JPL PUBLICATION 79-49

Computing Region Moments from Boundary Representations

J. M. Wilf
R. T. Cunningham

(NASA-CR-162685) COMPUTING REGION MOMENTS FROM BOUNDARY REPRESENTATIONS (Jet Propulsion Lab.) 36 p HC A03/MF A01 CSCL 09B

N80-16767

Unclas
G3/61 46966

November 1, 1979

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL PUBLICATION 79-49

Computing Region Moments from Boundary Representations

J. M. Wilf
R. T. Cunningham

November 1, 1979

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

CONTENTS

Tables

Figures

PRECEDING PAGE BLANK NOT FILMED

ABSTRACT

The moments of a region in an image can be used to describe the region's location, orientation, and shape. This paper derives the class of all possible formulas for computing arbitrary moments of a region from the region's boundary. The selection of a particular formula depends on the choice of an independent parameter. Several choices of this parameter are explored for region boundaries approximated by polygons. The parameter choice that minimizes computation time for boundaries represented by chain code is derived. Finally, two algorithms are presented. The first computes arbitrary moments for a region from a polygonal approximation of its boundary. The second algorithm is optimal for computing low order moments from chain-encoded boundaries.

I. INTRODUCTION

Many pattern recognition techniques in computer vision use structural or statistical features of regions and their outlines to characterize the shape of objects being viewed. An important class of statistical features is the set of $(p+q)$-th order moments defined by

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x,y)\, dxdy \tag{1}$$

where $f(x,y)$ is a density distribution function. In the context of pattern recognition, the moments are computed for a uniform density distribution over a closed region R in the xy-plane. Therefore, $f(x,y)$ reduces to

$$f(x,y) = \begin{cases} 1, & \text{if } (x,y) \in R \\ 0, & \text{if } (x,y) \notin R \end{cases}$$

In this case, (1) becomes

$$m_{pq} = \iint_R x^p y^q \, dxdy \tag{2}$$

The most common example of the use of moments is to compute the centroid $(\bar{x},\bar{y})$ of a region by

$$\bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}} \tag{3}$$

where m_{00} is the area of the region. Moments are also used to compute the angle of a region's axis of minimum moment of inertia Θ. This quantity defines the region's orientation within a two-fold degeneracy and is given by

$$\theta = \frac{1}{2}\tan^{-1}\left[\frac{2(m_{00}m_{11} - m_{10}m_{01})}{(m_{00}m_{20} - m_{10}^2) - (m_{00}m_{02} - m_{01}^2)}\right] \tag{4}$$

The quantities $\bar{x}$, $\bar{y}$, and θ are useful for pattern recognition because they specify the position and orientation of regions, defining a transformation between image and model coordinates, and thus allowing further analysis of shape features in a standard reference frame.

A more direct application of moments to pattern recognition is the use of moment invariants to describe objects. These are quantities, computed in terms of the moments of a region, that are invariant under translation, rotation, and scale changes. Hu [1] derives seven moment invariants and Wong and Hall [2] describe an application that uses them to recognize objects in aerial scenes.

If the region is scanned in a raster fashion, the moments may be calculated by using the discrete version of Eq. (2):

$$m_{pq} = \sum_x \sum_y x^p y^q \quad , \quad (x,y) \varepsilon\ R \tag{5}$$

However, a region is often represented by its boundary. This is the case when edge detection is used to separate objects in a scene [3]. It is possible to reconstruct the region from its boundary and use Eq. (5). However, this is computationally inefficient, since a region usually contains many more points than its boundary. Therefore, a method that computes moments while traversing the boundary is desirable.

In this paper, such a method is derived for boundaries approximated by polygons.

Chain code is an ordered list of numbers that represent the orientations of segments comprising the boundary. It is a commonly used special case of polygonal approximation with many attractive properties [4]. The method presented in this paper is used to find an algorithm that minimizes the time it takes to calculate moments from chain-encoded boundaries.

In Section II, we derive the general formula for computing region moments from a boundary representation. The initial steps of the derivation rely on the theory of differential forms. Readers unfamiliar with this branch of mathematics are referred to Flanders [5]. In Section III, we explore the possible choices of the independent parameter in the general formula to obtain equations suitable for computation. In Section IV, the execution time of these equations is analyzed for chain-encoded curves. Finally, Section V presents an algorithm that computes arbitrary moments from polygonal boundaries and an algorithm that is optimal for computing low order moments from chain-encoded curves.

