Incremental Algorithms Based on Discrete Green Theorem*

Srečko Brlek, Gilbert Labelle, and Annie Lacasse

Laboratoire de Combinatoire et d'Informatique Mathématique
Université du Québec à Montréal,
CP 8888, Succ. Centre-ville, Montréal (QC) Canada H3C3P8
{brlek,gilbert,lacasse}@lacim.uqam.ca

Abstract. By using the discrete version of Green's theorem and bivariate difference calculus we provide incremental algorithms to compute various statistics about polyominoes given, as input, by 4-letter words describing their contour. These statistics include area, coordinates of the center of gravity, moment of inertia, higher order moments, size of projections, hook lengths, number of pixels in common with a given set of pixels and also q-statistics.

Keywords: Discrete Green Theorem, statistics about polyominoes.

1 Introduction

In this paper, the word polyomino means a finite union of unit lattice closed squares (pixels) in the plane whose boundary consists of a simple closed polygonal path using 4-connectedness. In particular, our polyominoes are simply connected (contain no holes), and have no multiple points.

The polygonal path γ (contour) of a polyomino can be encoded by an ordered pair $(s,\, \omega)$ where s is a lattice point belonging to γ and ω is a word over the 4-letter alphabet

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

known as the Freeman chain code [8,9], corresponding to the unit translations, respectively, in the right, up, left and down direction. The word ω represents the perimeter of the polyomino described in a counterclockwise manner starting from point s. For example, the polyomino of Figure 1 is coded by $(s,\, \omega)$ where $s = (0, 0)$ and $\omega = rdrdrrruuruullluuldldddlld$.

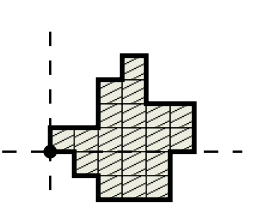
Many basic parameters associated to polyominoes (see Figure 1) can be represented by surface integrals. For example, the area $\mathrm{A}(P)$, center of gravity $\mathrm{CG}(P)$ and moment of inertia $\mathrm{I}(P)$, of a polyomino P are defined by the integrals

$$\mathrm{A}(P) = \iint_P dx\, dy, \quad \mathrm{CG}(P) = (\bar{x}, \bar{y}) = \left(\frac{\iint_P x\, dx\, dy}{\iint_P dx\, dy}, \frac{\iint_P y\, dx\, dy}{\iint_P dx\, dy} \right),$$

$$\mathrm{I(P)} = \iint_P ((x-\bar{x})^2 + (y-\bar{y})^2)\, dx\, dy = \iint_P (x^2+y^2)\, dx\, dy - (\bar{x}^2+\bar{y}^2)\mathrm{A}(P).$$

* With the support of NSERC (Canada).

I. Nyström et al. (Eds.): DGCI 2003, LNCS 2886, pp. 277–287, 2003.



Area= 20
Center of gravity = (3.3, 0.55)
Moment of inertia = 73.4833
Horizontal projections = (3,4,6,4,2,1)
Vertical projections = (1,2,5,6,4,2)

Fig. 1. Some parameters for polyominoes.

The classical Green's Theorem (see below) relates surface integrals to contour integrals. Since our 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 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 to the sum of two polyominoes is the sum of the outputs associated to each polyomino. The use of Green's Theorem is not new in discrete geometry [9]. Our present approach is similar to the one given in [8, 11, 12] where discrete Green's Theorem is applied to efficient moment computations. For a general presentation of polyominoes and their properties see [7]. A survey of enumerative results concerning polyominoes can be found in [10](see also [2, 3, 5]).

2 Green's Theorem and Incremental Algorithms

The following version of Green's Theorem will be sufficient to start our analysis.

