabet Σ, such that $m < n$ (see Fig. 1). For simplicity we restrict the consideration on square arrays only.

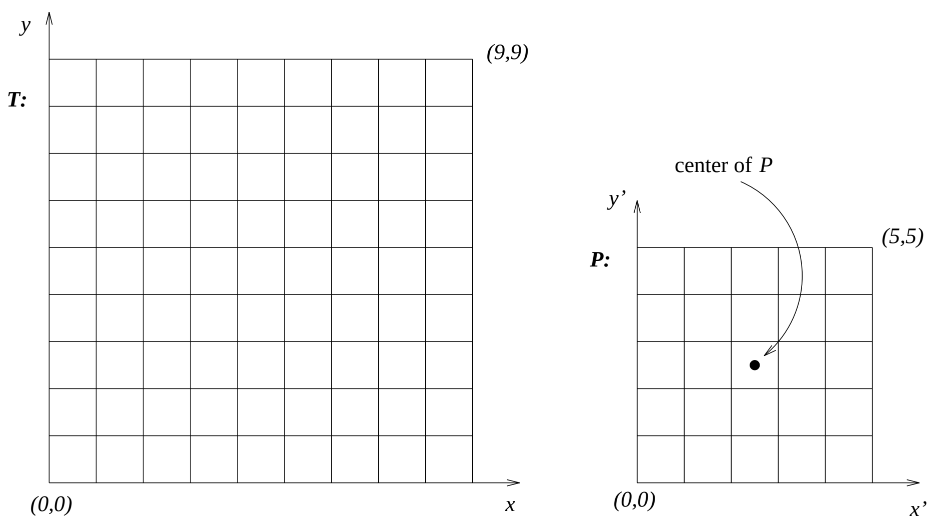

Fig. 1. The array of pixels for T and P ($n = 9$, $m = 5$).

To define a possibly rotated occurrence of P in T we need a geometric interpretation of P and T. To this end, we say that the array of *unit pixels* for T consists of n^2 unit squares called *pixels*, in the real plane $\mathbf{R^2}$ ((x, y)–plane). The four corners of the pixel for $T[1, 1]$ are points $(0, 0), (0, 1), (1, 0)$ and $(1, 1)$ in $\mathbf{R^2}$, and generally, the corners of the pixel for $T[i, j]$ are $(i-1, j-1), (i, j-1), (i-1, j)$ and (i, j). Hence the pixels for T form a regular $n \times n$ array covering the area between $(0, 0), (n, 0), (0, n)$ and (n, n). Each pixel has a *center* which is the geometric center point of the pixel, i.e., the center of the pixel for $T[i, j]$ is $(i - \frac{1}{2}, j - \frac{1}{2}) \in \mathbf{R^2}$.

The array of pixels for pattern P is defined similarly. In the sequel we identify the original array entries with the corresponding pixels and speak of, say, pixel

$T[i,j]$ whose value is in Σ. The values are called *colors*; hence each pixel $T[i,j]$ has a color in Σ.

The *center* of the whole pattern P is the center of the pixel which is in the middle of P. Precisely, assuming for simplicity that m is odd, the center of P is the center of pixel $P[\frac{m+1}{2}, \frac{m+1}{2}]$, that is, point $(\frac{m}{2}, \frac{m}{2}) \in \mathbf{R^2}$.

Consider now a rigid motion (translation and rotation) that moves P on top of T. We restrict the consideration on the special case such that the translation moves the center of P exactly on top of the center of some pixel of T. We call this the *center–to–center translation*.

Assume that the center of P is on top of the center of $T[i,j]$ after the translation. Then some rotation is applied that creates angle α between the x–axis of T and the x–axis of P (see Fig. 2). We say now that P is at *location* $((i,j),\alpha)$ on top of T.