II. DERIVATION OF THE GENERAL FORMULA

The strategy of this derivation is to reduce the surface integral in formula (2) to a line integral, using Stokes' theorem. In our case, we want to find functions $A = A(x,y)$ and $B = B(x,y)$ such that

$$m_{pq} = \int_R x^p y^q dxdy = \int_{\partial R} Adx + Bdy \tag{6}$$

where ∂R is the boundary of R. Stokes' theorem states that for Eq. (6) to hold, we must have

$$x^p y^q dxdy = d(Adx + Bdy) \tag{7}$$

where d, the differential operator, is defined in two dimensions by $d = dx \frac{\partial}{\partial x} + dy \frac{\partial}{\partial y}$ [5]. Evaluating the d operator in Eq. (7) gives

$$x^p y^q dxdy = \left(\frac{\partial B}{\partial x} - \frac{\partial A}{\partial y}\right) dxdy$$

or (8)

$$x^p y^q = \frac{\partial B}{\partial x} - \frac{\partial A}{\partial y}$$

The solution to this partial differential equation is

$$A = ax^p y^{q+1} \quad , \quad B = bx^{p+1} y^q \tag{9}$$

where a and b are real numbers. We can find a and b by substituting the values of A and B from Eq. (9) back into Eq. (8). This yields the following constraint:

$$b(p + 1) - a(q + 1) = 1 \tag{10}$$

The line integral to be solved then becomes

$$m_{pq} = \int_{\partial R} (aydx + bxdy)x^p y^q \tag{11}$$

where a and b are constrained by Eq. (10). For any piecewise continuous boundary representation, Eq. (11) will give a region's moments from its boundary.

Let ∂R, the boundary of the region, be approximated by a closed polygon as in Fig. 1. Then, $\partial R = \bigcup_{\ell=1}^{n} S_\ell$, where the S_ℓ are linked oriented line segments with endpoints $(x_{\ell-1}, y_{\ell-1})$ and (x_ℓ, y_ℓ), respectively. Since ∂R is a piecewise continuous function, the integral in equation (11) can be broken into a sum of integrals over each boundary segment. Let

