Validation and Decomposition of Partially Occluded Images with Holes

Julien Allali[1], Pavlos Antoniou[2], Costas S. Iliopoulos[2], Pascal Ferraro[1], and Manal Mohamed[2]

[1] LaBRI, University of Bordeaux I, UMR5800, 33405 Talence, France
[2] Dept. of Computer Science, King's College London, London WC2R 2LS, England, UK

Abstract. A partially occluded image consists of a set of objects where some may be partially occluded by others. Validating occluded images distinguishes whether a given image can be covered by the members of a finite set of objects, where both the image and the object range over identical alphabet. The algorithm presented here validates a one-dimensional image x of length n, over a given set of objects all of equal length and each composed of two parts separated by a transparent hole.

Keywords: valid image, approximate valid image, image decomposition

1 Introduction

In recent studies of repetitive structures of strings, generalized notions of periods have been introduced [2]. Here we present practical methods to study the following type of regularity: we want to "*cover*" a string using a set of "*objects*".These objects may "*occlude*" each other and may be separated by a hole.

Validating partially occluded images is a classical problem in computer vision and its computational complexity is exponential. An input image is valid, if it can be composed from a members of a finite set of objects, with some of the appearing objects being partially occluded by other ones. This problem is also typical in pattern recognition and computer graphics. There is a great number of artificial intelligence and neural net solutions to this problem.

Validating occluded one dimensional images has been a well studied problem in algorithm design. Iliopoulos and Simpson [6] focused on the theoretical aspect of the problem and produced a sequential on-line algorithm for validating occluded one-dimensional images. Furthermore, different aspects of this problem have been studied and solved by Iliopoulos and Reid. In [5], the authors provided a linear time solution to the problem in the presence of errors, in [4] they presented an optimal $\mathcal{O}(\log \log n)$-time algorithm using parallel computation and in [3] solved the problem for discrete two-dimensional partially occluded images in linear time.

In this paper, we move a step forward, based on the above analyses and we extend the previous work by considering the validity of a family of images, that we call *valid images with holes*. In this context, given a set of objects $s_1, \ldots, s_k$, each composed of two parts separated by a small transparent hole, an image x of length n is a valid image with hole, if x is iteratively obtained from a string $z = \#^n$ by substituting substrings of z by some objects s_i , for some $i \in \{1..k\}$ and a special "background" symbol #. We focus on designing an on-line algorithm for testing images in one dimension for validity, with restricted set of objects, e.g., objects of the same length, that are consisting of two parts separated by a hole of small size.

Julien Allali, Pavlos Antoniou, Costas S. Iliopoulos, Pascal Ferraro, Manal Mohamed: Validation and Decomposition of Partially Occluded Images with Holes, pp. 129–136.
Proceedings of PSC 2009, Jan Holub and Jan Žďárek (Eds.), ISBN 978-80-01-04403-2 © Czech Technical University in Prague, Czech Republic

The paper is organized as follows. In Section 2, we introduce basic definitions and notations used in this paper. In Section 3, we present the principles for validating images in one dimension. In Section 4, the main validating algorithm is presented with its time complexity analysis. Finally, we state our conclusions in Section 5.

2 Background

An *alphabet* Σ is a set of elements that are called letters, characters or symbols. A *string* x is a sequence of zero or more letters from Σ, that is $x[1]x[2]\cdots x[n]$ with $x[i] \in \Sigma$, $1 \leq i \leq n$. The *length* of x, denoted by $|x|$, is the total number of letters in x. The string of length zero is the empty string ε. The string xy is a *concatenation* of two strings x and y.

A string y is a *substring* of x, if and only if, there exist two strings u and v such that $x = uyv$. A string u is a *prefix* (respectively *suffix*) of x, if and only if, there exists a string v over such that $x = uv$ (respectively $x = vu$). If $v \neq \varepsilon$ then u is a *proper prefix* (respectively *proper suffix*) of x.

