Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 346 (2005) 200–225
Theoretical Computer Science
www.elsevier.com/locate/tcs

The discrete Green Theorem and some applications in discrete geometry☆

S. Brlek, G. Labelle, A. Lacasse*

LaCIM, Université du Québec à Montréal, C. P. 8888 Succursale "Centre-Ville", Montréal (QC), Canada H3C 3P8

Dedicated to the memory of Alberto Del Lungo

Abstract

The discrete version of Green's Theorem and bivariate difference calculus provide a general and unifying framework for the description and generation of incremental algorithms. It may be used to compute various statistics about regions bounded by a finite and closed polygonal path. More specifically, we illustrate its use for designing algorithms computing many statistics about polyominoes, regions whose boundary is encoded by four letter words: area, coordinates of the center of gravity, moment of inertia, set characteristic function, the intersection with a given set of pixels, hook-lengths, higher order moments and also q-statistics for projections.

Keywords: Discrete Green Theorem; Discrete geometry; Polyominoes

1. Introduction

The classical Green's Theorem may be seen as a generalization of the Fundamental Theorem of Calculus and links surface integrals to contour integrals. More precisely, for

☆ The work was supported by NSERC (Canada).

* Corresponding author.
E-mail addresses: brlek@lacim.uqam.ca (S. Brlek), gilbert@lacim.uqam.ca (G. Labelle), lacasse@lacim.uqam.ca (A. Lacasse).

0304-3975/$ - see front matter © 2005 Published by Elsevier B.V.
doi:10.1016/j.tcs.2005.08.019

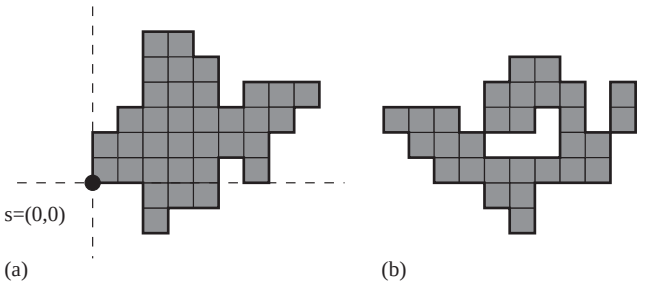


Fig. 1. (a) A typical polyomino; (b) a closed curve but not a polyomino.

any convenient closed region Ω of the plane with boundary $\delta(\Omega)$ we have

$$\iint_{\Omega}\left(\frac{\partial Q}{\partial x}-\frac{\partial P}{\partial y}\right)\mathrm{d}x\,\mathrm{d}y=\int_{\delta(\Omega)}P(x,y)\,\mathrm{d}x+Q(x,y)\,\mathrm{d}y.$$

This is particularly true for regions defined on regular lattices such as square, hexagonal or triangular lattices of the plane. On the other hand, many basic parameters associated with closed regions are represented by surface integrals. For instance, the area $\mathrm{A}(\Omega)$, center of gravity $\mathrm{CG}(\Omega)$, moment of inertia $\mathrm{I}(\Omega)$, of a closed region Ω are defined by the double integrals

$$\mathrm{A}(\Omega)=\iint_{\Omega}\mathrm{d}x\,\mathrm{d}y,\quad \mathrm{CG}(\Omega)=(\bar{x},\bar{y})=\left(\frac{\iint_{\Omega}x\,\mathrm{d}x\,\mathrm{d}y}{\mathrm{A}(\Omega)},\frac{\iint_{\Omega}y\,\mathrm{d}x\,\mathrm{d}y}{\mathrm{A}(\Omega)}\right),$$
$$\mathrm{I}(\Omega)=\iint_{\Omega}((x-\bar{x})^2+(y-\bar{y})^2)\,\mathrm{d}x\,\mathrm{d}y=\iint_{\Omega}(x^2+y^2)\,\mathrm{d}x\,\mathrm{d}y-(\bar{x}^2+\bar{y}^2)\mathrm{A}(\Omega).$$

In this paper we restrict the study to regions that are commonly used in discrete geometry, namely the polyominos, but one should keep in mind that a more general formulation could be presented. A polyomino **P** is a finite union of closed cells in the unit lattice square (pixels) of the plane whose boundary $\delta(\mathbf{P})$ consists of a simple closed polygonal path (see Fig. 1(a)). In particular, our polyominoes are simply connected (contain no holes), and have no multiple points (see Fig. 1(b)). The polygonal path γ (*contour*) of a polyomino is encoded by an ordered pair (s, w) where s is a lattice point belonging to γ and w is a word over the 4-letter alphabet

$$\mathcal{A}=\{\mathbf{r},\mathbf{u},\mathbf{l},\mathbf{d}\}=\{\mathbf{r}:\rightarrow\quad\mathbf{u}:\uparrow\quad\mathbf{l}:\leftarrow\quad\mathbf{d}:\downarrow\}$$

also known as the Freeman chain code [9,10], where the letters correspond to the unit translations in the lattice directions: right, up, left and down. The word w represents the perimeter of the polyomino read in a counterclockwise way starting from the point s. The use of s may be avoided in the encodings by assuming that s is always the lowest left most point of the polyomino and that $s=(0,0)$ by using a suitable translation. In this way, the polyomino of Fig. 1(a) is encoded by the single word

$$w=rrddrurruurdruururullldluululldddldldd.$$

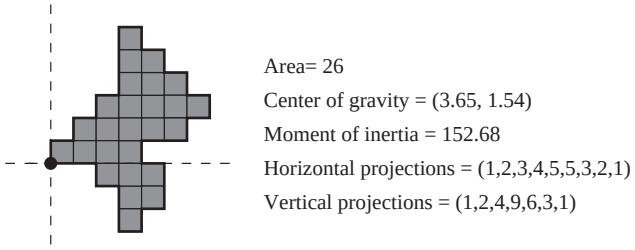


Fig. 2. Some parameters for polyominoes.

Since polyominoes are given by words describing their contours, it is natural to use Green's Theorem for the construction of our first general algorithms in order to compute not only some basic statistics such as the area, center of gravity, moment of inertia, projections (see Fig. 2) but the boolean operations on the underlying sets as well.

In Section 2, we introduce the notion of incremental algorithm for polyominoes given by their contour and show how Green's Theorem can be used to generate families of such algorithms. In Section 3, we drop the continuity conditions of Green's Theorem and deal with general additive incremental algorithms for which the output associated with the sum of two polyominoes is the sum of the outputs associated to each polyomino.

More general algorithms are then obtained by the use of weight functions $W : \mathbb{Z} \times \mathbb{Z} \longrightarrow \mathbb{A}$. In particular, if W is the boolean valued characteristic function of a point, then the output of the algorithm is boolean valued and decides if a given pixel belongs to a given polyomino. This result extends to sets of pixels, providing the computation of the set characteristic function and some particular instances such as the size of hook-lengths. Higher order moments are also obtained in this way when the weight function involves Stirling numbers of the second kind. When $\mathbb{A}$ is a ring of formal Laurent power series, the use of q-analogues yields the simultaneous computation of both the horizontal and vertical projections.

The power and effectiveness of Green's Theorem already appeared in the literature. More precisely, it is useful for region filling (see for example [15]) and also for the efficient computation of the moments of closed regions [14,17,18]. Our present approach is similar to the one given in [14,17,18], but differs by the choice of the Stirling numbers instead of the Bernouilli numbers. For a general presentation of polyominoes and their properties see [11]. A survey of enumerative results concerning polyominoes can be found in [16] (see also [2,4,7]). The core of the third author's Master thesis [12] contains in full detail—but is not limited to—the results presented here with numerous examples.

2. Green's Theorem and incremental algorithms

The following version of Green's Theorem [13] is sufficient to start our analysis.

Theorem 1. *Let $P(x, y)$, $Q(x, y)$ be two continuously differentiable functions on an open set containing a simply connected region Ω bounded by a simple piecewise continuously*

differentiable positively oriented curve Γ. Then

$$\iint_{\Omega}\left(\frac{\partial Q}{\partial x}-\frac{\partial P}{\partial y}\right)\mathrm{d}x\,\mathrm{d}y=\int_{\Gamma}P(x,y)\,\mathrm{d}x+Q(x,y)\,\mathrm{d}y.$$

Since the above parameters involve integrals of the form

$$\iint_{\Omega}f(x,y)\,\mathrm{d}x\,\mathrm{d}y,$$

where Ω will be a polyomino, our next step is to choose $P(x,y)$ and $Q(x,y)$, in Green's Theorem, such that $(\partial Q/\partial x-\partial P/\partial y)=f$. There are many ways to achieve this and three solutions are provided in the following useful lemma.

Lemma 2. *Let* **P** *be a polyomino with contour γ, and let $f(x,y)$ be a $\mathbb{R}$-valued continuous function. Then,*

$$\iint_{\mathbf{P}}f(x,y)\,\mathrm{d}x\,\mathrm{d}y=\int_{\gamma}f_1(x,y)\,\mathrm{d}y \tag{1}$$

$$=-\int_{\gamma}f_2(x,y)\,\mathrm{d}x \tag{2}$$

$$=\int_{\gamma}F(x,y)(x\,\mathrm{d}y-y\,\mathrm{d}x), \tag{3}$$

where

$$f_1(x,y)=\int^{x}f(u,y)\,\mathrm{d}u,\quad f_2(x,y)=\int^{y}f(x,v)\,\mathrm{d}v,$$

$$F(x,y)=\int_0^1 f(sx,sy)s\,\mathrm{d}s.$$

The notation $\int_\gamma$ denotes a line integral along γ while $\int^t \mathrm{d}t$ means indefinite integration.

Proof. For (1), take $P=0$, $Q=f_1$ in Green's Theorem. For (2), take $P=-f_2$, $Q=0$. Formula (3) is more delicate and can be established as follows. Take, in Green's Theorem, $P(x,y)=-yF(x,y)$ and $Q(x,y)=xF(x,y)$. We must show that $(\partial Q/\partial x-\partial P/\partial y)=f$. In order to do this, note first that

$$\left(\frac{\partial Q}{\partial x}-\frac{\partial P}{\partial y}\right)=2F+x\frac{\partial F}{\partial x}+y\frac{\partial F}{\partial y}.$$

Next, consider an extra variable u such that $0<u\leqslant 1$. Then,

$$u^2F(ux,uy)=u^2\int_0^1 f(sux,suy)s\,\mathrm{d}s$$

$$=\int_0^u f(\sigma x,\sigma y)\sigma\,\mathrm{d}\sigma\quad(\text{via }\sigma=su).$$

Differentiating with respect to u gives

$$2uF(ux,uy)+u^2\frac{\partial F}{\partial x}(ux,uy)x+u^2\frac{\partial F}{\partial y}(ux,uy)y=uf(ux,uy).$$

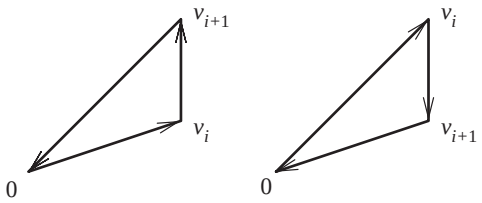


Fig. 3. A positive and a negative triangle.