Theorem 1. [Green] *Let $P(x,y)$, $Q(x,y)$ be continuously differentiable functions on an open set containing a simply connected region Ω bounded by simple piecewise continuously differentiable positively oriented curve Γ. Then*

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

Since the above parameters about polyominoes involve integrals of the form

$$\iint_{P} f(x,y)\, dx\, dy,$$

our next step is to choose $P(x,y)$ and $Q(x,y)$, in Green's Theorem, such that $\left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}\right) = f$. There are many ways to do this, and we list three important ones in the following Lemma.

Lemma 1 *Let P be a polyomino with contour γ, and let $f = f(x,y)$ be continuous. Then we have,*

$$\iint_{P} f(x,y)\, dx\, dy \overset{(1)}{=} \int_{\gamma} f_1(x,y)\, dy \overset{(2)}{=} -\int_{\gamma} f_2(x,y)\, dx \overset{(3)}{=} \int_{\gamma} F(x,y)(x dy - y dx),$$

where

$$f_1(x,y) = \int^x f(u,y)\,du,\ f_2(x,y) = \int^y f(x,v)\,dv,\ F(x,y) = \int_0^1 f(sx,sy)s\,ds.$$

The notation $\int_\gamma$ *stands for contour integration on* γ *while* $\int^t$ *means the 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 $Q(x,y) = xF(x,y)$. Using some analytical manipulations it can be shown that

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

□

Incremental Algorithms. The evaluation of each line integral in Lemma 1 can be broken into simpler integrals over successive unit (horizontal or vertical) line segments forming γ:

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

where $\boldsymbol{v}_i = (x_i, y_i)$, $i = 0, \ldots, n-1$, denotes the successive vertices of the polyomino P, $\boldsymbol{v}_n = \boldsymbol{v}_0$, $\boldsymbol{v}_{i+1} = \boldsymbol{v}_i + \Delta\boldsymbol{v}_i = (x_i + \Delta x_i, y_i + \Delta y_i)$.

Since our polyominoes are coded by (s, ω) where $s \in \mathbb{Z} \times \mathbb{Z}$ is the starting point and ω is a word over the alphabet $\mathcal{A} = \{r, u, l, d\}$, the above discussion gives rise to incremental algorithms in the following sense: *Starting from the source point* s*, the contour* γ *of the polyomino is described by reading* ω *letter by letter. At each step, the action made depends only on the current position on the boundary and on the letter read.* More precisely, consider four vectors

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

and take four functions (one for each letter in $\mathcal{A}$)

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

Then cumulate sequentially the partial sums on $\omega = \omega_1\omega_2\ldots\omega_n$ as follows:

$\boldsymbol{v} := (x_0, y_0); S := 0;$
for $i = 1$ **to** n **do** $S := S + \Phi_{\omega_i}(\boldsymbol{v}); \boldsymbol{v} := \boldsymbol{v} + \boldsymbol{\omega_i}$ **od**;
return S.

We will use the following suggestive notation to represent the output of our incremental algorithm:

$$\sum_{\rightarrow} \Phi_r(x_i,y_i) + \sum_{\uparrow} \Phi_u(x_i,y_i) + \sum_{\leftarrow} \Phi_l(x_i,y_i) + \sum_{\downarrow} \Phi_d(x_i,y_i).$$

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

Proposition 1 (Incremental Algorithms of Green's Type). *Let P be a polyomino encoded by (s, ω). Then,*

$$\iint_P f(x,y)\,dx\,dy = \sum_{\rightarrow} \Phi_r(x_i,y_i) + \sum_{\uparrow} \Phi_u(x_i,y_i) + \sum_{\leftarrow} \Phi_l(x_i,y_i) + \sum_{\downarrow} \Phi_d(x_i,y_i),$$

where the functions Φ_r, Φ_u, Φ_l, Φ_d are taken from any of the following three sets of possibilities