Additionally, $prefix_p(x)$ denotes the first p letters of x and $suffix_p(x)$ denotes the last p letters of x. Given two strings $x = x[1]x[2]\cdots x[n]$ and $y = y[1]y[2]\cdots y[m]$, such that $x[n-i+1]\cdots x[n] = y[1]\cdots y[i]$ for some $i \geq 1$ (that is such that x has a suffix equal to a prefix of y), the string $x[1]\ \cdots\ x[n]y[i+1]\cdots y[m]$ is called a *superposition* of x and y with i overlap. A string w of x is called *a cover* of x if and only if an extension of x can be constructed by concatenations and superposition of w.

Valid Image over set of Objects:

Definition 1. Let x be a string of length n over an alphabet Σ and let the dictionary $\mathcal{O} = \{s_1, \ldots, s_m\}$ be a set of strings called the *objects* also over Σ. Then x is called a *valid image* if and only if $x = z_i$ for some $i \geq 0$, where

$$\begin{aligned} z_0 &= \#^n \\ z_{i+1} &= \text{prefix}_p(z_i)\ s_l\ \text{suffix}_q(z_i)\,. \end{aligned} \tag{1}$$

for some $s_l \in \mathcal{O}$ and $p, q \in \{0, \ldots, n-1\}$ such that $p + |s_m| + q = n$. □

Equation (1) is called the *substitution rule* and the sequence $z_0, z_1, \ldots, z_i$ is called the *generating sequence* of x. The number of distinct generating sequences was proved to be exponential [6].

An example of such generating sequences for a specific string is as follows. Let $\mathcal{O} = \{s_1 = abc, s_2 = acde, s_3 = ade, s_4 = dc, s_5 = abd\}$. Then $x = abababacdedcdcade$ is a valid image over $\mathcal{O}$ with generating sequence:

$$\begin{aligned} z_0 &= \#^{17}, \\ z_1 &= \underline{abc}\#^{14}, \\ z_2 &= abc\#^{11}\underline{ade}, \\ z_3 &= ab\underline{abc}\#^{9}ade, \\ z_4 &= abab\underline{abc}\#^{7}ade, \\ z_5 &= abababa\underline{cde}\#^{4}ade, \end{aligned}$$

$$z_6 = abababacde\underline{dc}\#^2ade,$$
$$z_7 = abababacdedc\underline{dc}ade.$$

Note that the generating sequence of x is not unique. The following sequence:

$$z_0 = \#^{17},$$
$$z_1 = \underline{abd}\#^{14},$$
$$z_2 = ab\underline{abc}\#^{12},$$
$$z_3 = abab\underline{abc}\#^{10},$$
$$z_4 = abababc\#^7\underline{ade},$$
$$z_5 = abababc\#^3\underline{dc}\#^2ade,$$
$$z_6 = abababc\#^3dc\underline{dc}ade,$$
$$z_7 = abababacdedcdcade.$$

also generates x as a valid image over $\mathcal{O}$.

Valid Image over Set of Objects with Hole:
Let x be a string of length n over an alphabet Σ and let the dictionary $\mathcal{O} = \{s_1, \ldots, s_k\}$ be a set of strings called the objects*, where each object s_i is composed of two strings s_i^l and s_i^r* separated *by a hole of length h. Then x is called a* valid image *if and only if $x = z_i$ for some $0 \leq i$, where*

$$\begin{aligned} z_0 &= \#^n \\ z_{i+1} &= \text{prefix}_p(z_i)\ s_m\ \text{suffix}_q(z_i)\,. \end{aligned} \tag{2}$$

for some $s_m \in \mathcal{O}$ and $p, q \in \{0, \ldots, n-1\}$ such that $p + |s_m| + q = n$.

Figure 1, presents the notion of finding the objects comprising an image. If the image is observed from above, one can see some of the objects are partially occluded by others but can see some of the covered ones *through* the hole. We are trying to decompose what the eye sees to its sources. In this example of Figure 1 the valid image of the objects is composed of the following elements:
Image = *prefix*(s_2^l) s_1^l *suffix*(s_3^l) *substring*(s_4^l) *prefix*(s_2^r) s_1^r *suffix*(s_3^r) *suffix*(s_4^r)

