Two-Dimensional Pattern Matching in Linear Time and Small Space

(EXTENDED ABSTRACT)

Maxime Crochemore[1] Leszek Gąsieniec[2] * Wojciech Plandowski[2] * Wojciech Rytter[2] *

[1] Institut Gaspard Monge, Université de Marne la Vallée, Noisy le Grand,
mac@litp.ibp.fr
[2] Instytut Informatki, Uniwersytet Warszawski, Warszawa,
{lechu,wojtekpl,rytter}@mimuw.edu.pl

Abstract. We present the first known (alphabet independent) algorithm for two-dimensional pattern matching which works in linear time and small space simultaneously. The searching phase of our algorithm works in $O(1)$ space and is followed by pattern preprocessing performed in $O(\log m)$ space. Up to now there was not known even any efficient sublinear space algorithm for this problem. The main tools in our algorithm are several 2-dimensional variations of deterministic sampling, originally used in parallel pattern matching: *small*, *frame* and *wide* samples. Another novel idea used in our algorithm is the technique of *zooming sequences*: the sequences of nonperiodic decreasing parts of the pattern (samples) of similar regular shapes. Their regularity allows to encode the *zooming* sequences in small memory (logarithmic number of bits) while nonperiodicity allows to make shifts (*kill* positions as candidates for a *match*) in a way amortizing the work. The preprocessing phase is recursive, its structure is similar to the linear time algorithm for the selection problem. The stack of the recursion consists of logarithmic number of integers. Our algorithm is rather complicated, but all known alphabet-independent linear time algorithms (even with unrestricted space) for 2d-matching are quite complicated, too.

1 Introduction

The problem of *two dimensional pattern matching* (2d-matching, in short) is to find all occurrences of a two dimensional pattern array P in a text array T. The sides of P are of length m and sides of T are of length n. The total size of T is $N = n^2$ and the total size of P is $M = m^2$. Assume $m < n$. The total size of the entire problem is equal to N. The space is measured as a number of integers with logarithmic number of bits. The pattern and the text are given as *read only* tables and by the *space* we mean additional auxiliary memory cells.

There are known several (complicated) algorithms for 1-dimensional *string-matching* in linear time and small space, see [7, 8, 12]. However 1-dimensional

* Supported by the Grant KBN 2-1190-91-01

algorithms are hardly extendible to the 2-dimensional case. Up to now there was not known any 2d-matching algorithm working in linear time and small space for all patterns. Some partial results were obtained before in [9] for a class of patterns. The pattern-matching usually consists of two quite independent parts: the preprocessing and the searching phase. The alphabet-independent linear time algorithms for searching and preprocessing phase were constructed in [2] and [11], see also [10]. Our main result is the following theorem.

Theorem 1. (Main Result)
Two-dimensional text searching can be done in $O(N)$ time and $O(1)$ space (using constant number of logarithmic bits registers), after the $O(M)$-time preprocessing performed in $O(\log M)$ space. □

Properties of 2-dimensional periodicities are theoretical basis of algorithms for 2d-matching. The periods of a 2-dimensional pattern are 2-dimensional vectors with integers components. The size of a 2d-vector $\alpha = (\alpha_1, \alpha_2)$ is $|\alpha| = \max(|\alpha_1|, |\alpha_2|)$. Only (potential periods) vectors of size at most $c \cdot m$ are considered, where $c = 1/8$. We call such vectors *short*. We also consider subpatterns which are rectangles, in this case we define the *short* vectors as vectors whose each coordinate (horizontal, vertical) has absolute value not exceeding c times the corresponding side of the rectangle. The vector α is a *period* of P iff $P[x] = P[x + \alpha]$ for each position x in P such that both sides of the equation correspond to positions in the pattern. We say that the pattern P is *periodic* iff it has a *short* period.

A *sample* is a set $\mathcal{S}$ of some positions in the pattern P. By an *$\mathcal{S}$-occurrence* of the pattern P at position x in text T (this means we place P with its left upper corner at x in T) we mean a partial occurrence of P such that all positions in $\mathcal{S}$ match corresponding positions in T. Denote by $PartialMatch(x, \mathcal{S})$ the corresponding predicate. A *full match* or an *occurrence* of P in a text T is a position x in T such that left upper corner of P is aligned with x and P match proper subarray of T completely. The full match corresponds to an *$\mathcal{S}$-occurrence* with $\mathcal{S} = P$. If $PartialMatch(x, \mathcal{S})$ then one of the rectangles, such that all positions in it, except the position x, are *killed* as candidates for *full matches*, is distinguished and called the *field of fire* of $\mathcal{S}$. The position of x relative to the field of fire $\mathcal{F}$ of $\mathcal{S}$ is called the *handle* of $\mathcal{S}$. $\mathcal{F}$ is a rectangle placed on the plane, and the handle is a distinguished position on the plane, not necessarily in $\mathcal{F}$. Assume the rectangle $\mathcal{F}$ is placed on T in such a way that the handle is placed on a position x in T. Denote by $CoveredBy(T, \mathcal{F}, x)$ all positions in T covered by $\mathcal{F}$ excluding the position x. The *field of fire* $\mathcal{F}$ of $\mathcal{S}$ satisfies for each text array T the following condition: if $PartialMatch(x, \mathcal{S})$ then there is no (full) occurrence of P in $CoveredBy(T, \mathcal{F}, x)$. Assume the pattern or its subpattern is of shape $m' \times m''$. Two basic types of *samples* considered in the paper are listed below.