$$\begin{aligned}
&V\text{:}\ \Phi_r = 0, \quad \Phi_u = \textstyle\int_0^1 f_1(x,y+t)\,dt, \quad \Phi_l = 0, \quad \Phi_d = -\int_0^1 f_1(x,y-t)\,dt.\\
&H\text{:}\ \Phi_r = -\textstyle\int_0^1 f_2(x+t,y)dt, \quad \Phi_u = 0, \quad \Phi_l = \int_0^1 f_2(x-t,y)dt, \quad \Phi_d = 0.\\
&VH\text{:}\ \Phi_r = -y\textstyle\int_0^1 F(x+t,y)dt,\ \Phi_u = x\int_0^1 F(x,y+t)dt,\\
&\qquad \Phi_l = y\textstyle\int_0^1 F(x-t,y)dt,\ \Phi_d = -x\int_0^1 F(x,y-t)dt.
\end{aligned}$$

where $f_1(x,y)$, $f_2(x,y)$ and $F(x,y)$ are defined by Lemma 1.

Elementary instances of these algorithms are given in the following tables for the area (Table 1) where $f(x,y) = 1$, center of gravity (Table 2), where $f(x,y) = x$ and $f(x,y) = y$; and moment of inertia (Table 3), where $f(x,y) = x^2 + y^2$.

Table 1. Area.

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

For instance, using the polyomino $\omega = rrururulluldddldd$, we obtain:
VH-algorithm for the area: $\sum_{\rightarrow} -y_i/2 + \sum_{\uparrow} x_i/2 + \sum_{\leftarrow} y_i/2 + \sum_{\downarrow} -x_i/2$,

$$\begin{aligned}
\iint_P 1\,dx\,dy &= -y_0/2 - y_1/2 + x_2/2 - y_3/2 + x_4/2 - y_5/2 + x_6/2 + y_7/2 + y_8/2\\
&\quad + x_9/2 + y_{10}/2 - x_{11}/2 - x_{12}/2 + y_{13}/2 - x_{14}/2 - x_{15}/2\\
&= 1 - 1/2 + 3/2 - 1 + 2 + 3/2 + 3/2 + 1 + 2 - 1/2 - 1/2 + 1 = 9.
\end{aligned}$$

V-algorithm for $\bar{x}$ of the center of gravity: $\sum_{\rightarrow} 0 + \sum_{\uparrow} x_i^2/2 + \sum_{\leftarrow} 0 + \sum_{\downarrow} -x_i^2/2$.
$\iint_P x\,dx\,dy = x_2^2/2 + x_4^2/2 + x_6^2/2 + x_9^2/2 - x_{11}^2/2 - x_{12}^2/2 - x_{14}^2/2 - x_{15}^2/2 = 31/2.$

V-algorithm for the integral involved in the moment of inertia:
$\iint_P (x^2+y^2)\,dx\,dy = \sum_{\uparrow} \frac{x_i}{3} + x_i y_i + \frac{x_i^3}{3} + x_i y_i^2 + \sum_{\downarrow} -\frac{x_i}{3} + x_i y_i - \frac{x_i^3}{3} - x_i y_i^2 = 92.$

Table 2. Center of gravity.

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

Table 3. Moment of inertia.

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

The next example computes the probabilty that a random point $(x, y) \in \mathbb{R} \times \mathbb{R}$, under a normal bivariate probability distribution, $f(x, y) = \frac{1}{\pi}\exp(-x^2 - y^2)$, falls in a given polyomino P. In this case the *VH-algorithm* is complicated and only the *V* and *H-algorithms* are given (see 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* $f(x, y)$. That is those functions 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 corresponding *VH-algorithm* is described in Corollary 1.