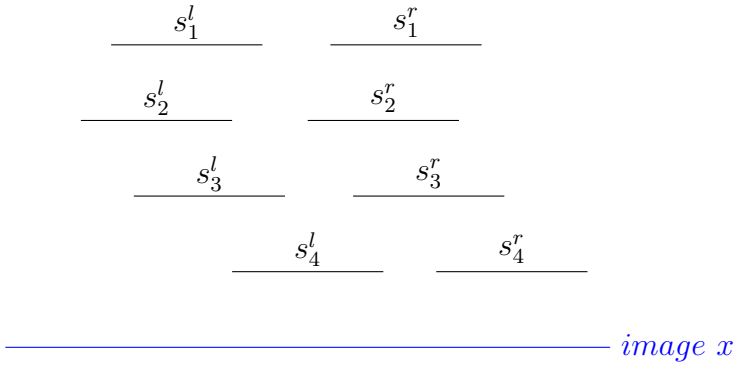

Figure 1. Image consisting from objects separated by a hole of same length.

In this paper, we consider the problem of validation of an image over a set of objects with holes. Each object $s_i \in \mathcal{O}$ consists of a *left part* (*head*) and a *right part*(*tail*) separated by a transparent hole of length h. We denote the left part of s_i as s_i^l and the the right part as s_i^r. For simplicity, we require that $|s_i^l| = |s_i^r|$ and $h \ll |s_i^l|$, for each $s_i \in \mathcal{O}$.

The definition of a valid image implies that constituent objects are contained within the image x. That is, there is no s_i for all $i \in \{1, \ldots, k\}$ that is 'cut' at $x[1]$ or $x[n]$.

This leads to the following facts:

If x is a valid image over $\mathcal{O} = \{s_1, s_2, \ldots, s_k\}$, then for some $i \in \{1, \ldots, k\}$,

Fact 1: there exists a suffix $\bar{s}_i^r$ of s_i^r that is also a suffix of x.

Fact 2: there exists a prefix $\hat{s}_i^l$ of s_i^l that is also a prefix of x.

Fact 3: there is no suffix of a left part s_i^l that occurs in x ending at position ℓ, where $\ell > n - h - |s_i^r|$.

Fact 4: there is no prefix of a right part s_i^r that occurs in x at position ℓ', where $\ell' < |s_i^l| + h$.

3 Validation of Images with Objects of Equal Length

In this section, we start by defining what part of a valid image one should expect to see within the hole *i.e.* between the left and the right parts of an object. Subsequently, we proceed and present the main mechanism for validating one-dimensional images over a set of objects with holes.

Given a set of objects $\mathcal{O}$, a string b of length h is a *binding* if it is a concatenation of the following three (possibly empty) parts:

Part 1: is a sequence of suffixes of left/right parts of objects in $\mathcal{O}$,
where the leading (first) suffix is a suffix of a left part of an object.

Part 2: is a substring of a left/right part of an object.

Part 3: is a sequence of prefixes of left/right parts of objects in $\mathcal{O}$,
where the leading (last) prefix is a prefix of a right part of an object.

Note that any substring of a left or a right part of an object is also a binding if it is of length h. A binding b is *satisfied*, if and only if, the length of the part of the valid image following the binding is big enough to insure that each object from $\mathcal{O}$ appears within the hole is totally occluded by the image.

Theorem 2. *Let x be a string over Σ. Let $\mathcal{O} = \{s_1, s_2, \ldots, s_k\}$ be a set of objects all of the same length and each composed of left part s_i^l and right part s_i^r separated by a hole of length h. The string x is a valid image over $\mathcal{O}$ if and only if*

$$x = \hat{s}_i^l y \qquad with \qquad i \in \{1..k\}, \tag{3}$$

or

$$x = y\bar{s}_i^r \qquad with \qquad i \in \{1..k\}, \tag{4}$$

or

$$x = y\widetilde{s}_i w \qquad with \qquad i \in \{1..k\}, \tag{5}$$

or

$$x = y\bar{s}_i^l b\hat{s}_i^r z \qquad with \qquad i \in \{1..k\}, \tag{6}$$

where $\hat{s}_i^l$, $\bar{s}_i^r$ and $\widetilde{s}_i$ denote a prefix of the left part s_i^l, suffix of the right part s_i^r and a substring of either parts of s_i respectively, y and w are valid images and b is a satisfied binding.