Small sample $\mathcal{S}$ consists of two pattern positions (with associated symbols) and *field of fire* $\mathcal{F}$ is an $c \cdot m' \times c \cdot m''$ rectangle with the *handle* not necessarily in the middle of the *field of fire*.

Frame sample is identified by a pair of numbers (e, i). The sample is the $e \times e$ square subarray of T with the $(i-1) \times (i-1)$ central subarray removed (the removed part is called the *hole*). We call it a *frame* with boundaries (e, i), and denote it by $\Theta(e, i)$. The *thickness* of the *frame sample* is $\phi(\Theta) = e - i + 1$. The *field of fire* $\mathcal{F}$ is a square of the shape $c \cdot \phi(\Theta) \times c \cdot \phi(\Theta)$. The *handle* of *frame sample* is in the center of its field of fire $\mathcal{F}$. Such position of the *handle* is of great advantage.

The cost of (work needed to check) an $\mathcal{S}$-occurrence of the sample is its size (the total number of positions in $\mathcal{S}$). The efficiency corresponds to the area of the *field of fire*. In this sense the most efficient is the *small sample*. However it is inefficient with respect to the position of its *handle*. On the other hand *frame samples* are inefficient since the area of such a *sample* can be quadratic with respect to the area of the *field of fire*. Notice that in the 1-dimensional string-matching problem the position of the *handle* with respect to the field of fire has essentially no importance while in the 2-dimensional case it is crucial.

2 Searching for Nonperiodic 2D-Patterns

In this section we assume, to simplify the presentation, that P is a square array. However the results can be extended to the case of rectangular P. We consider first the patterns having small samples. Assume that P is nonperiodic and has a small sample $\mathcal{S}$. If the handle of $\mathcal{S}$ is not in the center of $\mathcal{F}$ then we can assume that it is in the corner of $\mathcal{F}$, by taking a suitable square part of $\mathcal{F}$ containing the *handle* in the corner. We can assume later (without loss of generality) that in the case of *small* samples the handle is one position to the right of the lower right corner of the *field of fire*. Hence the *small* sample *kills* a square of candidates positioned in the columns strictly to the left of x and rows (simultaneously) above or at the same level as x. The crucial role in the searching play special sequences of subpatterns. A *zooming sequence* is a sequence of similar *regular* objects of decreasing size. The similarity and regularity means that the next object can be computed from the previous one in a constant time and space. We explain the idea of zooming by the example of 1-dimensional matching. a given 1-dimensional pattern p is nonperiodic (a given word w is periodic iff its shortest period is $\leq \frac{\lfloor w \rfloor}{4}$). Then the prefix or the suffix of p of length $\frac{2}{3}|p|$ is nonperiodic. Denote it by $next(p)$. Then the sequence $zoomseq = (p, next(p), next(next(p)), \ldots)$ is the *zooming sequence* of nonperiodic subpatterns of p. It ends with a subpattern of a constant size. When we search for p in a 1-dimensional text t then we start to match consecutive subpatterns of the sequence $zoomseq$, extending $next^{(i+1)}(p)$ to $next^{(i)}(p)$, starting from the shortest constant size one. Whenever the whole p is found we report a *full match*. The shift, caused by some mismatch, is proportional to the last matched subpattern in the sequence $zoomseq$. Since each element of the zooming sequence, is nonperiodic all shifts amortize the total work efficiently. The sequence $zoomseq$ can be stored using logarithmic number of bits (each bit indicates whether we choose the prefix or the suffix), so constant space in the sense of small integers. This gives probably a simplest algorithm for

1-dimensional *string-matching* in linear time and constant space, see [13].