Finally, taking $u = 1$, one obtains the desired equality

$$2F + x\frac{\partial F}{\partial x} + y\frac{\partial F}{\partial y} = f. \qquad \square$$

Remark 3. The above proof of (3) uses Green's Theorem but it is more algebraic than geometric. An alternate geometric proof of (3), not using Green's Theorem, is provided now. Its advantage relies on the fact that it may be adapted to any piecewise continuously differentiable curve γ.

Alternate geometric proof of (3): Let $\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_n$ be the successive vertices of the contour γ of $\mathbf{P}$. For any two successive vertices $\mathbf{v}_i$ and $\mathbf{v}_{i+1}$ on γ, consider the triangle T_i whose vertices are $0, \mathbf{v}_i$ and $\mathbf{v}_{i+1}$ taken in this order. The triangle T_i is considered to be positive if the angle defined by the vectors $\mathbf{v}_i$ and $\mathbf{v}_{i+1}$ is positive and T_i is negative otherwise (see Fig. 3).

We obviously have,

$$\int\int_{\mathbf{P}} f(x, y)\,\mathrm{d}x\,\mathrm{d}y = \sum_{i=0}^{n-1} \int\int_{T_i} f(x, y)\,\mathrm{d}x\,\mathrm{d}y.$$

Now let $\mathbf{v}_i = (x_i, y_i)$ and $\mathbf{v}_{i+1} = (x_{i+1}, y_{i+1}) = (x_i + \Delta x_i, y_i + \Delta y_i)$ and take the following parametrization for the triangle T_i:

$$x = x(s, t) = s(x_i + t\Delta x_i), \qquad y = y(s, t) = s(y_i + t\Delta y_i),$$

where $0 \leqslant s \leqslant 1$ and $0 \leqslant t \leqslant 1$. The Jacobian of this transformation is

$$\frac{\partial(x, y)}{\partial(s, t)} = s(x_i\Delta y_i - y_i\Delta x_i).$$

By the change of variables for double integrals, we have, for $i = 0, 1, \ldots, n-1$:

$$\begin{aligned}
\int\int_{T_i} f(x, y)\,\mathrm{d}x\,\mathrm{d}y &= \int_0^1\int_0^1 f(x(s, t), y(s, t))\frac{\partial(x, y)}{\partial(s, t)}\,\mathrm{d}s\,\mathrm{d}t \\
&= \int_0^1\int_0^1 f(s(x_i + t\Delta x_i), s(y_i + t\Delta y_i))s(x_i\Delta y_i - y_i\Delta x_i)\,\mathrm{d}s\,\mathrm{d}t \\
&= \int_0^1 F(x_i + t\Delta x_i, y_i + t\Delta y_i)(x_i\Delta y_i - y_i\Delta x_i)\,\mathrm{d}t \\
&= \int_{[\mathbf{v}_i, \mathbf{v}_{i+1}]} F(x, y)(x\,\mathrm{d}y - y\,\mathrm{d}x). \qquad \square
\end{aligned}$$

2.1. Incremental algorithms

The evaluation of each line integral (1)–(3) of Lemma 2 can be broken into simpler integrals over successive unit (horizontal or vertical) line segments forming γ:

$$\int_{\gamma} \alpha = \sum_{i=0}^{n-1} \int_{[\mathbf{v}_i, \mathbf{v}_{i+1}]} \alpha,$$

where $\mathbf{v}_i = (x_i, y_i)$, $i = 0, \ldots, n-1$, denote the successive vertices of the contour of $\mathbf{P}$, and satisfy $\mathbf{v}_n = \mathbf{v}_0$, $\mathbf{v}_{i+1} = \mathbf{v}_i + \Delta\mathbf{v}_i = (x_i + \Delta x_i, y_i + \Delta y_i)$.

Since polyominoes are coded by (s, w) where $s \in \mathbb{Z} \times \mathbb{Z}$ is the starting point and w is a word over the alphabet $\mathcal{A} = \{r, u, l, d\}$, the translation into incremental algorithms follows easily:

start from the source point s and traverse the contour $\delta(P) = \gamma$ by reading w letter by letter. At each step, the performed action depends only on the current position on $\delta(P)$ and on the letter read.

More precisely, consider four vectors identified with the letters of $\mathcal{A}$

$$\mathbf{r} = (1, 0), \quad \mathbf{u} = (0, 1), \quad \mathbf{l} = (-1, 0), \quad \mathbf{d} = (0, -1)$$

and take four functions indexed by $\mathcal{A}$,

$$\Phi_r(x, y), \quad \Phi_u(x, y), \quad \Phi_l(x, y), \quad \Phi_d(x, y).$$

Now read the word $w = w_1 w_2 \ldots w_n$ sequentially from the left, cumulating the partial sums as follows, where $\mathbf{w}_i$ is the vector corresponding to the letter w_i:

$\mathbf{v} := (x_0, y_0)$; $S := 0$;
for $i := 1$ **to** n **do**
  $S := S + \Phi_{w_i}(\mathbf{v})$; $\mathbf{v} := \mathbf{v} + \mathbf{w}_i$
end for
return S.

Hereafter an incremental algorithm is denoted by $\bullet = \langle\, \Phi_d\,, \Phi_h\,, \Phi_g\,, \Phi_b\, \rangle$ and the following suggestive notation represents its output:

$$\mathrm{Output}(\bullet, \mathbf{P}) = \sum_{\rightarrow} \Phi_r(x_i, y_i) + \sum_{\uparrow} \Phi_u(x_i, y_i) + \sum_{\leftarrow} \Phi_l(x_i, y_i) + \sum_{\downarrow} \Phi_d(x_i, y_i).$$

The formulas (1), (2) and (3) of Lemma 2 yield the corresponding incremental algorithms called, respectively, *V-algorithm*, *H-algorithm* and *VH-algorithm*, where the letters V and H stand for the vertical and horizontal directions: in a *V-algorithm* (resp. *H-algorithm*) only vertical (resp. horizontal) sides of the polyomino are used, while in a *VH-algorithm* both vertical and horizontal sides are used.

Proposition 4 (*Green's type algorithms*). *Let* $\mathbf{P} = (s, w)$ *and* $f(x, y)$ *be continuous. Then,*

$$\iint_{\mathbf{P}} f(x, y)\, \mathrm{d}x\, \mathrm{d}y = \sum_{\rightarrow} \Phi_r(x_i, y_i) + \sum_{\uparrow} \Phi_u(x_i, y_i) + \sum_{\leftarrow} \Phi_l(x_i, y_i) + \sum_{\downarrow} \Phi_d(x_i, y_i),$$

where the functions $\Phi_r,\ \Phi_u,\ \Phi_l,\ \Phi_d$ *are taken from any of the following three sets of possibilities*:

V-algo. $\Phi_r = 0, \quad \Phi_u = \int_0^1 f_1(x, y+t)\,\mathrm{d}t, \quad \Phi_l = 0, \quad \Phi_d = -\int_0^1 f_1(x, y-t)\,\mathrm{d}t.$

H-algo. $\Phi_r = -\int_0^1 f_2(x+t, y)\,\mathrm{d}t, \quad \Phi_u = 0, \quad \Phi_l = \int_0^1 f_2(x-t, y)\,\mathrm{d}t, \quad \Phi_d = 0.$

VH-algo. $\Phi_r = -y\int_0^1 F(x+t, y)\,\mathrm{d}t, \quad \Phi_u = x\int_0^1 F(x, y+t)\,\mathrm{d}t,$

$$\Phi_l = y\int_0^1 F(x-t, y)\,\mathrm{d}t, \qquad \Phi_d = -x\int_0^1 F(x, y-t)\,\mathrm{d}t,$$

where $f_1(x, y)$, $f_2(x, y)$ *and* $F(x, y)$ *are defined by Lemma* 2.

Proof. Let α be any one of the three differential forms

$$f_1(x, y)dy, \quad -f_2(x, y)\,\mathrm{d}x, \quad F(x, y)(x\,\mathrm{d}y - y\,\mathrm{d}x)$$

appearing in the line integrals (1), (2), (3) of Lemma 2. Then,

$$\iint_{\mathbf{P}} f(x, y)\,\mathrm{d}x\,\mathrm{d}y = \int_\gamma \alpha = \sum_{i=0}^{n-1} \int_{[\mathbf{v}_i, \mathbf{v}_{i+1}]} \alpha,$$

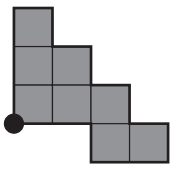
where $\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{n-1}, \mathbf{v}_n (= \mathbf{v}_0)$ are the vertices of the contour γ of $\mathbf{P}$. Now if (s, w) encodes $\mathbf{P}$, with the starting point (x_0, y_0) and the 4-letter word $w = w_1 w_2 \ldots w_n$, then the side $[\mathbf{v}_i, \mathbf{v}_{i+1}]$ of the contour γ is parametrized by $(x, y) = (x(t), y(t)),\ 0 \leqslant t \leqslant 1$, where

$$\begin{array}{llll}
x = x(t) = x_i + t, & y = y(t) = y_i & (\mathrm{d}x = \mathrm{d}t\mathrm{d}y = 0) & \text{if } w_{i+1} = r,\\
x = x(t) = x_i, & y = y(t) = y_i + t & (\mathrm{d}x = 0\mathrm{d}y = \mathrm{d}t) & \text{if } w_{i+1} = u,\\
x = x(t) = x_i - t, & y = y(t) = y_i & (\mathrm{d}x = -\mathrm{d}t, \mathrm{d}y = 0) & \text{if } w_{i+1} = l,\\
x = x(t) = x_i, & y = y(t) = y_i - t & (\mathrm{d}x = 0, \mathrm{d}y = -\mathrm{d}t) & \text{if } w_{i+1} = d.
\end{array}$$

We conclude by evaluating the line integrals (1), (2), (3) of Lemma 2 using the corresponding parametrizations. □

2.2. Elementary applications and examples

The tables below contain elementary instances of these algorithms for the computation of $\iint_{\mathbf{P}} f(x, y)\,\mathrm{d}x\,\mathrm{d}y$ and some computations are carried out on the simple polyomino $w = rrdrrululululdddd$ and $s = (0, 0)$:



Below are listed the algorithms for the area (Table 1), where $f(x, y) = 1$; for the center of gravity (Table 2), where $f(x, y) = x$ and $f(x, y) = y$; and for the moment of inertia

Table 1
Area

Algorithm	Φ_r	Φ_u	Φ_l	Φ_d
V	0	x	0	$-x$
H	$-y$	0	y	0
VH	$-y/2$	$x/2$	$y/2$	$-x/2$

(Table 3), where $f(x, y) = x^2 + y^2$.