The above theorem provides the main mechanism for validating images over a set of objects with holes and all of equal length. Equations (3) and (4) are restatements of Facts 1 and 2. Equations (5) and (6) state what one should expect at the decomposition of two valid sub-images.

If an image x is of the form (5), and $s_i = u\widetilde{s}_i v$ for some strings u, v and a non-empty substring $\widetilde{s}_i$ of either the left or the right part of s_i, then x is a valid image, since x can be generated by the sequence:

$z_0 = \#^n$, $z_0 = \#^p s_i \#^q$, where $p = |y| - |u|$,
followed by an application of the generating sequence of y on the first $|y|$ symbols of z_1 and the generating sequence of w on the last $|w|$ symbols of z_1.

If an image x is of the form (6), and s_i^l, and s_i^r are both the left and the right part of s_i separated by a hole of length h, then x is a valid image, since x can be generated as:

$z_{i+1} = prefix(z_i) s_i^l b s_i^r suffix(z_i)$,
where b is the part of z_i appearing in the hole separating the left and the right part of s_i, followed by an application of the generating sequence of y on the first $|y| = |prefix(z_i)| + |s_i^l| - |\bar{s}_i^l|$ symbols of z_1 and the generating sequence of w on the last $|w|$ symbols of z_1.

4 An On-Line Algorithm

Here we present the algorithm for validating an image over a set of objects with holes and of equal length. Algorithm 1 presents the main commands of the algorithm in the form of pseudocode.

Algorithm 1 On-line Image Validation ALgorithm

Input: image $x[1\ldots,n]$, the set of objects $\mathcal{O} = \{s_1, s_2 \ldots s_k\}$ all of equal length.
Output: `T` if and only if x is a valid image, `F` otherwise.

initialization
`valid`$[0,\ldots,n] \leftarrow [\mathtt{T},\mathtt{F},\ldots,\mathtt{F}]$
`p_valid`$[0] \leftarrow 1$
$last_prefix \leftarrow last_valid \leftarrow 0$
begin
for $i = 1$ to N **do**
  do
  `p_valid`$[i] \leftarrow last_valid$
  case study
  (1) $\hat{s}_j^l = x[\ell..i]$ is the longest **prefix** of some s_j^l
  if `valid`$[\ell-1] = \mathtt{T}$ OR `prefix`$[\ell-1] = \mathtt{T}$ **then**
    `prefix`$[\max\{last_prefix+1,\ell\}\ldots i] \leftarrow \mathtt{T}$
  end if
  if $x[p_valid[\ell-1]+1\ldots\ell-1]$ is a substring of some $s_j \in \mathcal{O}$ **then**
    `prefix`$[\max\{last_prefix+1,\ell\}\ldots i] \leftarrow \mathtt{T}$
  end if
  if `p_valid`$[\ell-1] \geq \ell - |s_j| - 1$ **then**
    Return "Invalid Image"
  end if
  $last_prefix \leftarrow i$
  (2) $\hat{s}_j^r = x[\ell..i]$ is the longest **prefix** of some s_j^r.
  if `prefix`$[\ell-1] = \mathtt{T}$ and $first_prefix \leq \ell - h + |s_j^r|$ **then**
    `prefix`$[\max\{last_prefix+1,\ell\}\ldots i] \leftarrow \mathtt{T}$
  end if
  if $\hat{s}_j^r = s_j^r$ **then**
    `valid`$[i] \leftarrow \mathtt{T}$
    $last_valid \leftarrow i$
  end if
  if $l_suffix[j][\ell-h-1] = \mathtt{T}$ and $x[\ell-h..\ell-1]$ is a satisfied binding **then**
    `prefix`$[\max\{last_prefix+1,\ell\}\ldots i] \leftarrow \mathtt{T}$
  end if
  if $\hat{s}_j = s_j$ **then**
    `valid`$[i] \leftarrow \mathtt{T}$
    $last_valid \leftarrow i$
  end if
  (3) $\bar{s}_j^l = x[\ell..i]$ is the largest **suffix** of some s_j^l.
  $l_suffix[j][i] \leftarrow \mathtt{T}$
  (4) $\bar{s}_j^r = x[\ell..i]$ is the largest **suffix** of some s_j^r.
  if `p_valid`$[i] \geq \ell - 1$ **then**
    `valid`$[i] \leftarrow \mathtt{T}$
    $last_valid \leftarrow i$
  end if
end for

