Theoretical Computer Science 624 (2016) 121–135

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Efficient operations on discrete paths ☆

Alexandre Blondin Massé, Srečko Brlek *, Hugo Tremblay

ARTICLE INFO

Article history:
Received 12 March 2015
Accepted 13 July 2015
Available online 3 August 2015

Keywords:
Freeman code
Lattice paths
Radix tree
Discrete sets
Outer hull
Convex hull
Polyomino intersection
Union
Complement
Difference

ABSTRACT

We present linear time and space operations on discrete paths. First, we compute the outer hull of any discrete path. As a consequence, a linear time and space algorithm is obtained for computing the convex hull. Next, we provide a linear algorithm computing the overlay graph of two simple closed paths. From this overlay graph, one can easily compute the intersection, union and difference of two Jordan polyominoes, i.e. polyominoes whose boundary is a Jordan curve. The linear complexity is obtained by using an enriched version of a data structure introduced by Brlek, Koskas and Provençal: a quadtree for representing points in the discrete plane $\mathbb{Z} \times \mathbb{Z}$ augmented with neighborhood links, which was introduced in particular to decide in linear time if a discrete path is self-intersecting.

1. Introduction

The ever-growing use of digital screens in industrial, military and civil applications gave rise to a new branch of study of discrete objects, digital geometry, where the most basic objects are pixels. In particular, their geometric properties play an essential role in the design of efficient algorithms for recognizing patterns and extracting features: these are mandatory steps for an accurate interpretation of acquired images.

Fundamental geometric operations on sets of pixels, or discrete figures have been extensively studied. For instance, algorithms computing rotations, translations, symmetries, unions, intersections, dilations or segmentations of discrete figures are well documented (see [15] for a survey of the many algorithms available). However, none of the previous method is based on encodings of discrete figures by their boundary using combinatorics on words, a field which recently led to the development of efficient tools to study digital geometry (see [1,18]).

To illustrate the validity of this approach, consider the problem of finding the convex hull of a set of points. It is well known that for the Euclidean case, algorithms for computing the convex hull of a set $S \subset \mathbb{R}^2$ run in $\mathcal{O}(n \log n)$ time where $n = |S|$ (see [8,16]). One can also show that such algorithms are optimal (see [9,15,20] for the general case). In the digital case, the situation is made surprisingly easier with the help of combinatorics on words. For instance, linear asymptotic bounds are obtained when considering discrete paths encoded by elementary steps. Indeed, Brlek et al. designed a linear time algorithm for computing the discrete convex hull of nonself-intersecting closed paths in the square grid [6]. It is based on an optimal linear time and space algorithm for factorizing a word in Lyndon words designed by Duval [12].

☆ With the support of NSERC (Canada).

* Corresponding author.
E-mail address: brlek.srecko@uqam.ca (S. Brlek).

http://dx.doi.org/10.1016/j.tcs.2015.07.033
0304-3975/

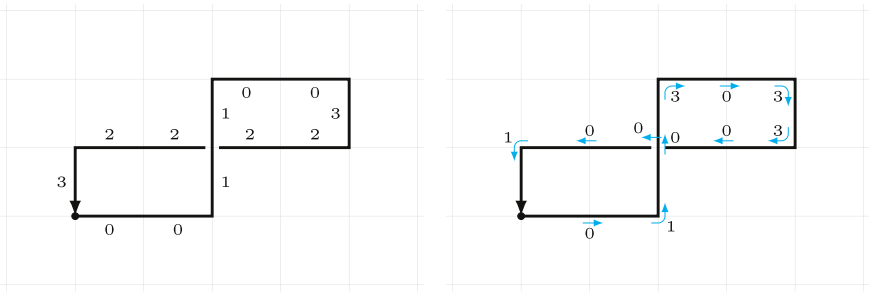

Fig. 1. (a) A discrete path coded by the word $w = \mathbf{001100322223}$ and (b) its first difference word $\Delta(w) = \mathbf{01030330001}$.

In this paper, we study fundamental geometric operations on connected discrete figures, or polyominoes with the help of combinatorics on words. As a first step, we describe a linear algorithm for computing the outer hull of any discrete path using the data structure described in [3] where the authors designed a linear time and space algorithm for detecting path intersection. It rests on a quadtree data structure induced by a natural radix order of $\mathbb{N} \times \mathbb{N}$. Then, each path is dynamically encoded by adding a pointer for each step of the discrete path encoded on the four letter alphabet $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$. Then, we extend those ideas to develop linear time and space algorithms for computing the overlay of two Jordan curves on $\mathbb{Z}^2$. As a byproduct, the convex hull of any discrete path, the intersection, the union and the difference of two Jordan polyominoes are computed in linear time.