- *V-algo* for the area: $\sum_{\rightarrow} 0 + \sum_{\uparrow} x_i + \sum_{\leftarrow} 0 + \sum_{\downarrow} - x_i$,

$$\begin{aligned}\iint_P 1\,\mathrm{d}x\,\mathrm{d}y &= 0 + 0 - x_3 + 0 + 0 + x_5 + 0 + x_7 + 0 + x_9 + 0 + x_{11}\\ &\quad + 0 - x_{13} - x_{14} - x_{15}\\ &= -2 + 4 + 3 + 2 + 1 - 0 - 0 - 0 = 8.\end{aligned}$$

- *VH-algo* for the area: $\sum_{\rightarrow} - y_i/2 + \sum_{\uparrow} x_i/2 + \sum_{\leftarrow} y_i/2 + \sum_{\downarrow} - x_i/2$,

$$\begin{aligned}\iint_P 1\,\mathrm{d}x\,\mathrm{d}y &= -y_0/2 - y_1/2 - x_2/2 - y_3/2 - y_4/2 + x_5/2 + y_6/2 - y_7/2 + y_8/2\\ &\quad - x_9/2 + y_{10}/2 + x_{11}/2 + y_{12}/2 - x_{13}/2 - x_{14}/2 - x_{15}/2\\ &= -1 + 1/2 + 1/2 + 2 + 3/2 + 1/2 + 1 + 1 + 1/2 + 3/2 = 8.\end{aligned}$$

- *V-algo* for $\bar{x}$ of the center of gravity: $\sum_{\rightarrow} 0 + \sum_{\uparrow} x_i^2/2 + \sum_{\leftarrow} 0 + \sum_{\downarrow} - x_i^2/2$,

$$\begin{aligned}\iint_P x\,\mathrm{d}x\,\mathrm{d}y &= 0 + 0 - x_2^2/2 + 0 + 0 + x_5^2/2 + 0 + x_7^2/2 + 0 + x_9^2/2 + 0\\ &\quad + x_{11}^2/2 + 0 - x_{13}^2/2 - x_{14}^2/2 - x_{15}^2/2\\ &= (-2^2 + 4^2 + 3^2 + 2^2 + 1^2)/2 = 26/2 = 13.\end{aligned}$$

- *V-algo* for the integral involved in the moment of inertia:

$$\begin{aligned}\iint_P (x^2 + y^2)\,\mathrm{d}x\,\mathrm{d}y &= \sum_{\uparrow} x_i^3/3 + x_i y_i^2 + x_i y_i + x_i/3\\ &\quad + \sum_{\downarrow} - x_i^3/3 - x_i y_i^2 + x_i y_i - x_i/3 = 425/24.\end{aligned}$$

- We compute now the probability that a random point $(x, y) \in \mathbb{R} \times \mathbb{R}$, under a normal bivariate probability distribution, $f(x, y) = (1/\pi)\exp(-x^2 - y^2)$, falls in a given polyomino **P**. In this case the *VH-algorithm* is complicated and only the V and *H-algorithms* are given (Table 4). Discrete probability distributions (such as uniform distributions over rectangles) will be considered in the next section.

Due to its formulation, the *VH-algorithm* is in general more complicated than the corresponding V and *H-algorithms*. There is, however, an important class of functions for which the *VH-algorithm* is generally preferable: the class of *homogeneous functions*, i.e. functions $f(x, y)$, satisfying a functional equation of the form $f(sx, sy) = s^k f(x, y)$ for a constant k, called the *degree of homogeneity*. The *VH-algorithm* is given now.

Table 2
Center of gravity

Algorithm	Φ_r	Φ_u	Φ_l	Φ_d
V (num $\bar{x}$)	0	$x^2/2$	0	$-x^2/2$
(num $\bar{y}$)	0	$x/2+xy$	0	$x/2-xy$
H (num $\bar{x}$)	$-y/2-xy$	0	$-y/2+xy$	0
(num $\bar{y}$)	$-y^2/2$	0	$y^2/2$	0
VH (num $\bar{x}$)	$-xy/3-y/6$	$x^2/3$	$xy/3-y/6$	$-x^2/3$
(num $\bar{y}$)	$-y^2/3$	$xy/3+x/6$	$y^2/3$	$-xy/3+x/6$

Table 3
Moment of inertia

V	$\Phi_r = 0$	$\Phi_u = x/3+xy+x^3/3+xy^2$
	$\Phi_l = 0$	$\Phi_d = -x/3+xy-x^3/3-xy^2$
H	$\Phi_r = -y/3-xy-x^2y-y^3/3$	$\Phi_u = 0$
	$\Phi_l = y/3-xy+x^2y+y^3/3$	$\Phi_d = 0$
VH	$\Phi_r = -y/12-xy/4-x^2y/4-y^3/4$	$\Phi_u = x/12+xy/4+x^3/4+xy^2/4$
	$\Phi_l = y/12-xy/4+x^2y/4+y^3/4$	$\Phi_d = -x/12+xy/4-x^3/4-xy^2/4$

Table 4
$f(x,y) = (1/\pi)\exp(-x^2-y^2)$, $\operatorname{erf}(x) = (2/\sqrt{\pi})\int_0^x \exp(-t^2)\,\mathrm{d}t$

V	$\Phi_r = 0$	$\Phi_u = \frac{1}{4}\operatorname{erf}(x)(\operatorname{erf}(y+1)-\operatorname{erf}(y))$
	$\Phi_l = 0$	$\Phi_d = \frac{1}{4}\operatorname{erf}(x)(\operatorname{erf}(y-1)-\operatorname{erf}(y))$
H	$\Phi_r = -\frac{1}{4}\operatorname{erf}(y)(\operatorname{erf}(x+1)-\operatorname{erf}(x))$	$\Phi_u = 0$
	$\Phi_l = -\frac{1}{4}\operatorname{erf}(y)(\operatorname{erf}(x-1)-\operatorname{erf}(x))$	$\Phi_d = 0$

Corollary 5. *Let $f(x,y)$ be continuous and homogeneous of degree $k > -2$. Assume that*

$$\Phi_r = -\frac{y}{k+2}(f_1(x+1,y)-f_1(x,y)), \quad \Phi_u = \frac{x}{k+2}(f_2(x,y+1)-f_2(x,y)),$$

$$\Phi_l = -\frac{y}{k+2}(f_1(x-1,y)-f_1(x,y)), \quad \Phi_d = \frac{x}{k+2}(f_2(x,y-1)-f_2(x,y)),$$

where $f_1(x,y)$ and $f_2(x,y)$ are defined in Lemma 2. Then the corresponding incremental VH-algorithm computes $\iint_{\mathbf{P}} f(x,y)\,\mathrm{d}x\,\mathrm{d}y$, for any polyomino **P**.

Proof. Let $f(x,y)$ be homogeneous of degree k. Then the function $F(x,y)$ of Proposition 4 takes the very simple form

$$F(x,y) = \int_0^1 f(sx,sy)s\,\mathrm{d}s = \int_0^1 s^{k+1}f(x,y)\,\mathrm{d}s = \frac{1}{k+2}f(x,y).$$

Hence, for the corresponding *VH-algorithm*, we have,

$$\Phi_r(x, y) = -\int_0^1 F(x+t, y)y \, \mathrm{d}t = -\frac{y}{k+2}\int_0^1 f(x+t, y) \, \mathrm{d}t$$
$$= -\frac{y}{k+2}\int_x^{x+1} f(x, y) \, \mathrm{d}x = -\frac{y}{k+2}(f_1(x+1, y) - f_1(x, y)),$$

by definition of $f_1(x, y)$. The verification of the formulas for Φ_u, Φ_l and Φ_d is left to the reader. □

A typical illustration of Corollary 5, for which the *VH-algorithm* is simpler than the corresponding *V* or *H-algorithms*, is provided by the computation of the average euclidean distance from a given point $(a, b) \in \mathbb{Z} \times \mathbb{Z}$ to a random point in a polyomino **P** is given by the formula

$$\frac{\iint_{\mathbf{P}} \sqrt{(x-a)^2 + (y-b)^2} \, \mathrm{d}x \, \mathrm{d}y}{\mathrm{A}(\mathbf{P})},$$

where A(**P**) is computed by some of our previous algorithms. We only need to compute the integral $\iint_{\mathbf{P}} f(x, y) \, \mathrm{d}x \, \mathrm{d}y$. This is achieved easily by replacing the starting point $s = (x_0, y_0)$ by $s - (a, b) = (x_0 - a, y_0 - b)$. It corresponds to the choice $f(x, y) = \sqrt{x^2 + y^2}$ and $k = 1$ in Corollary 5. In this case, the functions $f_1(x, y)$ and $f_2(x, y)$ are given by the formulas

$$f_1(x, y) = \begin{cases} \frac{1}{2}x|x| & \text{if } y = 0, \\ \frac{1}{2}x\sqrt{x^2+y^2} + \frac{1}{2}y^2 \ln(x + \sqrt{x^2+y^2}) & \text{otherwise,} \end{cases}$$

$$f_2(x, y) = \begin{cases} \frac{1}{2}y|y| & \text{if } x = 0, \\ \frac{1}{2}y\sqrt{x^2+y^2} + \frac{1}{2}x^2 \ln(y + \sqrt{x^2+y^2}) & \text{otherwise.} \end{cases}$$

Note that $f_1(0, 0) = f_2(0, 0) = 0$ by taking limits.