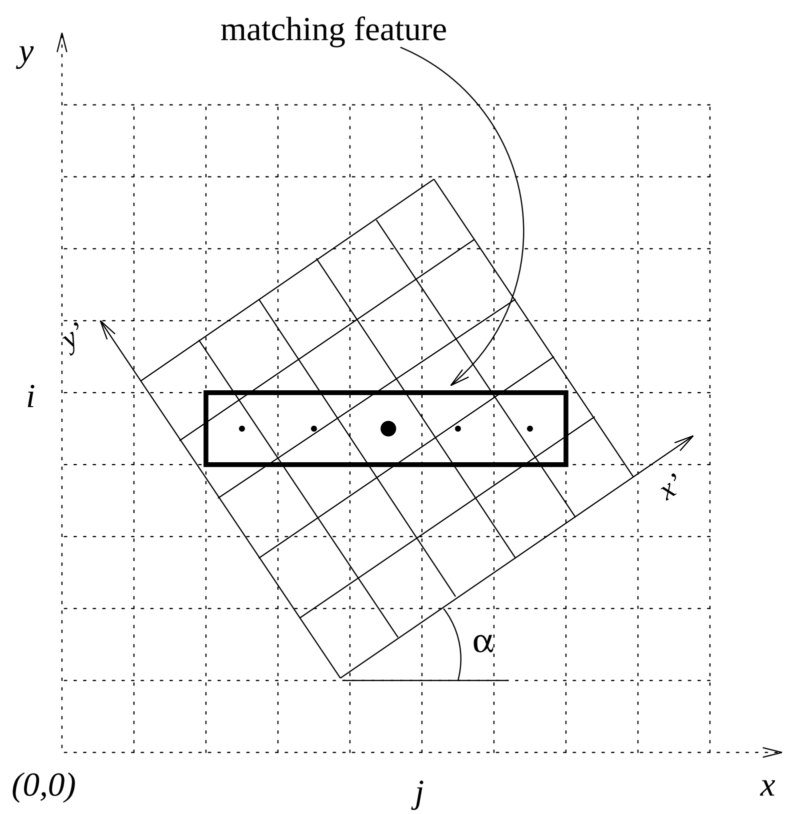

Fig. 2. An occurrence with a matching feature.

Pattern P is said to have an *occurrence* at location $((i,j),\alpha)$ if the pixel colors of P match with the pixel colors of T below them. More precisely, we will require that the colors match at the centers of the pixels of P. To this end, let us define the *matching function* M at location $((i,j),\alpha)$ as follows. For each pixel $P[r,s]$ of P, let $T[r',s']$ be a pixel of T such that the center of $P[r,s]$ belongs to the area covered by $T[r',s']$. Then $M(P[r,s]) = T[r',s']$.

We may assume that function M is uniquely defined; otherwise adjust α "infinitesimally" such that no center of P hits the pixel boundaries of T.

Definition 1. *Pattern P occurs at location $((i,j),\alpha)$ of T if the color of $P[r,s]$ is the same as the color of $M(P[r,s])$ for all pixels $P[r,s]$ of P.*

Our problem is to develop an efficient algorithm that, given T and P, finds all locations $((i,j),\alpha)$ such that P occurs in T at $((i,j),\alpha)$.

One easily observes that even if P occurs at some location of T, there can be pixels of T that are covered by P but not by any center of P. The occurrence test does not depend on the color of such a pixel of T. (The reader can convince himself of this by examining Fig. 2 with somewhat larger α). This phenomenon will slightly complicate our algorithm.

3 The Features

Let us assume that P occurs at $((i,j),\alpha)$ in the sense of Definition 1. Then it should be obvious that row i and column j of T must have around the center pixel $T[i,j]$ a sequence of colors that also occurs, at least in some approximate sense, in P. We will base our method on such sequences of length m, called *features*.

Pattern P will define a set of features and associated angles α. They are then searched for from T, to find potential locations for occurrences of P.

To read a feature for α from P, let P be on top of T, on location $((i,j),\alpha)$. Consider the pixels $T[i,j-\frac{m-1}{2}],\ldots,T[i,j+\frac{m-1}{2}]$. We denote them as $t_1,t_2,\ldots,t_m$. Let c_i be the color of the pixel of P that covers the center of t_i.

Definition 2. *The (horizontal)* feature *of P with angle α is the sequence $F(\alpha) = c_1c_2\cdots c_m$.*

Note that feature $c_1\cdots c_m$ contains colors from P, not from T. We only used the centers of pixels $t_1,\ldots,t_m$ of T as technical trick in defining $F(\alpha)$.

Fig. 2 gives an example of reading a feature. Here $m = 5$, and the high–lighted block contains the pixels $t_1,t_2,\ldots,t_5$. Their centers are shown as dots. The center of t_1 also belongs to pixel $P[4,1]$ of P. Therefore the first color c_1

of the feature is the color of $P[4,1]$. The colors of $P[4,2]$, $P[3,3]$, $P[2,4]$, and $P[2,5]$ form the rest of the feature.

If P really occurs at $((i,j),\alpha)$ in the sense of Definition 1, what is then the relation between $F(\alpha)$ and the colors of $T[i, j - \frac{m-1}{2}], \ldots, T[i, j + \frac{m-1}{2}]$? The answer is given by the following theorem.