Corollary 1 *Let $f(x, y)$ be a continuous homogeneous function of degree $k > -2$ and let $\Phi_r, \Phi_u, \Phi_l, \Phi_d$ be defined by*

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

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

where $f_1(x, y)$ and $f_2(x, y)$ are defined in Lemma 1. *Then the corresponding additive incremental VH-algorithm computes $\iint_P f(x, y)\,dx\,dy$, for P.*

Here is a typical illustration of Corollary 1 for which the *VH-algorithm* is simpler than the corresponding *V* or *H-algorithms*. 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

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

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

$$\frac{\iint_P \sqrt{(x-a)^2+(y-b)^2}\,dx\,dy}{\mathrm{A}(P)}.$$

This is reducible to the computation of the integral $\iint_P f(x,y)\,dx\,dy$ by simply replacing the starting point $s = (x_0, y_0)$ by $s - (a,b) = (x_0 - a, y_0 - b)$. This corresponds to the choice $f(x,y) = \sqrt{x^2+y^2}$ and $k = 1$ in Corollary 1. In this case, the functions $f_1(x,y)$ and $f_2(x,y)$ are given by the formulas

$$f_1(x,y) = \frac{1}{2}x\sqrt{x^2+y^2} + \frac{1}{2}y^2\ln(x+\sqrt{x^2+y^2}),$$

$$f_2(x,y) = \frac{1}{2}y\sqrt{x^2+y^2} + \frac{1}{2}x^2\ln(y+\sqrt{x^2+y^2}).$$

3 Additive Incremental Algorithms

In the above examples, the function $f = f(x,y)$ was assumed to be continuous. We can often drop this condition on f and still use Proposition 1 as a guideline to devise corresponding algorithms. For example, algorithms for the computation of horizontal and vertical projections of a polyomino can be found in this way: take an integer α and define $f(x,y) = \chi(\alpha \le x < \alpha + 1)$, where χ denotes the characteristic function (which takes the value 1 if the inequations are satisfied, and 0 otherwise). Then, obviously, $\iint_P f(x,y)\,dx\,dy$ is the α-vertical projection of the polyomino P:

$$\iint_P f(x,y)\,dx\,dy = \#\{\beta \in \mathbb{Z} \mid pix_{\alpha,\beta} \subseteq P\} = v_\alpha(P),$$

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

$$pix_{\alpha,\beta} = \{(x,y) \in \mathbb{R}\times\mathbb{R} \mid \alpha \le x < \alpha+1,\ \beta \le y < \beta+1\}.$$

In this case, using Proposition 1, we find that

$$f_1(x,y) = \int^x \chi(\alpha \le u < \alpha+1)\,du = \begin{cases} 0 & \text{if } x < \alpha; \\ x-\alpha & \text{if } \alpha \le x < \alpha+1; \\ 1 & \text{if } \alpha+1 \le x. \end{cases}$$

This gives the following *V-algorithm for the vertical projection* $v_\alpha(P)$:

$$\Phi_r = 0, \quad \Phi_u = \mathcal{X}(x \ge \alpha+1), \quad \Phi_l = 0, \quad \Phi_d = -\mathcal{X}(x \ge \alpha+1).$$

Similarly, taking $f(x,y) = \chi(\beta \leq y < \beta + 1)$, the β-horizontal projection of the polyomino P defined by

$$\#\{\alpha \in \mathbb{Z} \mid pix_{\alpha,\beta} \subseteq P\} = h_\beta(P),$$

can be computed by the *H-Algorithm for the horizontal projection* $h_\beta(P)$:

$$\Phi_r = -\mathcal{X}(y \geq \beta + 1), \quad \Phi_u = 0, \quad \Phi_l = \mathcal{X}(y \geq \beta + 1), \quad \Phi_d = 0.$$

These algorithms for the projections are special instances of the general notion of additive incremental algorithm defined as follows.

Definition 1 *An incremental algorithm* $\Phi_r(x,y)$, $\Phi_u(x,y)$, $\Phi_l(x,y)$, $\Phi_d(x,y)$, *is called additive if, whenever* P *is the union of two polyominoes* P_1, P_2 *with disjoint interiors, we have*

$$\text{output}(P) = \text{output}(P_1 \cup P_2) = \text{output}(P_1) + \text{output}(P_2).$$

An example of a non additive incremental algorithm is given by $\Phi_r = \Phi_u = \Phi_l = \Phi_d = 1$ which simply computes the perimeter of a polyomino.