3. Additive incremental algorithms and applications

In the foreseen examples, the function $f(x, y)$ was assumed to be continuous. Nevertheless this much restrictive condition may be dropped by assuming, for example, that f is piecewise continuous in each variable, and we still may use Proposition 4 as a guideline for producing corresponding algorithms. Indeed, algorithms for the computation of horizontal and vertical projections of a polyomino can be found in this way: fix an integer α and define f by

$$f(x, y) = \chi(\alpha \leqslant x < \alpha + 1),$$

where χ denotes the characteristic function. Then, $\iint_{\mathbf{P}} f(x, y) \, \mathrm{d}x \, \mathrm{d}y$ is clearly the α-vertical projection of the polyomino **P**:

$$\iint_{\mathbf{P}} f(x, y) \, \mathrm{d}x \, \mathrm{d}y = \#\{\beta \in \mathbb{Z} | \mathrm{Pix}_{\alpha,\beta} \subseteq \mathbf{P}\} = v_\alpha(\mathbf{P}),$$

where $\mathrm{Pix}_{\alpha,\beta}$ denotes the unit pixel of the plane having the point $(\alpha, \beta) \in \mathbb{Z} \times \mathbb{Z}$ as its lowest left corner, that is:

$$\mathrm{Pix}_{\alpha,\beta} = \{(x, y) \in \mathbb{R} \times \mathbb{R} | \alpha \leqslant x < \alpha + 1,\ \beta \leqslant y < \beta + 1\},$$

and its closure (with condition $x \leqslant \alpha+1$, $y \leqslant \beta+1$) is denoted $\overline{\mathrm{Pix}_{\alpha,\beta}}$. In this case, following Proposition 4, we find that

$$f_1(x, y) = \int^x \chi(\alpha \leqslant x < \alpha + 1)\, \mathrm{d}x = \begin{cases} 0 & \text{if } x < \alpha, \\ x - \alpha & \text{if } \alpha \leqslant x < \alpha + 1, \\ 1 & \text{if } \alpha + 1 \leqslant x. \end{cases}$$

This gives the following algorithm as the reader can easily check:

V-algorithm for the vertical projection $v_\alpha(\mathbf{P})$:

$$\Phi_r = 0, \quad \Phi_u = \chi(x \geqslant \alpha + 1), \quad \Phi_l = 0, \quad \Phi_d = -\chi(x \geqslant \alpha + 1).$$

Similarly, taking $f(x, y) = \chi(\beta \leqslant y < \beta + 1)$, the β-horizontal projection of $\mathbf{P}$

$$\#\{\alpha \in \mathbb{Z} \mid \mathrm{Pix}_{\alpha,\beta} \subseteq \mathbf{P}\} = h_\beta(\mathbf{P}),$$

is computed by the following algorithm:

H-Algorithm for the horizontal projection $h_\beta(\mathbf{P})$:

$$\Phi_r = -\chi(y \geqslant \beta + 1), \quad \Phi_u = 0, \quad \Phi_l = \chi(y \geqslant \beta + 1), \quad \Phi_d = 0.$$

These algorithms for the projections are special instances of the general notion of additive incremental algorithm which we now define.

Definition 6. An incremental algorithm $\bullet = \langle\, \Phi_r\,, \Phi_u\,, \Phi_l\,, \Phi_d\, \rangle$ is additive if, whenever $\mathbf{P} = \mathbf{P}_1 \cup \mathbf{P}_2$ with disjoint interiors (see Fig. 4), we have

$$\mathrm{Output}(\bullet, \mathbf{P}) = \mathrm{Output}(\bullet, \mathbf{P}_1 \cup \mathbf{P}_2) = \mathrm{Output}(\bullet, \mathbf{P}_1) + \mathrm{Output}(\bullet, \mathbf{P}_2).$$

An example of a nonadditive incremental algorithm is given by the computation of a polyomino's perimeter in which case $\Phi_r = \Phi_u = \Phi_l = \Phi_d = 1$.

Proposition 7. *An incremental algorithm* $\bullet = \langle \Phi_r, \Phi_u, \Phi_l, \Phi_d \rangle$, *where the* Φ*'s are* $\mathbb{R}$*-valued* (*or more generally* $\mathbb{A}$*-valued where* $\mathbb{A}$ *is a ring*) *is additive if and only if*

$$\Phi_l(x, y) = -\Phi_r(x - 1, y) \quad \textit{and} \quad \Phi_d(x, y) = -\Phi_u(x, y - 1). \tag{4}$$

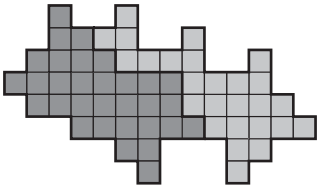


Fig. 4. $\mathbf{P} = \mathbf{P}_1 \cup \mathbf{P}_2$ with disjoint interiors.

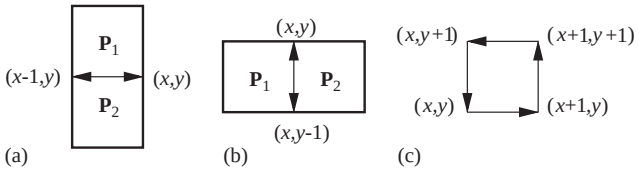


Fig. 5. (a) Vertical domino, (b) horizontal domino, (c) a pixel $\mathrm{Pix}_{x,y}$.

Moreover, the output of an additive incremental algorithm $\bullet$*, on a polyomino* $\mathbf{P}$ *is given by*

$$\mathrm{Output}(\bullet, \mathbf{P}) = \sum_{\mathrm{Pix}_{\alpha,\beta} \subseteq \mathbf{P}} \Delta_x \Phi_u(\alpha, \beta) - \Delta_y \Phi_r(\alpha, \beta), \tag{5}$$

where $\Delta_x \Phi(x, y) = \Phi(x+1, y) - \Phi(x, y)$ *and* $\Delta_y \Phi(x, y) = \Phi(x, y+1) - \Phi(x, y)$.

Proof. Since any polyomino $\mathbf{P}$ can be written as a finite union of the closure $\overline{\mathrm{Pix}_{\alpha,\beta}}$ of its pixels

$$\mathbf{P} = \bigcup_{\mathrm{Pix}_{\alpha,\beta} \subseteq \mathbf{P}} \overline{\mathrm{Pix}_{\alpha,\beta}},$$

the output of an additive incremental algorithm satisfies

$$\mathrm{Output}(\bullet, \mathbf{P}) = \sum_{\mathrm{Pix}_{\alpha,\beta} \subseteq \mathbf{P}} \mathrm{Output}(\bullet, \overline{\mathrm{Pix}_{\alpha,\beta}}).$$

In particular, if $\mathbf{P}_1, \mathbf{P}_2$ are both single pixels and $\mathbf{P}$ is a vertical domino as in Fig. 5(a), then,

$$\mathrm{Output}(\bullet, \mathbf{P}) = \mathrm{Output}(\bullet, \mathbf{P}_1 \cup \mathbf{P}_2) = \mathrm{Output}(\bullet, \mathbf{P}_1) + \mathrm{Output}(\bullet, \mathbf{P}_2).$$

Hence

$$\Phi_l(x, y) = -\Phi_r(x-1, y)$$

in order to cancel the contribution of the common horizontal edge of the domino $\mathbf{P}$. A similar argument (see Fig. 5(b)) shows that using an horizontal domino

$$\Phi_d(x, y) = -\Phi_u(x, y-1).$$

This shows that the stated conditions are necessary for additivity. Their sufficiency follows from the automatic cancellation of the common boundaries of $\mathbf{P}_1$ and $\mathbf{P}_2$ (see Fig. 5)

for general polyominoes with disjoint interiors such that $\mathbf{P} = \mathbf{P}_1 \cup \mathbf{P}_2$. The formula for Output$(\bullet, \mathbf{P})$ also follows from these conditions since for any closed pixel $\overline{\mathrm{Pix}_{\alpha,\beta}}$, we must have (see Fig. 5(c)), for any additive incremental algorithm,

$$\begin{aligned}\mathrm{Output}(\bullet, \overline{\mathrm{Pix}_{\alpha,\beta}}) &= \Phi_r(\alpha,\beta) + \Phi_u(\alpha+1,\beta) + \Phi_l(\alpha+1,\beta+1) + \Phi_d(\alpha,\beta+1)\\ &= \Phi_r(\alpha,\beta) + \Phi_u(\alpha+1,\beta) - \Phi_r(\alpha,\beta+1) - \Phi_u(\alpha,\beta)\\ &= \Delta_x\Phi_u(\alpha,\beta) - \Delta_y\Phi_r(\alpha,\beta). \qquad \square\end{aligned}$$

Proposition 7 may be used for proving, for instance, that a given additive incremental algorithm is actually correct. Indeed, one can check by using it, that the above algorithms for the projection $v_\alpha(\mathbf{P})$ and $h_\beta(\mathbf{P})$ are valid. The validity of the boolean valued additive incremental algorithms in the next sections can also be checked with it. Another use of this proposition is to produce new algorithms starting first from an arbitrary choice of functions $\Phi_r(x, y)$, $\Phi_u(x, y)$; secondly, by defining the associated functions $\Phi_l(x, y)$, $\Phi_d(x, y)$ from (4); and, finally, by computing the corresponding output using (5). See Section 3.4 for such an example.

The next corollary may be considered as an inverse of Proposition 7. It shows how to find $\Phi_r(x, y)$, $\Phi_u(x, y)$, $\Phi_l(x, y)$, $\Phi_d(x, y)$ starting from the desired output. It also describes a close connection between general additive incremental algorithms and the bivariate calculus of finite differences.

Corollary 8. *Let $\mathbb{A}$ be a ring and $W : \mathbb{Z} \times \mathbb{Z} \to \mathbb{A}$ be a weight function associated with each pixel $\mathrm{Pix}_{x,y}$ in the plane. Then, the most general additive incremental algorithm, $\bullet = \langle \Phi_r, \Phi_u, \Phi_l, \Phi_d \rangle$ having the output*

$$\sum_{\mathrm{Pix}_{x,y} \subseteq \mathbf{P}} W(x, y) \in \mathbb{A},$$

for each polyomino $\mathbf{P}$, is of the form

$$\begin{aligned}\Phi_r(x, y) &= V^0(x, y) + \Omega(x, y-1) - \Omega(x-1, y-1),\\ \Phi_u(x, y) &= U^0(x, y) + \Omega(x-1, y) - \Omega(x-1, y-1),\\ \Phi_l(x, y) &= -\Phi_r(x-1, y),\\ \Phi_d(x, y) &= -\Phi_u(x, y-1),\end{aligned}$$

where $(U^0(x, y), V^0(x, y))$ is a particular solution of the difference equation

$$\Delta_x U(x, y) - \Delta_y V(x, y) = W(x, y)$$

and $\Omega : \mathbb{Z} \times \mathbb{Z} \to \mathbb{A}$ is arbitrary.