Unfortunately, this approach does not work for 2d-patterns directly. It is possible that the whole pattern P is nonperiodic but all its proper rectangular subpatterns are periodic. However, a similar approach works if rectangles are replaced by frames (rectangles with rectangular holes inside). Recall, that we defined the *frame sample* $\Theta(e,i)$ as a central $e \times e$ subarray of P with its central $i-1 \times i-1$ subarray removed and by the thickness of the *frame sample* we mean the number $\phi(\Theta) = e - i + 1$. Let $\Theta = \Theta(e,i)$ be a *frame* of thickness $t = \phi(\Theta)$. Denote by $Outer(\Theta)$, $Inner(\Theta)$ and $Center(\Theta)$ the *subframes* $\Theta(e, i + 2c \cdot t)$, $\Theta(e - 2c \cdot t, i)$, and $\Theta((e+i)/2 - c \cdot t, (e+i)/2 + c \cdot t)$, respectively. They are called *outer*, *inner* and *central* subframes of Θ. The *inner* and *outer* subframes play the same role in the defined below zooming sequence $\mathcal{Z}$ as the suffix and prefix of the 1d-pattern in the sequence *zoomseq*. Assume, that the whole pattern is its *subframe* with the *empty hole*. By a *zooming sequence* $\mathcal{Z} = ZoomSeq(P)$ of frame samples we mean the sequence of *nonperiodic* (in the sense defined below) subframes $\mathcal{Z} = (\Theta_1, \ldots, \Theta_k)$ where $\Theta_1 = P$, Θ_k has thickness $\phi(\Theta_k)$ of a constant size. And for each $1 \leq i < k$ $\Theta_{i+1} = Outer(\Theta_i)$ or $\Theta_{i+1} = Inner(\Theta_i)$.

The sequence $\mathcal{Z}$ can be stored using constant space, i.e. one integer with logarithmic number of bits. We have only to store for each $1 \leq i < k$ whether $\Theta_{i+1} = Outer(\Theta_i)$ or $\Theta_{i+1} = Inner(\Theta_i)$. The area (cost) of the frame sample of thickness $t = \phi(\Theta)$ is $O(m \cdot t)$.

The concept of a *period* can be defined for the frames in a similar way as for the rectangular text arrays, however we make a subtle difference. We say that two points $x, y \in \Theta$ are *line-connected* in Θ iff all integer points between x and y lying on the line connecting x and y are in Θ. In a frame with the empty hole each two points are line-connected. The vector α is a *period* of a frame Θ iff $P[x] = P[x + \alpha]$ for each position x in P, whenever points x, $x + \alpha$ are *line-connected* in Θ. We say that the frame Θ is periodic iff it has a *period* of size at most $c \cdot \phi(\Theta)$.

If the frame Θ is nonperiodic then it is a sample with the $c \cdot \phi(\Theta) \times c \cdot \phi(\Theta)$ *field of fire* $\mathcal{F}$ with the *handle* in the center of $\mathcal{F}$.

Lemma 2.

(1) *Assume* $Inner(\Theta)$ *is periodic and two points* $x, y \in \Theta$ *are line-connected in* $Inner(\Theta)$*, where* $x - y$ *is a short vector with respect to* Θ*. Then there are two line-connected points* $x', y' \in Center(\Theta)$ *such that* $P[x] = P[x']$, $P[y] = P[y']$ *and* $x - y = x' - y'$.

(2) *If the frame* Θ *is nonperiodic then at least one of its* Inner(Θ) *or* Outer(Θ) *subframes is nonperiodic.*

Proof.

add (1) Assume there is a small period π of the subframe $\Theta' = Inner(\Theta)$. Let L be the line which contains the pair of points x, y. Consider the case that π is not parallel to L. We can assume that if we go in direction π from L then we do not enter the *hole* of the frame Θ. Transpose the pair x, y by multiple applications of the vector π as far as it is possible inside Θ'. Then the pair x, y

is shifted to the destination points x', y'. The case that π is parallel to L can be handled analogously.

add (2) If the inner subframe is nonperiodic we are done. Otherwise, assume that both the inner and outer subframes are periodic. Hence $\Theta' = Inner(\Theta)$ has a small period π and a small vector α is a period of $\Theta'' = Outer(\Theta)$. Due to nonperiodicity of the whole frame Θ there should be a pair (x, y) of line-connected positions inside Θ which "kills" this vector: i.e. $x - y = \alpha$ and $P[x] \neq P[y]$. It could happen that $x, y \in \Theta''$, however in this case x, y should be not line-connected in Θ'', and this implies they are close to the internal border of Θ'' and they are in Θ' due to large overlap of both subframes. Hence in each case we can assume that x, y are in the inner subframe. Due to case (1) the positions x and y can be shifted to some other positions $(x', y') \in Center(\Theta)$ such that $P[x'] = P[x]$ and $P[y'] = P[y]$. Notice that $Center(\Theta) \subseteq \Theta''$. The pair (x', y') breaks periodicity α in the outer frame if both points are *line-connected* in the outer frame. The crucial point is that each two positions inside $Center(\Theta)$ are always *line-connected* with respect to Θ'', since all points in $Center(\Theta)$ are far enough from the internal border (hole) of $Outer(\Theta)$. Hence $x', x' + \alpha \in \Theta''$, x' and $x'+\alpha$ are *line-connected* in $Outer(\Theta)$ and $P[x'] \neq P[x'+\alpha]$. This contradicts our assumption that α is a period of $Outer(\Theta)$. The contradiction proves that the inner and outer subframes cannot be both periodic, so one of them is nonperiodic. This completes the proof.