The algorithm is based on Facts 1-4 as well as the following principles:

(a) The occurrence of a proper prefix of either a left or a right part of an object in a valid image must be followed by a prefix (not necessarily proper) of a left or a right part of an object.
(b) If the occurrence of a proper prefix of either a left or a right part of an object is followed by an occurrence of a proper suffix of either a left or a right part of an object, then the image is not valid. In a valid image, the occurrence of a proper suffix of an object is always preceded by the suffix of either a left or a right part of an object.
(c) The occurrence of a suffix of either a left or a right part of an object can be followed by either a prefix or a substring or a suffix.

(d) If an occurrence of a suffix of a left part of an object is not followed by either an occurrence of a prefix of its corresponding right part in a distance h or an occurrence of a prefix of a left part of an object in a distance at most h, then the image is not valid. In both cases a satisfied binding should separate the two parts.
(e) The occurrence of a substring in a valid image may be preceded by and followed by valid images.

Preprocessing Stage
In this stage we preprocess the set of objects. We compute the suffix tree of the set of the left and right parts of all objects in $\mathcal{O}$ [7,9,8]. This data structure will allow us to perform a constant time on-line checks whether a suffix, or a substring of s_j^l/s_j^r occurs in any position of x. We will also build the Aho-Corasick automaton [1] for the set of the left and right parts of all objects in $\mathcal{O}$ that will allow us to compute the largest prefixes of s_j^l/s_j^r occurring in x.

Main Algorithm
At the beginning of step i the algorithm has already determined whether $x[1..j]$ is a valid image or not, for all $j \in \{1..i-1\}$. Moreover, the algorithm should determine by the end of the current step whether $x[1..j]$ is valid or not for $j \in \{1..i\}$. This is achieved by examining the suffixes of $x[1..i]$. There are six possible cases: A suffix of $x[1..i]$ can be either a prefix of a left part, a prefix of a right part, a suffix of a left part, a suffix of a right part, a substring, a binding or a complete part of an object s_j for some $j \in \{1..k\}$. Otherwise, the string is not a valid image (Theorem 2).

Let $\hat{s}_j^l = x[\ell..i]$ be the longest prefix of a left part of an object in $\mathcal{O}$ that is also a suffix of $x[1..i]$. A prefix of a left part of an object is preceded by either a valid image, or a proper prefix of left/right part an object or a substring of an object.

- If *valid*$[\ell-1]$ is marked *TRUE*, then $x[1..\ell-1]$ is a valid image and position ℓ could be the beginning of a valid sub-image, thus we mark *prefix*$[i]$ = *TRUE*, *first-prefix* $= \ell$ and *last-prefix* $= i$.
- If *prefix*$[\ell-1]$ is marked *TRUE*, then we have a chain of prefixes, thus we mark *prefix*$[i]$ = *TRUE* and *last-prefix* $= i$.
- If there is no prefix of a left/right part of an object or a valid image preceding $\hat{s}_j^l$, then $x[1..i]$ is valid if and only if $x[$*previous-valid*$[\ell-1]+1..\ell-1]$ is a substring of left/right part of an object or $x[$*previous-valid*$[\ell-1]+1..i]$ is a prefix of a satisfied binding. If $x[$*previous-valid*$[\ell-1]+1..\ell-1]$ is a substring then ℓ is the start of a valid image.

Let $\hat{s}_j^r = x[\ell..i]$ be the longest prefix of a right part of an object in $\mathcal{O}$ that is also a suffix of $x[1..i]$. Similarly, a prefix of an object is preceded by either a proper prefix of left/right part an object or a substring of an object.

- If *prefix*$[\ell-1]$ is marked *TRUE* and *first-prefix* $\leq \ell - h + |s_j^r|$, then we have a chain of prefixes thus we mark *prefix*$[i]$ = *TRUE* and *last-prefix* $= i$. If $\hat{s}_j^r = s_j^r$ (a complete left part), then $x[1..i]$ is a valid image and we mark the relevant array as *TRUE*.
- If *l-suffix*$[j][\ell-h-1]$ is marked *TRUE* and $x[\ell-h..\ell-1]$ is a satisfied binding then we have a prefix of a valid image (Eq. (6)), thus we mark *prefix*$[i]$ = *TRUE* and *last-prefix* $= i$. If $\hat{s}_j^r = s_j^r$ (a complete left part), then $x[1..i]$ is a valid image and we mark the relevant array as *TRUE*.