Proof. Since the difference equation is linear, it is sufficient to show (see Proposition 7) that the general solution of the associated homogeneous equation

$$\Delta_x U(x, y) - \Delta_y V(x, y) = 0 \tag{6}$$

is given by

$$U(x, y) = \Omega(x-1, y) - \Omega(x-1, y-1), \tag{7}$$
$$V(x, y) = \Omega(x, y-1) - \Omega(x-1, y-1), \tag{8}$$

where $\Omega(x, y)$ is arbitrary. Indeed, substituting (7) in (6) gives

$$\begin{aligned}\Delta_x U - \Delta_y V &= [(\Omega(x, y) - \Omega(x, y-1)) - (\Omega(x-1, y) - \Omega(x-1, y-1))] \\ &\quad -[(\Omega(x, y) - \Omega(x-1, y)) - (\Omega(x, y-1) - \Omega(x-1, y-1))] \\ &= 0.\end{aligned}$$

Conversely, in order to show that for any solution (U, V) of the homogeneous equation there corresponds a function Ω, we introduce two auxiliary summation operators, Σ_1^x and Σ_1^y, defined on functions $g : \mathbb{Z} \times \mathbb{Z} \to \mathbb{A}$, by

$$\Sigma_1^x g(x, y) = \begin{cases} \sum_{k=1}^{x} g(k, y) & \text{if } x > 0, \\ 0 & \text{if } x = 0, \\ -\sum_{k=0}^{-x-1} g(-k, y) & \text{if } x < 0, \end{cases}$$

and similarly for $\Sigma_1^y g(x, y)$. The reader can check that, for any function $\omega(x, y)$, we have,

$$\nabla_x \omega(x, y) = \omega(x, y) - \omega(x-1, y) = g(x, y) \iff \omega(x, y) = \omega(0, y) + \Sigma_1^x g(x, y),$$
$$\nabla_y \omega(x, y) = \omega(x, y) - \omega(x, y-1) = g(x, y) \iff \omega(x, y) = \omega(x, 0) + \Sigma_1^y g(x, y),$$

and that the required function $\Omega(x, y)$ can be taken as

$$\Omega(x, y) = c + \Sigma_1^y U(1, y) + \Sigma_1^x V(x, y+1),$$

where c is an arbitrary constant ($c = \Omega(0, 0)$, in fact). □

There exist many ways to find a particular solution (U^0, V^0) of the equation $\Delta_x U - \Delta_y V = W$. One way is to force V^0 (resp. U^0) to be 0 and take U^0 (resp. V^0) to be a particular solution of the simpler difference equation

$$\Delta_x U(x, y) = W(x, y) \quad (\text{resp.} -\Delta_y V(x, y) = W(x, y)),$$

with particular solution

$$U^0 = \Sigma_1^x W(x-1, y), \qquad V^0 = 0 \quad (\text{resp. } U^0 = 0,\ V^0 = -\Sigma_1^y W(x, y-1)).$$

This method provides a particular *V-algorithm* (resp. *H-algorithm*). Formal power series may also be used: let z_1 and z_2 be formal variables and consider the formal Laurent series

$$\widetilde{U}(z_1, z_2) = \sum_{x,y} U(x, y) z_1^x z_2^y,$$
$$\widetilde{V}(z_1, z_2) = \sum_{x,y} V(x, y) z_1^x z_2^y,$$
$$\widetilde{W}(z_1, z_2) = \sum_{x,y} W(x, y) z_1^x z_2^y.$$

Then the difference equation

$$\Delta_x U(x, y) - \Delta_y V(x, y) = W(x, y)$$

rewrites as

$$(1 - z_1)z_2\widetilde{U}(z_1, z_2) - (1 - z_2)z_1\widetilde{V}(z_1, z_2) = z_1 z_2 \widetilde{W}(z_1, z_2),$$

which is solved for $\widetilde{U}(z_1, z_2)$ and $\widetilde{V}(z_1, z_2)$ by using algebraic manipulations. In fact, we used this method to find the general solution of the homogeneous equation appearing in the proof of Corollary 8. Another way to find solutions to the difference equation of Corollary 8 is to express, if possible, $W(x, y)$ in the basis $x^{(i)}y^{(j)}$, $i, j \geqslant 0$, where $t^{(k)} = t(t-1)\ldots(t-k+1)$ is the kth falling factorial power of t. Since $\Delta_t t^{(k)} = kt^{(k-1)}$, this basis is well adapted to difference equations. This method is illustrated in Section 3.4 below for the computation of higher moments of a polyomino.

3.1. Deciding if a polyomino contains a given pixel

Let $(\alpha, \beta) \in \mathbb{Z} \times \mathbb{Z}$ and consider the following boolean-valued function

$$W_{\alpha,\beta}(x, y) = \chi(x = \alpha)\chi(y = \beta).$$

Since

$$\sum_{\mathrm{Pix}_{x,y} \subseteq \mathbf{P}} W_{\alpha,\beta}(x, y) = \chi(\mathrm{Pix}_{\alpha,\beta} \subseteq \mathbf{P}) = \begin{cases} 1 & \text{if } \mathrm{Pix}_{\alpha,\beta} \subseteq \mathbf{P}, \\ 0 & \text{otherwise,} \end{cases}$$

then, the following additive incremental algorithms can be used to decide whether the pixel determined by (α, β) belongs or not to a polyomino $\mathbf{P}$.

V-algorithm: $\Phi_r = 0, \quad \Phi_u = \chi(x \geqslant \alpha + 1)\chi(y = \beta),$
$\Phi_l = 0, \quad \Phi_d = -\chi(x \geqslant \alpha + 1)\chi(y = \beta + 1).$

H-algorithm: $\Phi_r = -\chi(x = \alpha)\chi(y \geqslant \beta + 1), \quad \Phi_u = 0,$
$\Phi_l = \chi(x = \alpha + 1)\chi(y \geqslant \beta + 1), \quad \Phi_d = 0.$

For example, the *V-algorithm* applied to Fig. 6(a) with $(\alpha, \beta) = (3, 2)$ gives (only nonzero terms are listed):

$$\begin{aligned} \chi(\mathrm{Pix}_{3,2} \subseteq \mathbf{P}) &= \chi(x_{14} \geqslant 4)\chi(y_{14} = 2) - \chi(x_{16} \geqslant 4)\chi(y_{16} = 3) \\ &\quad + \chi(x_{26} \geqslant 4)\chi(y_{26} = 2) \\ &= 1 - 1 + 1 = 1 \quad (\text{since, } \mathrm{Pix}_{3,2} \subseteq \mathbf{P}) \end{aligned}$$

and to Fig. 6(b) with $(\alpha, \beta) = (6, 0)$

$$\begin{aligned} \chi(\mathrm{Pix}_{6,0} \subseteq \mathbf{P}) &= -\chi(x_{18} \geqslant 7)\chi(y_{18} = 1) + \chi(x_{24} \geqslant 7)\chi(y_{24} = 0) \\ &= -1 + 1 = 0 \quad (\text{since, } \mathrm{Pix}_{6,0} \not\subseteq \mathbf{P}). \end{aligned}$$

Of course, from Corollary 8, there is an uncountable family of algorithms $\langle \Phi_r^{\alpha,\beta}, \Phi_u^{\alpha,\beta}, \Phi_l^{\alpha,\beta}, \Phi_d^{\alpha,\beta} \rangle$ from which one can compute $\chi(\mathrm{Pix}_{\alpha,\beta} \subseteq \mathbf{P})$.

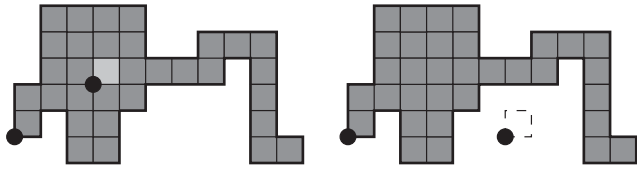


Fig. 6. (a) Pixel $\mathrm{Pix}_{3,2}$ in the polyomino (b) pixel $\mathrm{Pix}_{6,0}$ not in the polyomino.

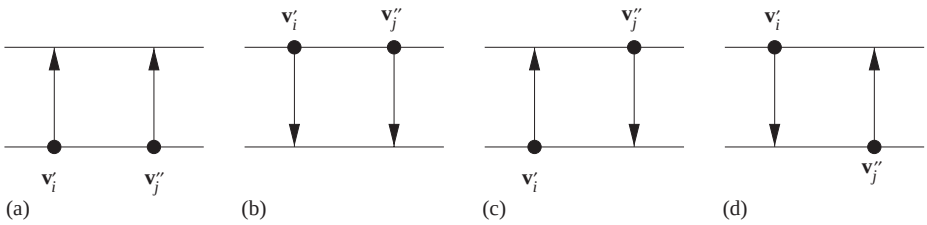


Fig. 7. Matching pairs.

3.2. Boolean operations

From the characteristic function χ, it is now straightforward to define formulas for the boolean operators on polyominoes. However, better results may be achieved with a bit of care. Let $\mathbf{P}_1$ and $\mathbf{P}_2$ be two polyominoes whose contours are given respectively by $\mathbf{v}'_i = (x'_i, y'_i)$, for $i = 0, 1, \ldots, n_1 - 1$, and $\mathbf{v}''_j = (x''_j, y''_j)$ for $j = 0, 1, \ldots, n_2 - 1$.

Proposition 9. *The number of pixels in* $\mathbf{P}_1 \cap \mathbf{P}_2$ *is given by*

$$\#(\mathbf{P}_1 \cap \mathbf{P}_2) = \mathrm{A}(\mathbf{P}_1 \cap \mathbf{P}_2) = \sum_{\substack{0 \leqslant i < n_1, 0 \leqslant j < n_2 \\ i,j, \text{ matching}}} \min(x'_i, x''_j)\Delta y'_i \Delta y''_j,$$

where the sum is extended to all the ordered pairs (i, j) *of indices that match in the following sense*:

$$(i,j) \text{ match} \iff \begin{cases} y'_i = y''_j \quad \text{and} \quad \Delta y'_i = \Delta y''_j \ (= \pm 1) \quad \text{(Fig. 7(a), (b))}, \\ \text{or} \\ y'_i = y''_j - 1 \quad \text{and} \quad \Delta y'_i = 1, \quad \Delta y''_j = -1 \quad \text{(Fig. 7(c))}, \\ \text{or} \\ y'_i = y''_j + 1 \quad \text{and} \quad \Delta y'_i = -1, \quad \Delta y''_j = 1 \quad \text{(Fig. (d))}. \end{cases}$$

Proof. The number of pixels in common between $\mathbf{P}_1$ and $\mathbf{P}_2$ is given by

$$\#(\mathbf{P}_1 \cap \mathbf{P}_2) = \sum_{(p,q) \in \mathbb{Z} \times \mathbb{Z}} \chi(\mathrm{Pix}_{p,q} \subseteq \mathbf{P}_1)\chi(\mathrm{Pix}_{p,q} \subseteq \mathbf{P}_2).$$