As a consequence of Lemma 2 we know that for each nonperiodic 2d-pattern P there exists a zooming sequence of frame samples which can be stored in $O(\log m)$ bits (or using one small integer). Assume that we have already computed:

a *small sample* $\mathcal{S}$ and a *zooming sequence* $\mathcal{Z}$ of *frame samples* for P.

Recall, that $PartialMatch(x, \Psi)$ is the boolean predicate which is satisfied iff the text array T agrees with the subset Ψ of positions of the pattern placed at x. We have different types of samples and Ψ we mean any of them. Recall that samples differ with respect to the field of fire. Divide the text array T into disjoint *windows*. Each *window* is a $\frac{c}{2} \cdot m \times \frac{c}{2} \cdot m$ square subarray of T. Due to nonperiodicity of P there is at most one occurrence of P in one *window*. It is enough to show how to find an occurrence (at most one) of the pattern P in a fixed *window* $\mathcal{W}$. We assume (without loss of generality) that the *handle* of the *field of fire* $\mathcal{F}$ of the small sample $\mathcal{S}$ is at the bottom-right corner of $\mathcal{F}$.

Remark. Assume P is nonperiodic and $x \in \mathcal{W}$. If $PartialMatch(x, \mathcal{S})$ then all points in columns to the left of x and (simultaneously) in rows above x can be eliminated (we know they are not full occurrences).

We construct an algorithm which scans a given window $\mathcal{W}$ and searches for the pattern. In every moment the window $\mathcal{W}$ is partitioned into the area of *"nonactive"* positions and the area $\mathcal{A}$ of *"active"* positions. We know (due to information we have at the moment) that there is no occurrence at any "nonactive" position, all *"active"* positions are candidates for a full match. Initially all positions of the window are *active* and $\mathcal{A} = \mathcal{W}$. We have only small memory to categorize positions. The set $\mathcal{A}$ should be simple enough to remember it easily. Let $x \in \mathcal{W}$ and $r \geq 0$, then $\Gamma(x,r)$ denotes the area consisting of all points $z \in \mathcal{W}$ which are in the same column as x or to the right of this column and in the rows not below x and which satisfy $dist(x,z) > r$, where $dist((x_1,x_2),(y_1,y_2)) = \max\{|x_1 - y_1|, |x_2 - y_2|\}$. If a *field of fire* is a $f \times f$ square then by its *radius* we mean the number $\max\{\lfloor \frac{f}{2} \rfloor, 1\}$. $\Gamma(x,r)$ has the following property. Assume that we know there is no full match below the row containing x and we found at position x the partial match (sample) $\mathcal{S}$ and the frame sample Θ_i whose *field of fire* has *radius* r. Then we know that the set of active (possible candidates for a full match) positions is contained in $\Gamma(x,r)$.

The algorithm scans (inspects) positions of $\mathcal{W}$ for an occurrence of P in a systematic way, due to limited memory the scanning order is quite important. Define the *linear order* $\prec$ of points of $\mathcal{A}$ as follows. The points of $\mathcal{W}$ are ordered from bottom to upper rows and right-to-left in each row. Hence the first point in $\mathcal{W}$ is its bottom-right corner. Denote by $Next(x,\mathcal{A})$ the point following x in $\mathcal{A}$ according to the order defined above. If there is no such point or $\mathcal{A} = \emptyset$ then $Next(x,\mathcal{A}) = nil$.