$$m_{pq\ell} = \int_{S_\ell} (aydx + bxdy)x^p y^q \tag{12}$$

Then

$$m_{pq} = \sum_{\ell=1}^{n} m_{pq\ell} \tag{13}$$

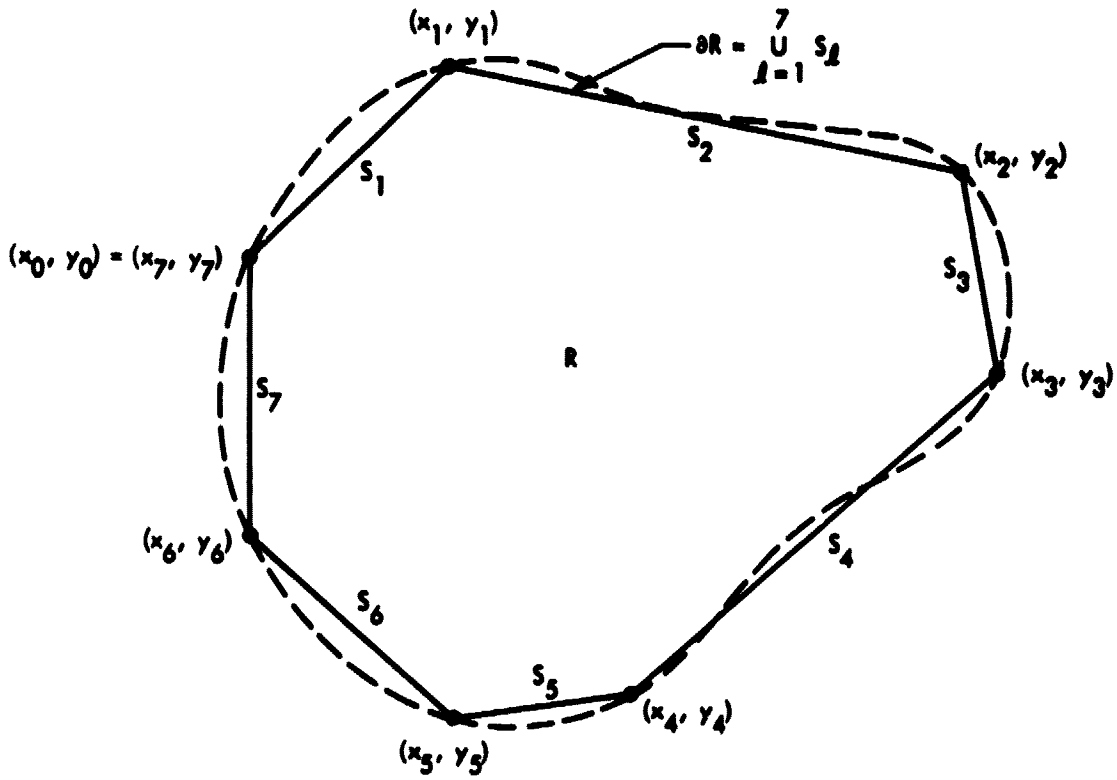

Figure 1. The region R, shown here as bordered by the dotted line, has its boundary, ∂R, approximated by an oriented polygon. The polygon is composed of the links S_ℓ and oriented in a clockwise direction.

The problem is to find $m_{pq\ell}$ by integrating Eq. (12). Let $\Delta x_\ell = x_\ell - x_{\ell-1}$ and $\Delta y_\ell = y_\ell - y_{\ell-1}$. Then all points (x,y) along the line segment S_ℓ can be parameterized as follows:

$$x = \Delta x_\ell t + x_\ell, \qquad y = \Delta y_\ell t + y_\ell \qquad -1 \le t \le 0 \tag{14}$$

$$dx = \Delta x_\ell dt, \qquad dy = \Delta y_\ell dt$$

Using Eq. (14) and the binomial theorem,

$$x^p y^q = \left[\sum_{i=0}^{p} \binom{p}{i} \Delta x_\ell^i t^i \, x_\ell^{p-i}\right] \left[\sum_{j=0}^{q} \binom{q}{j} \Delta y_\ell^j t^j \, y_\ell^{q-j}\right] =$$

$$\sum_{i=0}^{p} \sum_{j=0}^{q} \binom{p}{i} \binom{q}{j} x_\ell^{p-i} \, y_\ell^{q-j} \, \Delta x_\ell^i \, \Delta y_\ell^j t^{i+j} \tag{15}$$

Finally, by substituting the parameterized x, y, dx, dy, and $x^p y^q$ of Eqs. (14) and (15) into the integral that defines $m_{pq\ell}$ in Eq. (12), the general formula is obtained:

$$m_{pq\ell} = \int_{t=-1}^{0} \left[(a+b) \, \Delta x_\ell \, \Delta y_\ell t + a y_\ell \, \Delta x_\ell + b x_\ell \, \Delta y_\ell\right] \left[\sum_{i=0}^{p} \sum_{j=0}^{q} \binom{p}{i} \binom{q}{j} x_\ell^{p-i} \, y_\ell^{q-j} \, \Delta x_\ell^i \, \Delta y_\ell^j \, t^{i+j}\right] dt \tag{16}$$

where $b(p+1) - a(q+1) = 1$. Equation (16) is not only a formula for the $(p+q)$-th order moment; it also contains <u>every possible formula</u> for the $(p+q)$-th order moment. Each choice of parameters a and b, subject to the constraint given above, generates a new valid equation for $m_{pq\ell}$.

III. VALUES FOR THE INDEPENDENT PARAMETER

Before Eq. (16) can be evaluated, real values must be assigned to the parameters a and b. The goal is to choose the a and b that will make computing $m_{pq\ell}$ as simple as possible. The most obvious simplification is elimination of one or more of the terms in the integral. It follows directly from Eq. (16) and the constraint on a and b, that only three parameter choices will eliminate one of the terms in Eq. (16):

$$\begin{aligned} &\text{Choice 1:} \quad a + b = 0,\ b = 1/(p+q+2) \\ &\text{Choice 2:} \quad a = 0,\ b = 1/(p+1) \\ &\text{Choice 3:} \quad b = 0,\ a = -1/(q+1) \end{aligned} \tag{17}$$

We now look at each case in detail. For choice 1, the general formula reduces to

$$m_{pq\ell} = b(x_\ell \Delta y_\ell - y_\ell \Delta x_\ell) \int_{t=-1}^{0} \left[\sum_{i=0}^{p} \sum_{j=0}^{q} \binom{p}{i} \binom{q}{j} x_\ell^{p-i}\ y_\ell^{q-j}\ \Delta x_\ell^{i}\ \Delta y_\ell^{j}\ t^{i+j} \right] dt \tag{18}$$