Using now the *V-algorithm* described in Section 3.1, we can write

$$\chi(\mathrm{Pix}_{p,q} \subseteq \mathbf{P}) = \sum_i \Phi^{p,q}(\mathbf{v}_i, \Delta\mathbf{v}_i),$$

where $\Phi^{p,q}(\mathbf{v}, \Delta\mathbf{v}) = \chi(x \geqslant p+1)\chi(y = q + (1-\Delta y)/2)\Delta y$, since $\Delta y \in \{-1, 0, 1\}$. Let M (resp. N) be a lower bound for all the x_i' and x_j'' (resp. y_i' and y_j''). Then,

$$\begin{aligned}\#(\mathbf{P}_1 \cap \mathbf{P}_2) &= \sum_{p \geqslant M, q \geqslant N} \sum_{i,j} \Phi^{p,q}(\mathbf{v}_i', \Delta\mathbf{v}_i')\Phi^{p,q}(\mathbf{v}_j'', \Delta\mathbf{v}_j'') \\ &= \sum_{i,j} \Omega(\mathbf{v}_i', \Delta\mathbf{v}_i', \mathbf{v}_j'', \Delta\mathbf{v}_j''),\end{aligned}$$

where

$$\begin{aligned}\Omega(\mathbf{v}_i', \Delta\mathbf{v}_i', \mathbf{v}_j'', \Delta\mathbf{v}_j'') &= \sum_{p \geqslant M, q \geqslant N} \Phi^{p,q}(\mathbf{v}_i', \Delta\mathbf{v}_i')\Phi^{p,q}(\mathbf{v}_j'', \Delta\mathbf{v}_j'') \\ &= \sum_{p \geqslant M} \chi(x_i' \geqslant p+1)\chi(x_j'' \geqslant p+1 \\ &\quad \times \sum_{q \geqslant N} \chi(y_j' = q + (1-\Delta y_j')/2)\chi(y_j'' = q + (1-\Delta y_j'')/2) \\ &= (\min(x_i', x_j'') - M)\theta(y_i', \Delta y_i', y_j'', \Delta y_j'')\end{aligned}$$

and where,

$$\theta(y_i', \Delta y_i', y_j'', \Delta y_j'') = \begin{cases} 1 & \text{if } y_i' = y_j'' \quad \text{and} \quad \Delta y_i' = \Delta y_j''(= \pm 1), \\ -1 & \text{if } y_i',\ y_i' = y_j'' - 1, \quad \Delta y_i' = 1, \quad \Delta y_j'' = -1, \\ -1 & \text{if } y_j'',\ y_i' = y_j'' + 1, \quad \Delta y_i' = -1, \quad \Delta y_j'' = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Hence,

$$\#(\mathbf{P}_1 \cap \mathbf{P}_2) = \sum_{\substack{0 \leqslant i < n_1,\ 0 \leqslant j < n_2 \\ i,j,\ \text{matching}}} (\min(x_i', x_j'') - M)\Delta y_i' \Delta y_j'',$$

and since the left-hand side is independent of M, the result follows by replacing M by $M-1$. □

Using the de Morgan set formulas, the number of pixels in the union and difference of two polyominoes is computed by

$$\begin{aligned}&\#(\mathbf{P}_1 \cup \mathbf{P}_2) = \#(\mathbf{P}_1) + \#(\mathbf{P}_2) - \#(\mathbf{P}_1 \cap \mathbf{P}_2), \\ &\#(\mathbf{P}_1 \setminus \mathbf{P}_2) = \#(\mathbf{P}_1) - \#(\mathbf{P}_1 \cap \mathbf{P}_2).\end{aligned}$$

Intersection between a polyomino and a given set: Let S be a finite or infinite union of pixels and let $\langle \Phi_r^{p,q}, \Phi_u^{p,q}, \Phi_l^{p,q}, \Phi_d^{p,q} \rangle$ be algorithms for the computation of

$$\chi(\mathrm{Pix}_{p,q} \subseteq \mathbf{P}) \quad (p,q) \in \mathbb{Z} \times \mathbb{Z}.$$

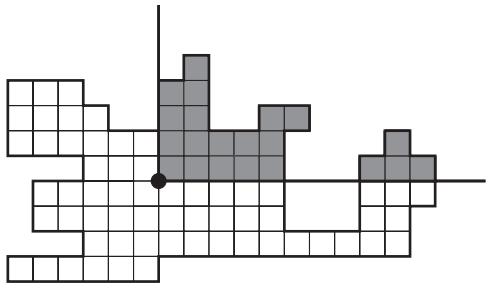


Fig. 8. There are 21 pixels in **P** to the north-east of (α, β).

The number $\#(S \cap \mathbf{P})$ of pixels in common between S and **P** can be computed by taking $\langle \Phi_r^S, \Phi_u^S, \Phi_l^S, \Phi_d^S \rangle$, where

$$\Phi_r^S(x, y) = \sum_{\mathrm{Pix}_{p,q} \subseteq S} \Phi_r^{p,q}(x, y), \quad \Phi_u^S(x, y) = \sum_{\mathrm{Pix}_{p,q} \subseteq S} \Phi_u^{p,q}(x, y),$$
$$\Phi_l^S(x, y) = \sum_{\mathrm{Pix}_{p,q} \subseteq S} \Phi_l^{p,q}(x, y), \quad \Phi_d^S(x, y) = \sum_{\mathrm{Pix}_{p,q} \subseteq S} \Phi_d^{p,q}(x, y).$$

In particular, to decide if a polyomino **P** intersects interior of S, one simply checks if the output of this algorithm is > 0.

Computation of hook-lengths: Consider the north-east corner in the $\mathbb{R} \times \mathbb{R}$ plane associated with a given lattice point $(\alpha, \beta) \in \mathbb{Z} \times \mathbb{Z}$

$$\mathrm{NE}_{\alpha,\beta} = \{(x, y) \in \mathbb{R} \times \mathbb{R} | \alpha \leqslant x, \beta \leqslant y\} = [\alpha, \infty) \times [\beta, \infty).$$

Then the reader can check that the following algorithms can be used to compute, for a polyomino **P**, the number of pixels in $\mathbf{P} \cap \mathrm{NE}_{\alpha,\beta}$. That is, the number of pixels of **P** which are to the north-east of (α, β)(see Fig. 8):

V-algorithm: $\Phi_r = 0, \quad \Phi_u = (x - \alpha)\chi(x \geqslant \alpha + 1)\chi(y \geqslant \beta),$
$\Phi_l = 0, \quad \Phi_d = -(x - \alpha)\chi(x \geqslant \alpha + 1)\chi(y \geqslant \beta + 1).$

H-algorithm: $\Phi_r = -(y - \beta)\chi(x \geqslant \alpha)\chi(y \geqslant \beta + 1), \quad \Phi_u = 0,$
$\Phi_l = (y - \beta)\chi(x \geqslant \alpha + 1)\chi(y \geqslant \beta + 1), \quad \Phi_d = 0.$

Definition 10. Let $(\alpha, \beta) \in \mathbb{Z} \times \mathbb{Z}$ and **P** be a polyomino. The hook-length $\mathrm{hook}_{\alpha,\beta}(\mathbf{P})$ is the number of pixels in the set $\mathbf{P} \cap \mathrm{Hook}_{\alpha,\beta}$ where $\mathrm{Hook}_{\alpha,\beta} = \mathrm{NE}_{\alpha,\beta} \backslash \mathrm{NE}_{\alpha+1,\beta+1}$.

In other words it is simply the number of pixels of **P** which are in the L-shaped hook $\mathrm{Hook}_{\alpha,\beta}$ determined by (α, β) (see Fig. 9).

Replacing (α, β) by $(\alpha + 1, \beta + 1)$ in the above algorithms and subtracting gives corresponding algorithms for the computation of hook-lengths.

V-algorithm: (for the number of pixels in $\mathbf{P} \cap \mathrm{Hook}_{\alpha,\beta}$)

$$\Phi_r = 0, \quad \Phi_l = 0,$$
$$\Phi_u = (x - \alpha)\chi(x \geqslant \alpha + 1)\chi(y \geqslant \beta) - (x - \alpha - 1)\chi(x \geqslant \alpha + 2)\chi(y \geqslant \beta + 1),$$
$$\Phi_d = -(x - \alpha)\chi(x \geqslant \alpha + 1)\chi(y \geqslant \beta + 1) + (x - \alpha - 1)\chi(x \geqslant \alpha + 2)\chi(y \geqslant \beta + 2).$$

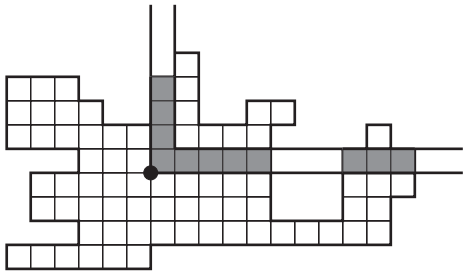


Fig. 9. There are 11 pixels of $\mathbf{P}$ in the $\mathrm{Hook}_{\alpha,\beta}$.

H-algorithm: (for the number of pixels in $\mathbf{P} \cap \mathrm{Hook}_{\alpha,\beta}$)

$$\begin{aligned}
&\Phi_u = 0, \quad \Phi_d = 0,\\
&\Phi_r = -(y-\beta)\chi(x \geqslant \alpha)\chi(y \geqslant \beta+1) + (y-\beta-1)\chi(x \geqslant \alpha+1)\chi(y \geqslant \beta+2),\\
&\Phi_l = (y-\beta)\chi(x \geqslant \alpha+1)\chi(y \geqslant \beta+1) - (y-\beta+1)\chi(x \geqslant \alpha+2)\chi(y \geqslant \beta+2).
\end{aligned}$$

3.3. Computation of higher order moments

Our approach for the computation of higher order moments uses Stirling numbers. It is essentially equivalent to the one given by Yang and Albregsten in [17,18] who uses Bernoulli numbers. It runs as follows.