Algorithm *Sampling-Zooming*
{searching for pattern P in a given window $\mathcal{W}$,
P is nonperiodic and has a small sample $\mathcal{S}$ }

1: $\mathcal{A} = \mathcal{W}$; $x :=$ bottom-right corner of $\mathcal{W}$;
2: **while** $x \neq nil$ **do**
  begin {main iteration}
    if $PartialMatch(x,\mathcal{S})$ **then**
    begin
      $j := \min\{i \geq 1 \; : \; PartialMatch(x, \Theta_i\}$;
      if $j = 1$ **then**
        report *full match* at x and STOP;
      $r :=$ the *radius* of *field of fire* of Θ_j;
      $\mathcal{A} := \Gamma(x,r)$; $x := Next(x,\mathcal{A})$;
    end ;
  end {main iteration}

Lemma 3. *The algorithm Sampling-Zooming finds an occurrence of P, if there is any, in $O(N)$ time and $O(1)$ space.*

Proof. Note first, that since the size of $\mathcal{S}$ is constant, the total cost of all executed operations $PartialMatch(\mathcal{S})$ is linear with respect to the size of the window. The most expensive part is the computation of the values of j, see algorithm *Sampling-Zooming*. It can be realized by testing partial match at x

of $\Theta_k, \Theta_{k-1}, \ldots, \Theta_j$ successively. The zooming sequence is kept in (and recovered from) a logarithmic sequence of bits, and can be stored in a single integer. Hence the space needed to compute j is constant. The successive partial matches for Θ_i are tested *naively*. Their sizes grow geometrically and their total cost is proportional to the size of Θ_j which is $O(\phi(\Theta_j) \cdot m)$. In the same time $\phi(\Theta_j)$ is proportional to the *radius* r of the *field of fire* of Θ_j. Notice that the value $j = \min\{i \geq 1 : PartialMatch(x, \Theta_i)\}$ can be computed in constant space and $O(r \cdot m)$ time, where r is the *radius* of the field of fire of Θ_j. In the course of the algorithm there is processed a sequence $x_1, x_2, \ldots, x_p$ and $r_1, r_2, \ldots, r_p$ of values of x and r, respectively. Hence, the total time of the algorithm is proportional to $\sum_{i=1}^{p} r_i \cdot m$. Notice that each point x_i is right and up to the previous one and $dist(x_i, x_{i+1}) \geq r_i$. Hence the sum of all r_i's is at most m. This implies that the sum above is bounded by m^2, in other words it is linear with respect to the total size of the pattern. There are $O(\frac{n^2}{m^2})$ windows $\mathcal{W}$ and they can be processed systematically in time $O(|\mathcal{W}|)$ each. Hence total time is linear with respect to the total size of the text array T.

We consider now general nonperiodc patterns, so we drop the assumption that P has a small sample.

Denote by *truncate*(P) the subarray of P of shape $(1-c) \cdot m \times m$ which consists of the bottom $(1-c) \cdot m$ rows of P. *Wide samples* are nonperiodic subpatterns resulting from P by cutting off some consecutive top rows. They are *wide* in the sense that they have m columns.

Lemma 4. *Assume P is nonperiodic and truncate(P) is periodic. Then P has a small sample $\mathcal{S}$ which consists of two positions.*

We construct the *zooming sequence* $\mathcal{Z}$ of *wide* samples as the sequence of nonperiodic subpatterns $P = P_0, P_1, \ldots, P_k$, where $P_{i+1} = truncate(P_i)$, for $1 \leq i < j$, and P_k has a small sample. We take P_k such that $truncate(P_j)$ is periodic or P_k is of a constant size.

We show how to search in a *window* $\mathcal{W}$ of the shape $\frac{c}{2} \cdot m \times \frac{c}{2} \cdot m$. The crucial point is that wide samples P_i are *wide* enough to guarantee that their field of fire has a width (number of columns) equal to $\frac{c}{2} \cdot m$. This implies that the *active area* (set of candidates for a *match*) has now much simpler shape and consists of a small number of topmost consecutive rows of $\mathcal{W}$. We use the linear ordering $\prec$ of points (right-left, bottom-up) in the window $\mathcal{W}$ defined in the preceding section. Let $FindNext(x, P', \mathcal{W})$ be the function which returns the first occurrence of some subpattern P', for $P' \in \{P = P_0, P_1, \ldots, P_k\}$ in $\mathcal{W}$ following a given point x (including x) with respect to the defined linear order $\prec$. The returned value is *nil* if there is no occurrence. Denote by $ShiftUp(x,\ s,\ \mathcal{W})$ the function which returns the first point in the row which is s rows higher than x in $\mathcal{W}$. If there is no such point (we go outside $\mathcal{W}$) then the returned value is *nil*. The work performed in one iteration is proportional to the *shift* (the total size of *skipped* rows) in the operation $ShiftUp$. Hence the time in a given window is proportional to its size. Finally we have shown that searching for nonperiodic patterns can be done in linear time and constant space, in case it is followed by pattern preprocessing.