Proposition 2 *An incremental algorithm* $\Phi_r(x,y)$, $\Phi_u(x,y)$, $\Phi_l(x,y)$, $\Phi_d(x,y)$, *is additive if and only if*

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

Moreover the output of an additive incremental algorithm on P *is given by*

$$\text{output}(P) = \sum_{pix_{\alpha,\beta} \subseteq P} \Delta_x \Phi_u(\alpha,\beta) - \Delta_y \Phi_r(\alpha,\beta), \tag{1}$$

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. (Sketch) The main idea is to reduce the analysis to the case where the polyomino is a (horizontal or vertical) domino, where the sum cancels over the common edge. □

Proposition 2 can be used, for example, to prove rigourously that a given additive incremental algorithm actually works. For example, the reader can check, using it, that the above algorithms for the projection $v_\alpha(P)$ and $h_\beta(P)$ are valid. The validity of the boolean valued additive incremental algorithms below can also be checked using Proposition 2. Another use of this proposition is to create new algorithms starting first from an arbitrary choice of functions $\Phi_r(x,y)$, $\Phi_u(x,y)$; then by defining the associated functions $\Phi_l(x,y)$, $\Phi_d(x,y)$; and, finally, by computing the corresponding output.

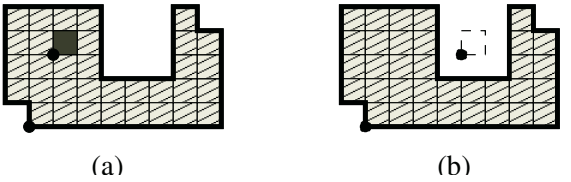


Fig. 2. (a) Pixel $pix_{1,3}$ in the polyomino (b) pixel $pix_{4,3}$ not in the polyomino.

Deciding if a Polyomino Contains a Given Pixel. Let $(\alpha, \beta) \in \mathbb{Z} \times \mathbb{Z}$ and consider the boolean-valued function $W_{\alpha,\beta}(x, y) = \chi(x = \alpha)\chi(y = \beta)$. Since

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

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

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

For example, the *V-algorithm* applied to Figure 2(a) with $(\alpha, \beta) = (1, 3)$ and to Figure 2(b) with $(\alpha, \beta) = (4, 3)$ gives respectively (only non-zero terms listed):

$$\begin{aligned} \chi(pix_{1,3} \subseteq P) &= \chi(x_{11} \geq 2)\chi(y_{11} = 3) - \chi(x_{16} \geq 2)\chi(y_{16} = 4) \\ &\quad + \chi(x_{22} \geq 3)\chi(y_{22} = 3) \\ &= 1 - 1 + 1 = 1 \quad (\text{since } pix_{1,3} \subseteq P); \\ \chi(pix_{4,3} \subseteq P) &= \chi(x_{11} \geq 2)\chi(y_{11} = 3) - \chi(x_{16} \geq 2)\chi(y_{16} = 4) \\ &= 1 - 1 = 0 \quad (\text{since } pix_{4,3} \not\subseteq P). \end{aligned}$$

Of course there is an uncountable family of algorithms $\Phi_r^{\alpha,\beta}$, $\Phi_u^{\alpha,\beta}$, $\Phi_l^{\alpha,\beta}$, $\Phi_d^{\alpha,\beta}$ from which one can compute $\chi(pix_{\alpha,\beta} \subseteq P)$.