Noting that

$$\int_{-1}^{0} t^{i+j}\, dt = \left. \frac{t^{i+j+1}}{i+j+1} \right|_{-1}^{0} = - \frac{(-1)^{i+j+1}}{i+j+1} = \frac{(-1)^{i+j}}{i+j+1}.$$

and setting $A_\ell = x_\ell \Delta y_\ell - y_\ell \Delta x_\ell$, the formula for choice 1 becomes

$$m_{pq\ell} = b\, A_\ell \sum_{i=0}^{p} \sum_{j=0}^{q} \frac{(-1)^{i+j}}{i+j+1} \binom{p}{i} \binom{q}{j} x_\ell^{p-i}\, y_\ell^{q-j}\, \Delta x_\ell^{i}\, \Delta y_\ell^{j} \tag{19}$$

For choice 2, where a = 0 and b = 1/(p+1), the middle term in the integral in Eq. (16) drops out, leaving

$$m_{pq\ell} = b \int_{t=-1}^{0} (\Delta x_\ell\, \Delta y_\ell t + x_\ell\, \Delta y_\ell) \left[\sum_{i=0}^{p} \sum_{j=0}^{q} \binom{p}{i} \binom{q}{j} x_\ell^{p-i}\, y_\ell^{q-j}\, \Delta x_\ell^{i}\, \Delta y_\ell^{j}\, t^{i+j} \right] dt \tag{20}$$

Evaluating the integral in (20) gives

$$m_{pq\ell} = b \left[\sum_{i=0}^{p} \sum_{j=0}^{q} \frac{(-1)^{i+j+1}}{i+j+2} \binom{p}{i} \binom{q}{j} x_\ell^{p-i}\, y_\ell^{q-j}\, \Delta x_\ell^{i+1}\, \Delta y_\ell^{j+1} \right.$$

$$\left. + \sum_{i=0}^{p} \sum_{j=0}^{q} \frac{(-1)^{i+j}}{1+j+1} \binom{p}{i} \binom{q}{j} x_\ell^{p-i+1}\, y_\ell^{q-j}\, \Delta x_\ell^{i}\, \Delta y_\ell^{j+1} \right] \tag{21}$$

The two double sums can be combined by making the substitution k=i+1 in the first double sum. Making this substitution yields

$$m_{pq\ell} = b\left[\sum_{k=1}^{p+1}\sum_{j=0}^{q}\frac{(-1)^{k+j}}{k+j+1}\binom{p}{k-1}\binom{q}{j}x_\ell^{p-k+1}\,y_\ell^{q-j}\,\Delta x_\ell^{k}\cdot\Delta y_\ell^{j+1}\right.$$

$$\left.+\sum_{i=0}^{p}\sum_{j=0}^{q}\frac{(-1)^{i+j}}{i+j+1}\binom{p}{i}\binom{q}{j}x_\ell^{p-i+1}\,y_\ell^{q-j}\,\Delta x_\ell^{i}\,\Delta y_\ell^{j+1}\right] \tag{22}$$

Finally, using the relation $\binom{p}{k-1} + \binom{p}{k} = \binom{p+1}{k}$, the sums can be combined as follows:

$$m_{pq\ell} = b\sum_{i=0}^{p+1}\sum_{j=0}^{q}\frac{(-1)^{i+j}}{i+j+1}\binom{p+1}{i}\binom{q}{j}x_\ell^{p-i+1}\,y_\ell^{q-j}\,\Delta x_\ell^{i}\,\Delta y_\ell^{j+1} \tag{23}$$

Choice 3, the case where $b = 0$ and $a = -1/(q+1)$, reduces Eq. (16) to

$$m_{pq\ell} = a\int_{t=-1}^{0}(\Delta x_\ell\,\Delta y_\ell t + y_\ell\,\Delta x_\ell)\left[\sum_{i=0}^{p}\sum_{j=0}^{q}\binom{p}{i}\binom{q}{i}x_\ell^{p-i}\,y_\ell^{q-j}\,\Delta x_\ell^{i}\,\Delta y_\ell^{j}\,t^{i+j}\right]dt \tag{24}$$

Comparison of Eqs. (20) and (24) allows us to exploit the symmetry of the $a = o$ and $b = o$ cases to write the solution of Eq. (24) as

$$m_{pq\ell} = \sum_{i=0}^{p}\sum_{j=0}^{q+1}\frac{(-1)^{i+j}}{i+j+1}\binom{p}{i}\binom{q+1}{j}x_\ell^{p-i}\,y_\ell^{q-j+1}\,\Delta x_\ell^{i+1}\,\Delta y_\ell^{j} \tag{25}$$

Now we have three formulas for $m_{pq\ell}$ - Eqs. (19), (23), and (25) - corresponding to three choices of the independent parameter. Which formula should be used for computation? In the next section, we will examine the special case of chain-encoded curves, when Δx and Δy take on only the values zero, one, and minus one. For now, let us consider the general case of polygonal approximation, where Δx and Δy are arbitrary real numbers. We assume that all moments up to order $n = p + q$ are to be computed in a single boundary traversal.