Theorem 1. *For $k = 1, \ldots, m$, if c_k is not equal to the color of $t = T[i, j - \frac{m-1}{2} + k - 1]$, then it is equal to the color of some of the four pixels above, below, left, or right to pixel t in T.*

Proof. Assume that c_k is not equal to the color of t. Recall that c_k is the color of the pixel, say p, of P that covers the center of t. The center of p can not belong to the area of t; otherwise p and t would have the same color because P was assumed to occur at $((i,j),\alpha)$. However, the distance between the centers of p and t must be $< \sqrt{2}/2$. Hence the center of p can belong only to some pixel above, below, left, or right to t (but not to any pixel in some diagonal direction from t. Because P has an occurrence, one of these four pixels must have color c_k. □

Theorem 1 suggest the following filtration scheme for finding the occurrences of P from T.

1. Extract all possible features $F(\alpha)$ from P.
2. Find the occurrences of features $F(\alpha)$ from the rows of T. The occurrence can be approximate in the sense of Theorem 1, i.e., if a text pixel does not match, try the four pixels above, below, left, and right of it.
3. If $F(\alpha)$ occurs centered at $T[i,j]$, check if the whole P occurs at $((i,j),\alpha)$.

This needs further refinement in order to understand how the proper values of α can be found.

4 Counting the Matching Functions

The matching function M gives the correspondence between P and T that is used in the occurrence test at $((i,j),\alpha)$. The locations of the centers of the pixels of P with respect to T determine M. Consider what happens to M when angle α grows continuously, starting from $\alpha = 0$. Function M changes only at values α such that some pixel center of P hits some pixel boundary of T. Some elementary analysis shows that the set of possible angles α, $0 \le \alpha \le \pi/2$, is

$$A = \{\beta, \pi/2 - \beta \mid \beta = \arcsin \frac{h + \frac{1}{2}}{\sqrt{i^2 + j^2}} - \arcsin \frac{j}{\sqrt{i^2 + j^2}};$$
$$i = 1, 2, \ldots, \lfloor m/2 \rfloor; j = 0, 1, \ldots, \lfloor m/2 \rfloor; h = 0, 1, \ldots, \lfloor \sqrt{i^2 + j^2} \rfloor\}.$$

By symmetry, the set of possible angles α, $0 \leq \alpha \leq 2\pi$, is

$$\mathcal{A} = A \ \cup \ A + \pi/2 \ \cup \ A + \pi \ \cup \ A + 3\pi/2$$

Similarly, feature $F(\alpha)$ can change only at angles α such that some pixel boundary of P hits some center of pixels $t_1, t_2, \ldots, t_m$. The set of such angles is

$$\mathcal{B} = B \ \cup \ B \ + \pi/2 \ \cup \ B + \pi \ \cup \ B + 3\pi/2$$

where

$$B = \{\gamma, \pi/2 - \gamma \mid \gamma = \arcsin \frac{h + \frac{1}{2}}{i} \mid i = 1, 2, \ldots, \lfloor m/2 \rfloor; h = 0, 1, \ldots, i-1\}.$$

Obviously, the size of $\mathcal{A}$ is $O(m^3)$, the size of $\mathcal{B}$ is $O(m^2)$, and $\mathcal{B} \subseteq \mathcal{A}$.

Let $(\beta_1, \beta_2, \ldots)$ be the elements of $\mathcal{A}$ and $(\gamma_1, \gamma_2, \ldots)$ the elements of $\mathcal{B}$ in increasing order. By construction, $F(\alpha)$ stays unchanged for α such that $\gamma_i \leq \alpha < \gamma_{i+1}$. So $F(\alpha) = F(\gamma_i)$. However, there might be one or more angles β_k such that $\gamma_i < \beta_k < \gamma_{i+1}$ because $\mathcal{B}$ is a subset of $\mathcal{A}$. Let us denote as $K(\gamma_i)$ the set of such angles β_k. Feature $F(\gamma_i)$ represents all angles in $K(\gamma_i)$ or the corresponding functions M. If we find an occurrence of $F(\gamma_i)$, all the functions M that correspond to $K(\gamma_i)$ must be checked for an occurrence of P.

5 The Method

The method works in the following main steps.