Algorithm *Searching_General_Nonperiodic*
$x :=$ bottom-right corner of $\mathcal{W}$;
while $x \neq nil$ **do**
begin {main iteration}
compute $x := FindNext(x, P_k, \mathcal{W})$ using
algorithm *Sampling-Zooming*;
$j := \min\{i \geq 1 \,:\, PartialMatch(x, P_i)\}$;
if $j = 1$ **then** report *full match* at x and STOP;
$h :=$ the *height* of P_j ; $x := ShiftUp(x,\ \frac{c \cdot h}{2},\ \mathcal{W})$;
end {main iteration}

3 The Searching for Periodic 2D-Patterns

We have to deal with three categories of 2-dimensional periodic patterns: *lattice-periodic*, *radiant-periodic* and *line-periodic* patterns, see [1, 11].

Lemma 5. *Text searching for periodic 2d-patterns can be done in $O(N)$ time and $O(1)$ space.*

Proof. We present the proof only for the line-periodic case, which is the hardest case. We can reduce this case to other (simpler) cases: nonperiodic, lattice-periodic and radiant-periodic.

Assume the pattern is line-periodic. Let π be the *shortest* period of P, its direction is the ratio $\alpha(\pi) = \frac{x}{y}$. Assume $|\alpha(\pi)| \geq 1$. A *horizontal stripe* of height h is an $h \times m$ subrectangle of P. Let $P1, P2$ be the upper and the lower stripes of height $h = 3|\pi|$. It is not difficult to prove that $P1, P2$ are *together nonperiodic*. This means that every nonzero vector $v = (x, y)$ such that $|x| \leq c \cdot h$ and $|y| \leq c \cdot m$ and which is not a period of P is *killed* by one of $P1, P2$. Denote by D the pair $P1, P2$. It can be treated as a *double pattern*. D can be also treated as a *sample* of the whole pattern P. In this sense it has the rectangular *field of fire* of shape $c \cdot h \times c \cdot m$ and the *handle* in the center. An algorithm for searching for such *together nonperiodic* double patterns D is essentially the same as the one for searching for nonperiodic ordinary patterns.

The algorithm for searching line-periodic patterns goes as follows. Divide text T into $c \cdot \frac{m}{2} \times c \cdot \frac{m}{2}$-size windows. Now every window is divided in $c \cdot \frac{h}{2} \times c \cdot \frac{m}{2}$-size subwindows. Since the field of fire of D is a rectangle of shape $c \cdot h \times c \cdot m$ there is at most one occurrence of D in one subwindow. Starting from the lowest subwindow of the window we search for D. Suppose we have found an occurrence of D in i-th subwindow starting from the bottom of the window. If there was an occurrence of the pattern in the window then we check if the direction of the line which contains the position of the last occurrence of the pattern and the current position is the direction of periods of P. If it is not, we continue searching for D starting from the $(i+1)$-th subwindow. Otherwise, we find the most thick possible outer frame of P which matches the text at the found position. We start by checking naively if the frame of thickness h matches the text T. If it does not match then we continue searching for D starting from the $(i+1)$-th subwindow. If the frame matches properly then we find the most thick possible matching frame of P avoiding comparisons of symbols belonging to the last occurrence of P. If there was any occurrence of P before then we know that the symbols from the last occurrence of P match. Assume the thickness of this frame is r. If tested position is not the occurrence of the pattern we omit $t = max(0, \frac{r-h}{\pi})$ subwindows and continue searching for D from the $(i+t+1)$-th subwindow. If the position is the occurrence of P we report the occurrence and store the position as the last found occurrence of the pattern. The linear time complexity of the algorithm follows from two facts. The first one is that comparisons involved in searching of D occurrences (but not P) are amortized by proportional shifts over subwindows. The second one is that each symbol of the text is tested at most constant number of times after we enter the phase when full occurrences of the pattern P are checked. We omit the details.

As a direct implication we get the following theorem.

Theorem 6. *The searching for 2d-patterns, followed by the pattern preprocessing, can be performed in linear time and constant space.* □

4 Preprocessing of Two-Dimensional Patterns

The crucial operation in the preprocessing is the computation of the shortest period in a given direction α, to simplify its computation the notion of the 2d-period is slightly redefined. For a given vector π define $LINES(\pi)$ as the set of lines in the pattern which are parallel to π. For example set $LINES(1,1)$ consists of all lines parallel to the one of the diagonals. A line $\mathcal{L}$ is *long with respect to* π iff its length is at least $c' \cdot |\pi|$, for some fixed constant $c' > 0$. , that a vector is *short* iff its size is at most $c \cdot m$. A vector π is a *weak period* of P iff $|\pi|$ is a period of each line $\mathcal{L} \in LINES(\pi)$ and is long with respect to π. The pattern is *strongly nonperiodic* (*weakly periodic*) iff it has no (it has a) *short* weak period. Each strongly nonperiodic pattern is nonperiodic in the standard periodicity meaning. We say, that a vector π is in a direction α iff it is parallel to π. Denote by $LinePeriod(\alpha, P)$ the shortest period in direction α in P. Define analogously $LinePeriod(\alpha, \Theta)$ for a frame Θ. In later presentation we replace standard *periods* by *weak periods* in the definition of samples and zooming sequences used in the searching phase. Define $m \times m$ square to be radiant periodic if after removing $2c \cdot m \times 2c \cdot m$ opposite corner squares it is a lattice. Then the searching phase works essentially in the same way as before.