The computational requirements for m_{00} are the same for all parameter choices. For $p + q > 0$, the number of terms, N, for each parameter choice can be read directly from the formulas:

$$\begin{aligned}
&\text{For choice 1, } a = -b,\ N = (p+1)(q+1) = pq + p + q + 1 \\
&\text{For choice 2, } a = 0,\ N = (p+2)(q+1) = pq + p + 2q + 2 \\
&\text{For choice 3, } b = 0,\ N = (p+1)(q+2) = pq + 2p + q + 2
\end{aligned} \tag{26}$$

The equations in (26) show that choice 1 requires at least p additions fewer than choice 2 and q additions fewer than choice 3. Equation (26) also shows that choices 2 and 3 differ by p-q terms. Therefore, choice 2 is more efficient than choice 3 when $p > q$ and choice 3 is more efficient when $p < q$.

Equations (23) and (25) are homogeneous polynomials in x, y, Δx, and Δy, of degree p+q+2. Equation (19) is homogeneous of degree p+q and requires one extra multiplication for the A_ℓ term. Therefore, choice 1 requires at most the same number of multiplications needed for choices 2 and 3. Clearly, for the general case of polygonal approximation to a curve, Eq. (19) is the logical formula for computing moments.

IV. MOMENTS FOR CHAIN-ENCODED CURVES

Chain code is a special case of polygonal approximation. Each segment of the boundary connects a grid point to one of its eight nearest neighbors, represented by the numbers zero through seven. Figures 2 and 3 show our conventions for labelling the chain code directions, image coordinates, and the positive orientation of the boundary.

Using a chain-encoded boundary representation, we would like to compute the zeroeth through second order moments: m_{00}, m_{10}, m_{01}, m_{20}, m_{11}, and m_{02}. We have a choice of two sets of formulas. The first set is derived from Eq. (19):

$$
\begin{aligned}
m_{00} &= \frac{1}{2}\sum_{\ell=1}^{n} A_\ell \\
m_{10} &= \frac{1}{3}\sum_{\ell=1}^{n} A_\ell \left(x_\ell - \frac{1}{2}\Delta x_\ell\right) \\
m_{01} &= \frac{1}{3}\sum_{\ell=1}^{n} A_\ell \left(y_\ell - \frac{1}{2}\Delta y_\ell\right) \\
m_{20} &= \frac{1}{4}\sum_{\ell=1}^{n} A_\ell \left(x_\ell^2 - x_\ell\,\Delta x_\ell + \frac{1}{3}\Delta x_\ell^2\right) \\
m_{11} &= \frac{1}{4}\sum_{\ell=1}^{n} A_\ell \left(x_\ell y_\ell - \frac{1}{2}x_\ell\,\Delta y_\ell - \frac{1}{2}y_\ell\,\Delta x_\ell + \frac{1}{3}\Delta x_\ell\,\Delta y_\ell\right) \\
m_{02} &= \frac{1}{4}\sum_{\ell=1}^{n} A_\ell \left(y_\ell^2 - y_\ell\,\Delta y_\ell + \frac{1}{3}\Delta y_\ell^2\right)
\end{aligned}
\tag{27}
$$