Pixels in Common between a Polyomino and a Given Set. Let S be a set of pixels and let $\Phi_r^{p,q}$, $\Phi_u^{p,q}$, $\Phi_l^{p,q}$, $\Phi_d^{p,q}$ be an algorithm for the computation of

$$\chi(pix_{p,q} \subseteq P), \quad (p, q) \in \mathbb{Z} \times \mathbb{Z}.$$

Then, to decide if a polyomino P intersects S, one must compute $\chi(S \cap P \neq \emptyset)$. This can obviously be done by taking Φ_r^S, Φ_u^S, Φ_l^S, Φ_d^S, where

$$\Phi_r^S(x, y) = \sup_{pix_{p,q} \subseteq S} \Phi_r^{p,q}(x, y), \quad \Phi_u^S(x, y) = \sup_{pix_{p,q} \subseteq S} \Phi_u^{p,q}(x, y),$$
$$\Phi_l^S(x, y) = \sup_{pix_{p,q} \subseteq S} \Phi_l^{p,q}(x, y), \quad \Phi_d^S(x, y) = \sup_{pix_{p,q} \subseteq S} \Phi_d^{p,q}(x, y).$$

To compute the number $\#(S \cap P)$ of pixels in common between S and P, simply replace in the last algorithm the sup symbols by summation symbols $\sum$.

Computation of Hook-Lengths. Consider the north-east corner in the $\mathbb{R} \times \mathbb{R}$ plane associated to a given lattice point $(\alpha, \beta) \in \mathbb{Z} \times \mathbb{Z}$

$$NE_{\alpha,\beta} = \{(x, y) \in \mathbb{R} \times \mathbb{R} \mid \alpha \leq x, \beta \leq y\} = [\alpha, \infty) \times [\beta, \infty).$$

Then the reader can check that the following algorithms can be used to compute, for a polyomino P, the number of pixels in $P \cap NE_{\alpha,\beta}$, i.e., the number of pixels of P which are to the north-east of (α, β) (see Figure 3):

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

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

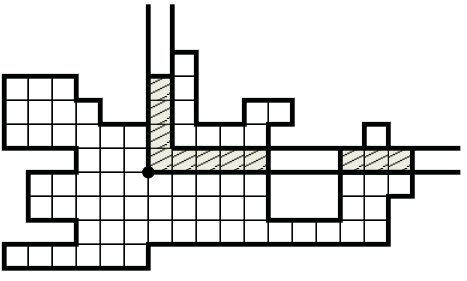


Fig. 3. There are 21 pixels in P to the north-east of (α, β), and 11 pixels in the $Hook_{\alpha,\beta}$.

Let $(\alpha, \beta) \in \mathbb{Z} \times \mathbb{Z}$ and P be a polyomino. The *hook-length* of $(\alpha, \beta) \in P$ is $\text{hook}_{\alpha,\beta}(P) = \#(P \cap \text{Hook}_{\alpha,\beta})$ where $\text{Hook}_{\alpha,\beta} = NE_{\alpha,\beta} \setminus NE_{\alpha+1,\beta+1}$. In other words, it is the number of pixels of P belonging to the L-shaped $\text{Hook}_{\alpha,\beta}$ determined by (α, β) (see Figure 3).

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

Computation of Higher Order Moments. Our approach for the computation of higher order moments is equivalent to the one given by Yang and Albregsten in [11, 12] and differ because we use Stirling instead of Bernoulli numbers. For sake of completeness, we provide it in our framework. Consider two integers $m, n \geq 0$ and a point $(a, b) \in \mathbb{Z} \times \mathbb{Z}$. By definition, the (m, n)-moment of a polyomino P relative to the point (a, b) is given by the following integrals

$$\iint_P (x - a)^m (y - b)^n \, dx \, dy = \iint_P x^m y^n \, dx \, dy,$$

where the second is obtained by a simple translation. In this case,

$$W(x,y) = \int\!\!\int_P x^m y^n \, dx \, dy = \frac{(x+1)^{m+1} - (x)^{m+1}}{m+1} \cdot \frac{(y+1)^{n+1} - (y)^{n+1}}{n+1}$$
$$= \frac{1}{(m+1)(n+1)} \Delta_x x^{m+1} \Delta_y y^{n+1}.$$

Now, it is well-known (see [4]) 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 is easily seen that,

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

To find solutions $(U,V) = (\Phi_u, \Phi_r)$ of the difference equation (1), 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,

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

and the problem is reduced to solve the linear system

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

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.