Let $\bar{s}_j^l = x[\ell..i]$ be the longest suffix of a left part of an object in $\mathcal{O}$ that is also a suffix of $x[1..i]$. If *valid*$[\ell - 1]$ then *l-suffix*$[j][i]$ is marked *TRUE*.

Finally, let $\bar{s}_j^r = x[\ell..i]$ be the longest suffix of a right part of an object in $\mathcal{O}$ that is also a suffix of $x[1..i]$. Note that, in a valid image, a suffix $\bar{s}_j^r$ is always preceded by a valid image.

- If *previous-valid*$[\ell - 1] \geq \ell - 1$, then $x[1..i]$ is valid.
- If there is no valid image preceding $\bar{s}_j^r$, then $x[1..i]$ is valid if and only if the length of $i-$*previous-valid*$[\ell - 1] < |s_j|$.

Theorem 3. *Algorithm 1 validates an image x over a set $\mathcal{O}$ of objects of equal length and all and each composed of two parts separated by a hole in linear $O(|x|+|\mathcal{O}|)$ time.*

Proof. The construction of the Aho-Corasick automaton and the suffix tree of the dictionary $\mathcal{O}$ both require $O(|\mathcal{O})|$ time.

At Stage i, finding the largest suffix that is a prefix of some part of an object requires constant time. At Stage $i-1$, we have traced on the Aho-Corasick automaton the largest prefix of a part of an object that is a suffix of $x[1..i-1]$; on Stage i, we can either extend this prefix with one symbol, $x[i]$, or we can follow the failure link that lead to the largest such prefix. Each of the other lines of Algorithm 1 requires constant time and thus the bound on the running time follows.

5 Conclusions

We have presented an on-line algorithm that determines whether a given image is valid or not over a given set of objects with holes where each object composed of two parts separated by a transparent hole. We have solved the problem for a restricted set of objects. *I.e.* objects of the same lengths and presented a linear time algorithm. As future work, the algorithm may be modified in the same way as the original validation algorithm by [6], in order to deal with a set of objects of different lengths. Another interesting problem is the computation of the depth of an object in an image, *i.e.* the number of rules applied after the placement of an object in an image.

References

1. A. V. Aho and M. J. Corasick: *Efficient string matching: an aid to bibliographic search.* Commun. ACM, 18(6) 1975, pp. 333–340.
2. C. S. Iliopoulos and L. Mouchard: *Quasiperiodicity and string covering.* Theoretical Computer Science, 218(1) 1999, pp. 205–216.
3. C. S. Iliopoulos and J. F. Reid: *Validating and decomposing partially occluded two-dimensional images*, in Proc. Prague Stringology Club Workshop (PSCW'98), J. Holub and M. Šimánek, eds., 1998, pp. 83–94.
4. C. S. Iliopoulos and J. F. Reid: *Optimal parallel analysis and decomposition of partially occluded strings.* Parallel Computing, 26(4) 2000, pp. 483–494.
5. C. S. Iliopoulos and J. F. Reid: *Decomposition of partially occluded strings in the presence of errors.* International Journal of Pattern Recognition and Artificial Intelligence, 15(7) 2001, pp. 1129–1142.
6. C. S. Iliopoulos and J. Simpson: *On line validation and analysis of partially occluded images.* Journal of Automata, Languages and Combinatorics, 6(3) 2001, pp. 291–303.
7. E. M. McCreight: *A space-economical suffix tree construction algorithm.* J. ACM, 23(2) 1976, pp. 262–272.
8. E. Ukkonen: *On-line construction of suffix trees.* Algorithmica, 14(3) 1995, pp. 249–260.
9. P. Weiner: *Linear pattern matching algorithms*, in SWAT '73: Proceedings of the 14th Annual Symposium on Switching and Automata Theory (swat 1973), Washington, DC, USA, 1973, IEEE Computer Society, pp. 1–11.