Remark. Assume the pattern P is weakly periodic in a direction α. Then all weak periods of P in the direction α are multiples of $LinePeriod(\alpha, P)$.

We omit later the words *weak* and *strong* and assume that periodicity means *weak* periodicity. We can compute periods of each line in a given direction in the decreasing order with respect to the lengths of these lines, using 1-dimensional string-matching on each line.

Lemma 7. *Vectors $LinePeriod(\alpha, P)$ and $LinePeriod(\alpha, \Theta)$ can be computed in linear time and constant space, for a given direction α,* $\square$

The *preprocessing data* for nonperiodic patterns consists of the following objects:
(1) A sequence $\mathcal{T} = P_1, \cdots, P_k$ of *wide* samples;
(2) A *small sample* $\mathcal{S}$ of P_k;
(3) A *zooming sequence* $\mathcal{Z}$ of *frame samples* for P_k.
If P is periodic then the preprocessing consists of computing the *periodicity* of P. This means that we have to recognize pattern periodicity type among line-periodic, radiant-periodic and lattice-periodic ones. Moreover, if P is line-periodic then we compute the shortest weak period of the pattern and if it is radiant periodic or lattice periodic we compute a pair of basis vectors defining the lattice inside the pattern, see [1].

Denote by P_{center} the central subsquare of the shape $c'' \cdot m \times c'' \cdot m$ of the pattern P where $c'' = 4c$. In the case of *frames* with nonempty holes we define P_{center} as the central subarray of the top rectangle of the frame. In the discussion below we assume that we deal with subpatterns without *holes*, the frame case can be considered similarly.

Lemma 8. *$T_{center}(M) = O(M)$, where $T_{center}(M)$ is the time to compute the periodicity of the pattern P of total size M in constant space under the assumption that the P_{center} has been preprocessed.*

Proof. Let $\mathcal{W}$ be a window in P of shape $2c \cdot m \times 2c \cdot m$ whose center is left upper corner of the square P_{center} in P. Notice that if π is a short weak period of the pattern P and x is a position of the left upper corner of P_{center} in P then the square P_{center} occurs in P at position $x + \pi$ inside $\mathcal{W}$ and π is a period (not necessarily short) of P_{center}. We use the linear time algorithm from the previous section to search P_{center} inside the pattern P in the window $\mathcal{W}$ (this can be done since P_{center} has been already processed). If there is no occurrence of P_{center} in $\mathcal{W}$ then P is nonperiodic. If all occurrences lie on one line then the pattern is either nonperiodic or line periodic and the direction of the periods is the direction of the line. In that case, by Lemma 7, the periodicity of P can be computed in linear time and constant space. If not all occurrences of the square lie on one line then the square P_{center} has two (not necessarily short with respect to P_{center}) periods in different directions and therefore the central part of it is a part of a lattice. Starting from the center of P (which is also the center of P_{center}) we check how far this lattice can be extended in P. If it extends to the whole pattern then the pattern is lattice periodic. Let a *defect* be a point which "breaks" lattice periodicity of the center of P_{center}. If there are no defects outside opposite $2c \cdot m \times 2c \cdot m$ square corners of the pattern then the pattern is radiant periodic.

Otherwise, the pattern is either line periodic in a direction α or nonperiodic. We describe briefly how to find a potential direction α. Once we find it, we use Lemma 7. Consider a quadrant of P containing two defects one of them lying outside its corner square, if there is no such quadrant then P is nonperiodic. Assume it is the left upper quadrant of P. Consider its $\frac{m}{2} \times \frac{m}{2}$ frame Θ of thickness $c \cdot m$. Denote by A, B left upper and right bottom $c \cdot m \times c \cdot m$ corner squares of the quadrant. Denote by C, D connected parts of Θ that result from Θ after removing parts A, B. If there are no defects inside Θ then P is nonperiodic.

If any defect from outside of A lies in C then we divide Θ into two parts, C with B, and D with A, otherwise we divide it in parts, C with A, and D with B. Denote these parts by C', D'. If one of them contains no defects then P is nonperiodic. Otherwise, we say that a line $\mathcal{L}$ is a *defect line* (*d_line* for short) iff it contains defects in C' and D'. Denote by D_LINES the set of all *d_line*s. Notice that there is exactly one line $\mathcal{L} \in D\_LINES$ such that all defects in Θ lie on the same side of $\mathcal{L}$ (as the left upper corner of Θ). Denote it by $d\_line(\Theta)$. This line is a tangent to the convex hulls of defects from each part of Θ.