Consider two integers $m, n \geqslant 0$ and a point $(a, b) \in \mathbb{Z} \times \mathbb{Z}$. By definition, the (m, n)-moment of a polyomino $\mathbf{P}$ relative to the point (a, b) is given by the integral

$$\iint_{\mathbf{P}} (x-a)^m (y-b)^n \,\mathrm{d}x\,\mathrm{d}y.$$

By a simple translation, the computation of such higher order moments can be reduced to central ones:

$$\iint_{\mathbf{P}} x^m y^n \,\mathrm{d}x\,\mathrm{d}y.$$

In this case,

$$\begin{aligned}
W(x, y) = \iint_{\mathbf{P}} x^m y^n \,\mathrm{d}x\,\mathrm{d}y &= \frac{(x+1)^{m+1} - (x)^{m+1}}{m+1} \, \frac{(y+1)^{n+1} - (y)^{n+1}}{n+1}\\
&= \frac{1}{(m+1)(n+1)} \Delta_x x^{m+1} \Delta_y y^{n+1}.
\end{aligned}$$

Now, it is well-known (see [5]) that $t^k = \sum_{v=0}^{k} S_v^k t^{(v)}$, where S_v^k denotes the Stirling numbers of the second kind and $t^{(v)} = t(t-1)\ldots(t-v+1)$. Since $\Delta_t t^{(v)} = v t^{(v-1)}$, it follows that,

$$W(x, y) = \sum_{0 \leqslant i \leqslant m, 0 \leqslant j \leqslant n} w_{i,j} x^{(i)} y^{(j)}, \quad w_{i,j} = \frac{(i+1)(j+1)}{(m+1)(n+1)} S_{i+1}^{m+1} S_{j+1}^{n+1}.$$

To find solutions (U, V) of the difference equation of Corollary 8, let

$$U(x, y) = \sum u_{i,j} x^{(i)} y^{(j)}, \quad V(x, y) = \sum v_{i,j} x^{(i)} y^{(j)}.$$

Then, we have

$$\Delta_x U - \Delta_x V = \sum ((i+1)u_{i+1,j} - (j+1)v_{i,j+1}) x^{(i)} y^{(j)},$$

and the problem is reduced to solve the linear system

$$(i+1)u_{i+1,j} - (j+1)v_{i,j+1} = w_{i,j}, \quad i, j \geqslant 0.$$

Of course, many choices are possible for the $u_{i,j}$'s, $v_{i,j}$'s and the same kind of approach can be used for other $w_{i,j}$'s.

Example. Let $m = 3$, $n = 2$. Then,

$$\begin{aligned} W(x, y) &= \frac{(x+1)^4 - x^4}{4} \frac{(y+1)^3 - y^3}{3} \\ &= \frac{4x^3 + 6x^2 + 4x + 1}{4} \frac{3y^2 + 3y + 1}{3}. \end{aligned}$$

On the other hand, we have,

$$x^3 = x^{(3)} + 3x^{(2)} + x^{(1)}, \quad x^2 = x^{(2)} + x^{(1)}, \quad x = x^{(1)}, \quad y^2 = y^{(2)} + y^{(1)}, \quad y = y^{(1)}.$$

Multiplying, we find,

$$\begin{aligned} W(x, y) &= \frac{4x^{(3)} + 18x^{(2)} + 14x^{(1)} + 1}{4} \frac{3y^{(2)} + 6y^{(1)} + 1}{3} \\ &= x^{(3)} y^{(2)} + 2x^{(3)} y^{(1)} + \frac{1}{3} x^{(3)} + \frac{9}{2} x^{(2)} y^{(2)} + 9x^{(2)} y^{(1)} + \frac{3}{2} x^{(2)} \\ &\quad + \frac{7}{2} x^{(1)} y^{(2)} + 7x^{(1)} y^{(1)} + \frac{7}{6} x^{(1)} + \frac{1}{4} y^{(2)} + \frac{1}{2} y^{(1)} + \frac{1}{12} \\ &= + \cdots + w_{i,j} x^{(i)} y^{(j)} + \cdots \end{aligned}$$

where $w_{i,j} = (i+1)u_{i+1,j} - (j+1)v_{i,j+1}$. For example, taking $v_{i,j} = 0$, for all i, j, we have $w_{i,j} = (i+1)u_{i+1,j}$. Hence, $u_{i+1,j} = w_{i,j}/(i+1)$ with the normalizing condition $u_{0,j} = 0$. In this way we can find all $u_{i,j}$. Then,

$$\begin{aligned} U^0(x, y) &= \sum u_{i,j} x^{(i)} y^{(j)} \\ &= \frac{1}{4} x^{(4)} y^{(2)} + \frac{1}{4} x^{(4)} y^{(1)} + \frac{1}{12} x^{(4)} + \frac{3}{2} x^{(3)} y^{(2)} + 3x^{(3)} y^{(1)} + \frac{1}{3} x^{(3)} \\ &\quad + \frac{7}{2} x^{(2)} y^{(2)} + \frac{7}{2} x^{(2)} y^{(1)} + \frac{7}{12} x^{(2)} + \frac{1}{2} x^{(1)} y^{(1)} + \frac{1}{12} x^{(1)} \\ &= \frac{1}{12} x^{(2)} (x+1)^{(2)} (3y^{(2)} + 3y + 1), \end{aligned}$$

and $V^0 = 0$.

The corresponding values for Φ_d, Φ_h, Φ_g, Φ_b, are obtained by using the formulas of Corollary 8 taking, for example, $\Omega(x, y) = 0$.

3.4. Computation of families of projections

We now give an example where the weights of pixels are taken in the ring $\mathbb{A} = \mathbb{R}((q))$ of formal Laurent power series in q. In analogy to the *V-algorithm* for the area given in Table 1, consider the algorithm associated to the functions

$$\langle \Phi_r(x, y) = 0, \quad \Phi_u(x, y) = [x]_q, \quad \Phi_l(x, y) = 0, \quad \Phi_d(x, y) = -[x]_q \rangle,$$

where

$$[x]_q = \frac{1 - q^x}{1 - q}, \quad x \in \mathbb{Z},$$

denotes the q-analogue of x ($q = 1$ corresponds to area). In this case,

$$\Delta_x \Phi_u = [x + 1]_q - [x]_q = \frac{1 - q^{x+1}}{1 - q} - \frac{1 - q^x}{1 - q} = q^x,$$
$$\Delta_y \Phi_r = 0.$$

So that, in view of Proposition 7, the output of this algorithm on $\mathbf{P}$ is

$$\mathrm{Output}(\bullet, \mathbf{P}) = \sum_{\mathrm{Pix}_{\alpha,\beta} \subseteq \mathbf{P}} \Delta_x \Phi_u(\alpha, \beta) - \Delta_y \Phi_r(\alpha, \beta) = \sum_{\mathrm{Pix}_{\alpha,\beta} \subseteq \mathbf{P}} q^\alpha = \sum_{\alpha \in \mathbb{Z}} v_\alpha(\mathbf{P}) q^\alpha.$$

This is the generating Laurent series of the family of all vertical projections $v_\alpha(\mathbf{P})$, $\alpha \in \mathbb{Z}$, and also a q-analogue of area. A similar approach can be used for the family $h_\beta(\mathbf{P})$, $\beta \in \mathbb{Z}$, of all horizontal projections. Factoring out $(1 - q)$ (resp. $(1 - t)$), the reader can easily check that the following holds:

Corollary 11. *Let q and t be formal variables and $\mathbf{P}$ be a polyomino. Then,*
(a) *for the* V-algorithm $\bullet = \langle \Phi_r = 0, \Phi_u = -q^x, \Phi_l = 0, \Phi_d = q^x \rangle$, *we have*

$$\sum_{\alpha \in \mathbb{Z}} v_\alpha(\mathbf{P}) q^\alpha = \frac{\mathrm{Output}(\bullet, \mathbf{P})}{1 - q},$$

(b) *for the* H-algorithm $\bullet = \langle \Phi_r = t^y, \Phi_u = 0, \Phi_l = -t^y, \Phi_d = 0 \rangle$, *we have*

$$\sum_{\beta \in \mathbb{Z}} h_\alpha(\mathbf{P}) t^\beta = \frac{\mathrm{Output}(\bullet, \mathbf{P})}{1 - t},$$

where $v_\alpha(\mathbf{P})$, $\alpha \in \mathbb{Z}$, and $h_\beta(\mathbf{P})$, $\beta \in \mathbb{Z}$, denote the families of vertical and horizontal projections of the polyomino $\mathbf{P}$.

We illustrate this corollary with the polyomino of Fig. 10.

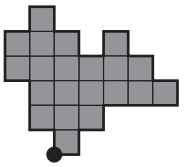


Fig. 10. Family of vertical projections.

The computation using the *V-algorithm* gives

$$\begin{aligned}
&\text{Output}(\bullet, \mathbf{P})\\
&\quad = 0 - q^{x_1} + 0 - q^{x_3} + 0 + 0 + 0 - q^{x_7} + 0 - q^{x_9} + 0 - q^{x_{11}} + 0 + q^{x_{13}} + 0\\
&\qquad -q^{x_{15}} + 0 - q^{x_{17}} + 0 + q^{x_{19}} + 0 + q^{x_{21}} + q^{x_{22}} + 0 + q^{x_{24}} + q^{x_{25}} + 0 + q^{x_{27}}\\
&\quad = -q^1 - q^2 - q^5 - q^4 - q^3 + q^2 - q^1 - q^0 + q^{-1} + q^{-2} + q^{-2}\\
&\qquad + q^{-1} + q^{-1} + q^0\\
&\quad = 2q^{-2} + 3q^{-1} - 2q^1 - q^3 - q^4 - q^5
\end{aligned}$$

and then,

$$\begin{aligned}
\sum_{\alpha\in\mathbb{Z}} v_\alpha(\mathbf{P})q^\alpha = \frac{\text{Output}(\bullet, \mathbf{P})}{1-q} &= \frac{(2q^{-2} + 3q^{-1} - 2q^1 - q^3 - q^4 - q^5)}{(1-q)}\\
&= 2q^{-2} + 5q^{-1} + 5 + 3q + 3q^2 + 2q^3 + q^4,
\end{aligned}$$

where the coefficients of the polynomial correspond to the vertical projections of the polyomino (see Fig. 10).

4. Conclusion

The Discrete Green Theorem provides a general framework allowing the discovery and development of new algorithms for the computation of many statistics on polyominoes. Let us mention, the computation of oblique projections or the computation of various probabilities related to polyominoes. The algorithms described in Corollary 11 or their variants might be of some help for the study of families of polyominoes defined by their projections (see [1,8]).

Computations on integer partitions are obtained along the same lines since partitions are special cases of polyominoes which are encoded by words of the following type $w = r^i\theta d^j$, where θ is a word on $\{u, l\}$ containing i times the letter l and j times the letter u (see Fig. 11).

It should also be possible to study salient and reentrant points on polyominoes in the sense of [6], by extending the concept of incremental algorithm to higher order (where, at

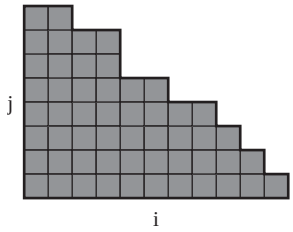


Fig. 11. A partition encoded by $w = rrrrrrrrrrrululululllulluullulddddddddd$.