1. Find and sort angle sets $\mathcal{A}$ and $\mathcal{B}$. This takes time $O(m^3 \log\ m)$.
2. For each α in $\mathcal{B}$, find the corresponding feature $F(\alpha)$. Associate with each $F(\alpha)$ the corresponding set of angles $K(\alpha) \subseteq \mathcal{A}$. The $O(m^2)$ different features $F(\alpha)$, each of length m, can be read from P in time $O(m^3)$. Using the sorted $\mathcal{A}$ the associated sets $K(\alpha)$ can also be found in time $O(m^3)$.
3. Build an Aho–Corasick string matching automaton [1] for features $F(\alpha)$. As the total length of the features is $O(m^3)$, the automaton can be constructed in time $O(m^3|\Sigma|)$ by standard methods.
4. Scan the rows of text T with the AC–automaton to find the occurrences of features $F(\alpha)$. The occurrences have to be found in the sense of Theorem 1, that is, if the text pixel does not match, then try the four neighboring text pixels. This can be implemented by maintaining a *set* of states of the AC–automaton that are reachable under different choices of matching pixels.
5. If the AC–automaton finds an occurrence of $F(\alpha)$ centered at (i, j) in T, check for each angle $\beta \in K(\alpha)$, whether or not P occurs at $((i, j), \beta)$.

In practise one can scan T in two directions, i.e., along the rows and the columns. A candidate location (i, j) is verified for an occurrence of P only if two features are found, centered at (i, j), and their associated angles are perpendicular. This effectively doubles the length of the features which considerably improves the filtration capability.

6 Experimental Results

The described algorithm was implemented in `C` and compiled using `gcc` version 2.8.0 on `DEC` Alpha workstation running Linux operating system. The performance was tested over different pattern sizes. Patterns were obtained from the input text in random positions and orientations. Summary of the experiments is shown in Table 1. The times reported are averages over ten runs. As expected, the preprocessing time grows fast when the feature length (the width of the pattern) increases.

feature length	preprocessing	scanning time	checking time	occurrences
3	0.003418	1.783105	0.031934	1.0
5	0.013086	1.216992	0.002637	1.0
7	0.032422	0.976758	0.002344	1.0
9	0.072168	1.013672	0.004297	1.0
11	0.132812	1.183496	0.004297	1.0
13	0.235059	1.493066	0.006836	1.0
15	0.382617	1.488770	0.007324	1.0
17	0.590625	1.617676	0.008789	1.0
19	0.855469	2.139160	0.010840	1.0
29	3.795020	3.797656	0.027246	1.0
39	11.247852	7.692480	0.074316	1.0
49	25.352930	12.933789	0.144336	1.0

Table 1. Experimental results for different feature lengths. Times included are for preprocessing, scanning the input text with AC–automaton and checking the potential occurrences. The times are given in seconds. The input text was a natural image ("Lena") of 512×512 pixels.

7 Conclusion

We have presented a filtering algorithm for exact two–dimensional pattern matching that allows rotating the pattern.

There are several directions for further study. It seems possible to remove the center–to–center assumption but the price is rather high because then the number of features becomes $O(m^5)$. If P is not very small, then it is not necessary to use features of maximal length m. A better average performance can probably be achieved with moderately short features. Generalizing the method for approximate matching is from the practical point of view the most important future goal. We are currently working on all these problems.

References

1. Alfred V. Aho and Margaret J. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, June 1975.
2. 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.
3. Ricardo Baeza-Yates and Mireille Regnier. Fast two-dimensional pattern matching. *Information Processing Letters*, 45(1):51–57, January 1993.
4. Robert S. Boyer and J. Strother Moore. A fast string searching algorithm. *Communications of the ACM*, 20(10):762–772, October 1977.
5. T. M. Caelli and Z. Q. Liu. On the minimum number of templates required for shift, rotation and size invariant pattern recognition. *Pattern Recognition*, 21:205–216, 1988.
6. 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.
7. Juha Kärkkäinen and Esko 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.
8. Marek Karpinski and Wojciech Rytter. Alphabet-independent optimal parallel search for three-dimensional patterns. Technical Report 85101-CS, University of Bonn, Department of Computer Science, November 1993.
9. Yehezkel Lamdan and Haim J. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *Second International Conference on Computer Vision: December 5–8, 1988, Innesbrook Resort, Tampa, Florida, USA*, pages 238–249. IEEE Computer Society Press, 1988.
10. G. M. Landau and U. Vishkin. Pattern matching in a digitized image. *Algorithmica*, 12(4/5):375–408, October 1994.
11. S. Ranka and T. Heywood. Two–dimensional pattern matching with k mismatches. *Pattern Recognition*, 24:31–40, 1991.
12. J. Tarhio. A sublinear algorithm for two-dimensional string matching. *PRL: Pattern Recognition Letters*, 17, 1996.
13. Rui Feng Zhu and Tadao Takaoka. A technique for two-dimensional pattern matching. *Communications of the ACM*, 32(9):1110–1120, September 1989.