Claim 9 *If the pattern is line periodic in a direction α then α is parallel to $d\_line(\Theta)$.*

Proof. Denote by P_LINES the set of lines crossing Θ and containing defects whose direction is the direction of periods. If $\mathcal{L} \in P\_LINES$ then there are defects on $\mathcal{L}$ which are inside Θ. Let $\mathcal{L}$ be the closest line to the center of the pattern among lines in P_LINES. Then $\mathcal{L}$ is a *d_line* and $\mathcal{L} = d\_line(\Theta)$. This completes the proof of the claim.

A line $d\_line(\Theta)$ can be found in $O(M)$ time and constant space by an algorithm which finds its successive approximations. We omit the details. Having the direction we check if the pattern is line periodic in this direction, applying algorithm used in Lemma 7. This completes the proof.

Theorem 10. *The pattern preprocessing can be done in linear time and logarithmic space.*

Proof. In the preprocessing we have first to find a *nonperiodic part* P' of P. If P is a nonperiodic pattern or a frame then *nonperiodic part* means $truncate(P)$ or nonperiodic subframe (if P is a frame). If P is periodic then nonperiodic part consists of two nonperiodic corners (in radiant-periodic case), central nonperiodic part (in lattice-periodic case) or a double pattern D which contains two stripes (in line-periodic case). The computation has a recursive structure and informally can be described as follows.

- Preprocess recursively the central subpattern P_{center};
- Compute the *periodicity* of P in time $O(T_{center}(M))$;

– Find the *nonperiodic part* P' of P;
– Preprocess recursively P'.

The total size of P' is bounded by $c'' \cdot M$ and the total size of P_{center} is bounded by $c' \cdot M$. Hence the time complexity $T_{pre}(M)$ to preprocess an object of total size M satisfies the following inequality:

$$T_{pre}(M) \leq T_{pre}(c' \cdot M) + T_{pre}(c'' \cdot M) + T_{center}(M) + O(M);$$

We choose c', c'' such that $c' + c'' < 1$, $T_{center}(M) = O(M)$. This implies that $T_{pre}(M) = O(M)$. The stack of the recursion is of logarithmic size, so the additional memory is $O(\log m)$. The details are omitted.

References

1. A. Amir and G. Benson, Two dimensional periodicity and its applications, *Proc. 3rd ACM-SIAM SODA'92*, p.440-452.
2. A. Amir, G. Benson and M. Farach, Alphabet independent two dimensional matching, *Proc. 24th ACM STOC'92*, p.59-68.
3. A. Amir, G. Benson and M.Farach, Optimal parallel two dimensional pattern matching, *Proc. 5th ACM SPAA'93*, p.79-85.
4. T. J. Baker, A technique for extending rapid exact-match string matching to arrays of more than one dimension, *SIAM J. Comp.* 7 (1978), p.533-541.
5. R. S. Bird, Two dimensional pattern matching, *Inf. Proc. Letters* 6, (1977) p.168-170.
6. R. Cole, M. Crochemore, Z. Galil, L. Gąsieniec, R. Hariharan, S. Muthukrishnan, K. Park and W. Rytter, Optimally fast parallel algorithms for preprocessing and pattern matching in one and two dimensions, *Proc. 34th IEEE FOCS'93*, p.248-258.
7. M. Crochemore, String matching for ordered alphabets, *TCS* 92 (1992), p.225-251.
8. M. Crochemore and D. Perrin, Two-way string matching, *JACM* 38:3 (1991), p.651-675.
9. M. Crochemore, L. Gąsieniec and W. Rytter, Two-dimensional pattern matching by sampling, *IPL* 46 (1993), p.159-162.
10. M. Crochemore and W. Rytter, A version of Galil-Park two-dimensional matching, *Report IGM* 94-05, Universite Marne la Valle (1994).
11. Z. Galil and K. Park, Truly alphabet independent two dimensional matching, *Proc. 33th IEEE FOCS'92*, (1992) p.247-256.
12. Z. Galil and J. Seiferas, Time-space optimal string matching, *J. Comp. Syst. Sc.* 26 (1983), p.280-294.
13. L. Gąsieniec, W. Plandowski and W. Rytter, The zooming method: a recursive approach to time-space efficient string-matching, *TCS*, to appear.
14. U. Vishkin, Deterministic sampling - a new technique for fast pattern matching, *SIAM J. Comput.* 20 (1991), p. 22-40.