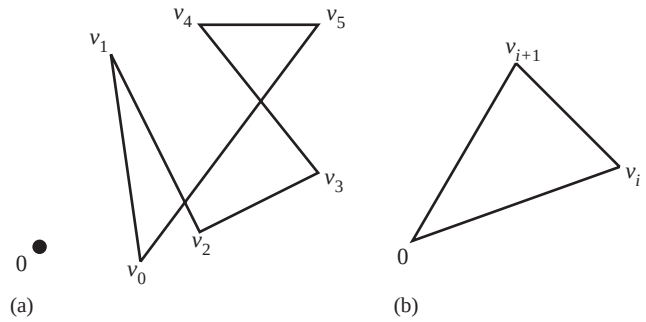


Fig. 12. (a) Closed polygonal path **P**, (b) triangle with oblique segment $[\mathbf{v}_i, \mathbf{v}_{i+1}]$.

each step, the action made depends on the current position on the boundary and on the next k letters read).

Since polyominoes are easily encoded by 4-letter words, we can classify polyominoes according to the value of various parameters by using the appropriate algorithm. For instance, given an integer n, the n-ominoes can be classified according to (weakly) increasing moments of inertia. If two n-ominoes, **P** and **Q** satisfy $\mathrm{I}(\mathbf{P}) \leqslant \mathrm{I}(\mathbf{Q})$, we can say that **P** is rounder than **Q**. We give in the Appendix the output of a simple Maple program implementing some of the algorithms on a polyomino having perimeter 44, and also a classification according to roundness for small n-ominoes, $n \leqslant 5$ (roundest first). It turns out that the roundest n-omino is not necessarily unique.

Note also that their complexity is (time and space) linear in the boundary size of a polyomino: indeed the Freeman chain code of a polyomino is its perimeter, whose size determines the number of iterations in the incremental algorithms. The careful reader has certainly noticed that the algorithms carried out can be straightforwardly adapted to more general objects: for a polyomino with holes it suffices to subtract the holes; needless to say that they can also be extended to planar objects coded by a closed polygonal paths (self-intersecting or not). The alternate proof of Lemma 2 can be adapted to create such algorithms using triangles of the form $\triangle 0\mathbf{v}_i\mathbf{v}_{i+1}$ where the segment $[\mathbf{v}_i, \mathbf{v}_{i+1}]$ can be oblique (see Fig. 12). The resulting algorithms for closed polygonal paths **P** will have

the form

$$\text{Output}(\bullet, \mathbf{P}) = \sum_{i=0}^{n-1} \Phi(\mathbf{v}_i, \Delta\mathbf{v}_i) = \sum_{i=0}^{n-1} \Phi(x_i, y_i, \Delta x_i, \Delta y_i)$$

for suitable functions $\Phi(\mathbf{v}, \Delta\mathbf{v}) = \Phi(x, y, \Delta x, \Delta y)$.

In particular, when the $\Delta\mathbf{v}_i$ are restricted to be in a finite set $\{\delta_1, \delta_2, \ldots, \delta_m\}$, then the corresponding algorithm takes the form

$$\text{Output}(\bullet, \mathbf{P}) = \sum_{i=0}^{n-1} \sum_{j=1}^{m} \Phi_{\delta_j}(x_i, y_i),$$

where the

$$\Phi_{\delta_j}(x_i, y_i) = \begin{cases} \Phi(x_i, y_i, \Delta x_i, \Delta y_i) & \text{if } \Delta\mathbf{v}_i = \delta_j, \\ 0 & \text{otherwise.} \end{cases}$$

For example, algorithms for paths on hexagonal lattices in the complex plane can be analyzed by taking the δ_k's to be the complex 6th roots of unity.

5. Uncited reference

[3].

Acknowledgements

The authors wish to thank the anonymous referees for the very careful reading of the paper and their valuable comments. Part of the material produced here was presented at the DGCI'03 conference held in Napoli (Italia).

Appendix

It is easy to implement in Maple the incremental algorithms developed above. Here is the output of a program on the human-like polyomino (see Fig. 14) described by the word $w = rruuurddrrrulluuuurrrullllluldllulddrrddddddld$:

font=tir at 6pt

Area:	27.
Center of gravity:	[47/18, 239/54], [2.611111111, 4.425925926].
Moment of inertia:	11719/54, 217.0185185.
Vertical projections:	$2q^{-1} + 2 + 7q + 5q^2 + 6q^3 + 2q^4 + 2q^5 + q^6$.
Horizontal projections:	$2 + 4t + 2t^2 + 3t^3 + 3t^4 + 3t^5 + 8t^6 + 2t^7$.

In Fig. 13, we classify n-ominoes according to weakly increasing moment of inertia (for $n = 1, 2, \ldots, 5$). In this figure, $\mathbf{P} \leqslant \mathbf{Q}$ means that $\mathrm{I}(\mathbf{P}) \leqslant \mathrm{I}(\mathbf{Q})$ or, equivalently, that $\mathbf{P}$ is rounder than $\mathbf{Q}$. Equality of roundness is possible for distinct n-ominoes. The roundest n-omino is not always unique (see $n = 5$). We leave open the problem of the explicit geometrical description of the roundest n-omino(es) as a function of n.

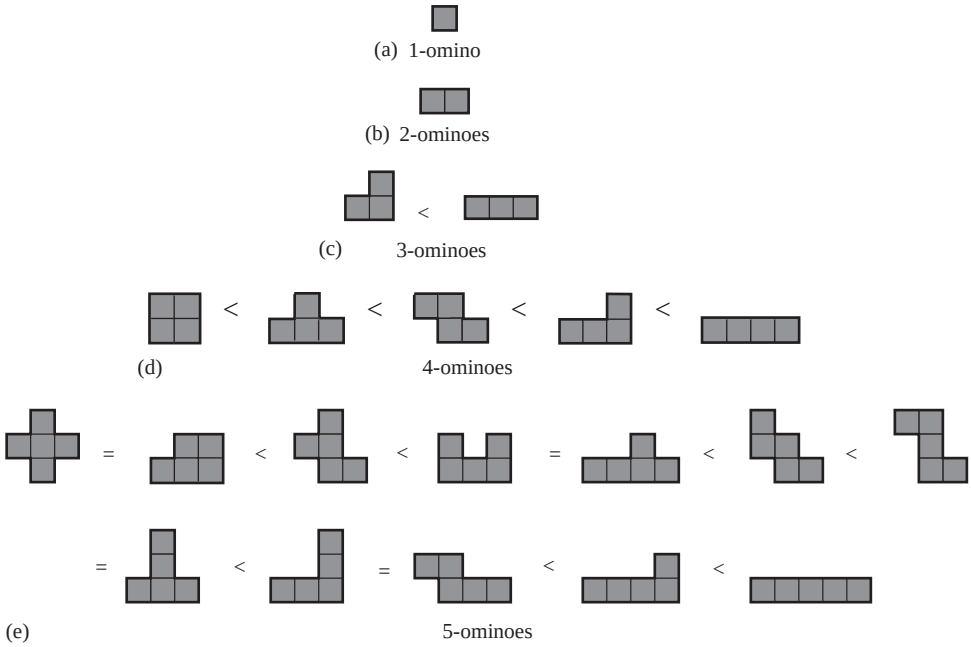


Fig. 13. Polyominoes of given area classified according to decreasing roundness.

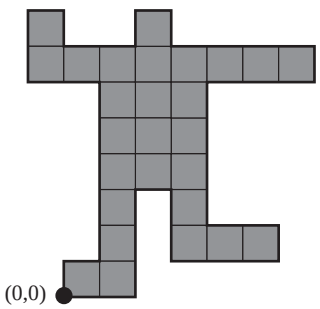


Fig. 14. Human-like polyomino.

References

[1] E. Barcucci, A. Del Lungo, M. Nivat, R. Pinzani, Reconstruction convex polyominoes from horizontal and vertical projections, Theoret. Comput. Sci. 155 (1996) 321–347.
[2] M. Bousquet-Mélou, New enumerative results on two-dimensional directed animals, Discrete Math. 180 (1–3) (1998) 73–106.
[3] S. Brlek, G. Labelle, A. Lacasse, Incremental algorithms based on discrete Green Theorem, in: I. Nyström, G. Sanniti di Baja, S. Svensson (Eds.), Proc. DGCI'03, Naples, Italy, 2003, Lecture Notes in Computer Science, Vol. 2886, Springer, Berlin, 2003, pp. 277–287.
[4] A.L. Clarke, Isometrical polyominoes, J. Recreational Math. 13 (1980) 18–25.

[5] L. Comtet, Advanced Combinatorics, Reidel, Dordrecht, 1974.
[6] A. Daurat, M. Nivat, Salient and reentrant points of discrete sets, in: A. Del Lungo, V. Di Gesù, A. Kuba (Eds.), Proceedings IWCIA'03, Palermo, Italy, 2003, Electronic Notes in Discrete Mathematics, Vol. 12, Elsevier, Amsterdam, 2003.
[7] M.P. Delest, D. Gouyou-Beauchamps, B. Vauquelin, Enumeration of parallelogram polyominoes with given bound and site perimeter, Graphs Combin. 3 (1987) 325–339.
[8] A. Del Lungo, Polyominoes defined by two vectors, Theoret. Comput. Sci. 127 (1) (1994) 187–198.
[9] H. Freeman, On the encoding of arbitrary geometric configurations, IRE Trans. Electron. Comput. 10 (1961) 260–268.
[10] H. Freeman, Boundary encoding and processing, in: B.S. Lipkin, A. Rosenfeld (Eds.), Picture Processing and Psychopictorics, Academic Press, New York, 1970, pp. 241–266.
[11] S.W. Golomb, Polyominoes: Puzzles, Patterns, Problems and Packings, Princeton University Press, Princeton, NJ, 1996.
[12] A. Lacasse, Contribution à l'étude des polyominos, Mémoire de maı̂trise, Université du Québec à Montréal, 2004.
[13] W.G. McCallum, D. Hughes-Hallett, A.M Gleason, et al., Multivariable Calculus, Wiley, Indianapolis, 2002.
[14] W. Philips, A new fast algorithm for moment computation, Pattern Recognition 26 (11) (1993) 1619–1621.
[15] G.Y. Tang, B. Lien, Region filling with the use of the discrete Green theorem, Comput. Vision Graphics Image Process. 42 (1988) 297–305.
[16] X.G. Viennot, A survey of polyomino enumeration, in: P. Leroux, C. Reutenauer (Eds.), Proc. FPSAC'92, Montréal, QC, 1992, Publications du LACIM, Vol. 11, 1992, pp. 399–420.
[17] L. Yang, F. Albregtsen, Fast computation of invariant geometric moments: a new method giving correct results, Proc. ICPR'94, Jerusalem, Israel, 1994, Cedar Publications, Buffalo, 1994, pp. A:201–204.
[18] L. Yang, F. Albregtsen, Fast and exact computation of Cartesian geometric moments using discrete Green's theorem, Pattern Recognition 29 (7) (1996) 1061–1073.