2. Preliminaries

Given a finite alphabet Σ, a *word* w is a function $w : [1, 2, \ldots, n] \longrightarrow \Sigma$ denoted by its sequence of letters $w = w_1 w_2 \cdots w_n$, and $|w| = n$ is its *length*. For $a \in \Sigma$, $|w|_a$ is the number of letters a in w. The set of all words of length k is denoted by Σ^k. Consequently, $\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i$ is the set of all finite words on Σ where $\Sigma^0 = \{\varepsilon\}$, the set consisting of the empty word. The set Σ^* together with the operation of concatenation form a monoid called *the free monoid on* Σ.

Let w be any word. We say that the word u is a *factor* of w is there exist words x and y such that $w = xuy$. If $|x| = 0$ (resp. $|y| = 0$), then u is called a *prefix* (resp. *suffix*) of w.

There is a bijection between the set of pixels and $\mathbb{Z}^2$ obtained by mapping $(a, b) \in \mathbb{Z}^2$ to the unitary square whose bottom left vertex coordinate is (a, b). Therefore, we may consider pixels as elements of $\mathbb{Z}^2$. By definition, a *discrete set* S is a set of pixels, i.e. $S \subset \mathbb{Z}^2$. Also, two pixels are called 4-*adjacent* (resp. 8-adjacent) if their intersection is a unit segment (resp. a point). A set S is called 4-*connected* (resp. 8-connected) if for any pair of pixels $p, q \in S$, there exist pixels $p = p_0, p_1, p_2, \ldots, p_{k-1}, p_k = q$ such that p_i and p_{i+1} are 4-adjacent (resp. either 4- or 8-adjacent) for $i = 0, 1, \ldots, k-1$. Since any discrete set is a disjoint collection of 8-connected sets, we consider from now on that discrete sets are 4 or 8-connected. Also, define a *hole* of a discrete set S as a finite connected region of $\overline{S}$. Any 4-connected (resp. 8-connected) hole is called a 4-*hole* (resp. 8-*hole*).

A convenient way of representing discrete sets without hole is to use a word describing its contour (or boundary). In 1961, Freeman proposed an encoding of discrete objects by specifying their contour using the four elementary steps $(\rightarrow, \uparrow, \leftarrow, \downarrow) \simeq (\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3})$ [14]. This encoding provides an advantageous representation of discrete paths in $\mathbb{Z}^2$. A *discrete path* is a sequence of points $P = (p_1, p_2, \ldots, p_n)$ where p_i and p_{i+1} are 4-adjacent for $i = 1, 2, \ldots, n-1$. More precisely, two points p and q are called *neighbors* if $q = p + e$ for some elementary unit vector $e \in \{(1, 0), (0, 1), (-1, 0), (0, -1)\}$.

It is not hard to see that every discrete figure without hole can be represented by a closed and simple discrete path. From now on, we concentrate on the more general concept of discrete paths, referencing discrete figures without hole as "simple and closed discrete paths". For example, the discrete figure of the single pixel $(0, 0)$ is regarded as the path $((0, 0); \mathbf{0123})$.

It is worth mentioning that in the case of a closed discrete path, w is unique up to a circular permutation of its letters and the sense of travel. For example, any circular permutation of the word $w = \mathbf{001100322223}$ represents the discrete path shown in Fig. 1(a). One says that a word $w \in \mathcal{F}^*$ is *closed* if and only if $|w|_0 = |w|_2$ and $|w|_1 = |w|_3$. Further, w is called *simple* if it codes a nonself-intersecting discrete path, i.e. its only closed factors are ε and possibly w itself. For instance, $w = \mathbf{001100322223}$ is nonsimple and closed.

It is sometimes useful to consider encoding of paths with turns instead of elementary steps. Such encoding is obtained from the contour word $w = w_1 \cdots w_n$ by setting

$$\Delta(w) = (w_2 - w_1)(w_3 - w_2) \cdots (w_n - w_{n-1})$$

where subtraction is computed modulo 4. $\Delta(w)$ is called the *first differences word of* w. Letters of $\Delta(w) \in \mathcal{F}^*$ are interpreted via the bijection $(\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}) \simeq (\text{forward, left turn, U-turn, right turn})$. It is worth mentioning that for any closed path w, the first difference word of w is $\Delta(w)(w_1 - w_n)$, where $\Delta(w)$ is defined as above. For example, one can verify in Fig. 1(b) that $\Delta(w) = \mathbf{01030330001}$ and that it codes the turns of w.