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 also mention, the simultaneous computation of vertical projections of a polyomino P: setting $\Phi_r(x,y) = 0$, $\Phi_u(x,y) = q^x$, where q is a formal variable, the coefficients of $\sum_{\alpha \in \mathbb{Z}} v_\alpha(P) q^\alpha = -\frac{\text{output}(P)}{1-q}$ are the vertical projections (horizontal or oblique projections are obtained in a similar way). This might be of some help for the study of families of polyominoes defined by their projections (see [1, 6]). Computations on integer partitions are obtained along the same lines since partitions are special cases of polyominoes which are encoded by words of the type $\omega = r^i \theta d^j$, where θ is a word on $\{u,l\}$ containing i times the letter l and j times the letter u.

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 substract the holes; needless to say that it also extends to objects coded by a closed curve. The lack of space permits to show only a small part of the results of this method. For the detailed proofs, discussion, as well as other features not presented here, the reader is referred to the research report [3] that can be obtained from the authors on special request.

Acknowledgements

The authors wish to thank the anonymous referees for the valuable comments that improved greatly the paper readability.

References

1. Barcucci, E., Del Lungo, A., Nivat, M., Pinzani, R.: Reconstructing convex polyominoes from their vertical and horizontal projections, Theoret. Comput. Sci., **155** (1996) 321–347
2. Bousquet-Mélou, M.: New enumerative results on two-dimensional directed animals, Discrete Math. **180** (1-3) (1998) 73–106
3. Brlek, S., Labelle, G., Lacasse, A.: Incremental Algorithms for Polyominoes Coded by their Contour, Research Report, Lacim (Un. Quebec à Montréal) (2003)
4. Clarke, A. L.: Isometrical polyominoes. J. Recreational Math. **13** (1980) 18–25
5. Comtet, L.: Advanced Combinatorics. Reidel (1974)
6. Delest, M. P., Gouyou-Beauchamps, D., Vauquelin, B.: Enumeration of parallelogram polyominoes with given bound and site perimeter, Graphs Comb. **3** (1987) 325–339
7. Del Lungo, A.: Polyominoes defined by two vectors. Theoret. Comput. Sci. **127** (1) (1994) 187–198
8. Freeman, H.: On the Encoding of Arbitrary Geometric Configurations, IRE Trans. Electronic Computer **10** (1961) 260–268
9. Freeman, H.: Boundary encoding and processing, in Picture Processing and Psychopictorics, B.S. Lipkin and A. Rosenfeld, Editors. Academic Press: New York. (1970) 241-266.
10. Golomb, S. W.: Polyominoes: Puzzles, Patterns, Problems, and Packings. Princeton University Press (1996)
11. Philips, W.: A new fast algorithm for moment computation: Pattern Recognition. **26**(11), (1993) 1619–1621
12. Tang, G.Y., Lien, B.: Region Filling With The Use Of The Discrete Green Theorem. Proc. CVGIP(42) (1988) 297–305
13. Viennot, X. G.: A survey of polyomino enumeration, Proc. Séries formelles et combinatoire algébrique, Montréal, Juin 1992. Publications de LACIM 11, Université du Québec à Montréal (1996)
14. Yang, L., Albregtsen, F.: Fast computation of invariant geometric moments. A new method giving correct results. In Proceeding of the International Conference on Pattern Recognition (ICPR'94) (1994) A:201–204
15. Yang, L., Albregtsen, F.: Fast and exact computation of Cartesian geometric moments using discrete Green's theorem. Pattern Recognition. bf 29 No. 7 (1996) 1061–1073