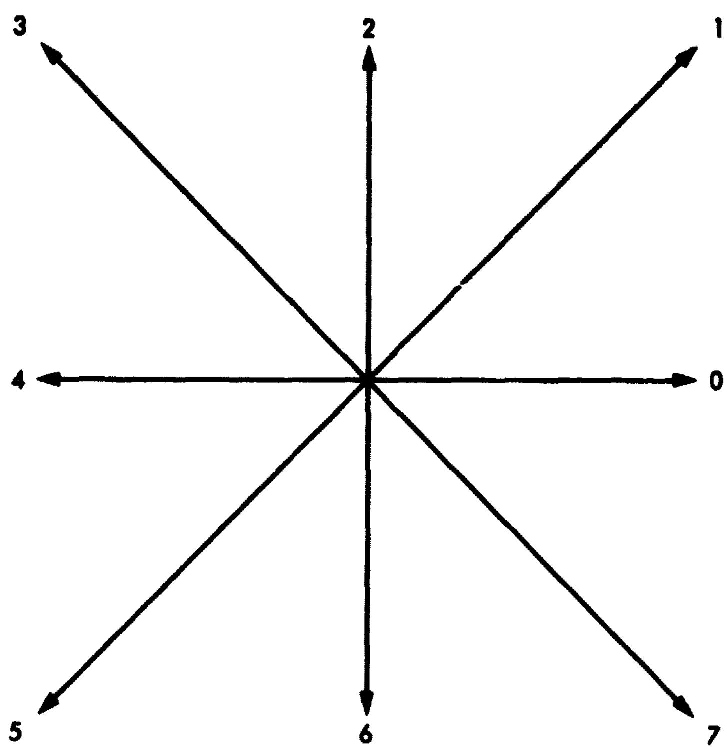

Figure 2. The eight possible directions between a grid point and its nearest neighbors are represented by the chain code numbers as shown above.

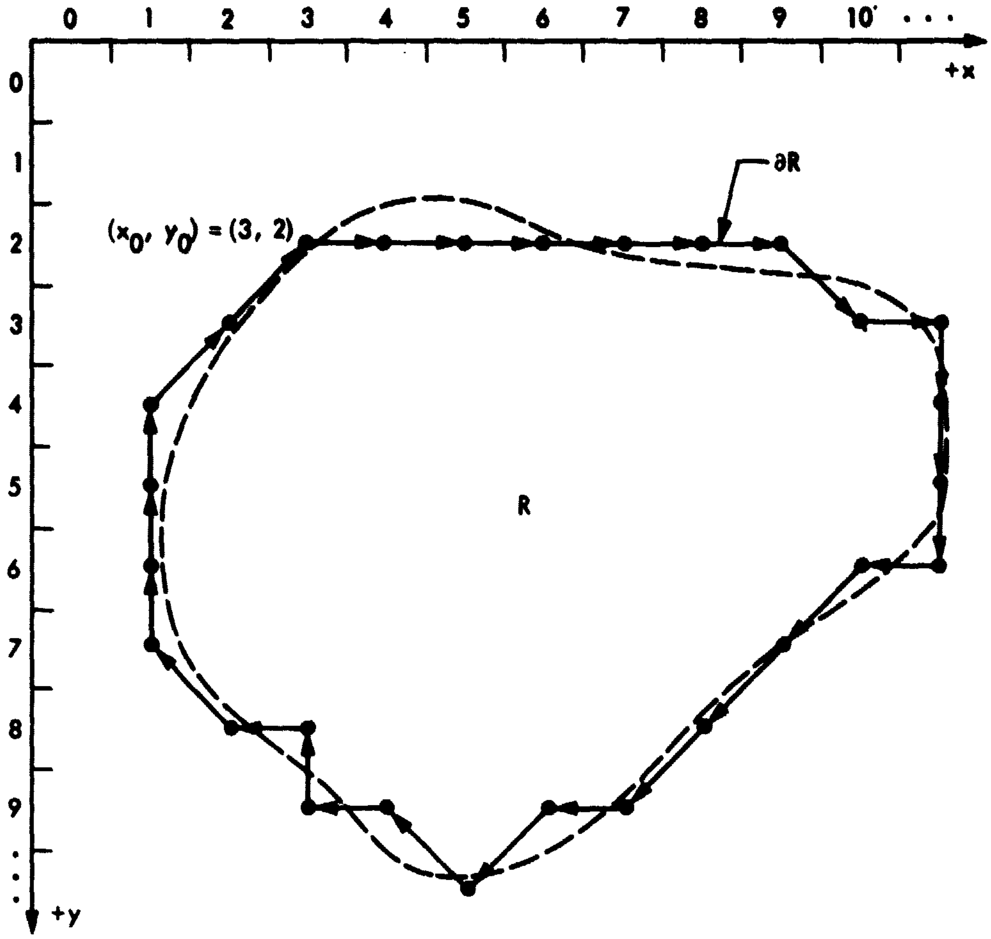

Figure 3. The region R, shown here as bordered by the dotted line, has its boundary, ∂R, approximated by a chain-encoded curve. The chain code for ∂R is 0,0, 0,0,0,0,0,7,0,6,6,6,4,5,5,5,4,5,3,4,2,4,3,2,2,2,1,1.

The second set of formulas will derive from Eqs. (23) and (25) by using (23) if $p \geq q$, and Eq. (25) otherwise. The formulas are

$$m_{00} = \sum_{\ell=1}^{n} \left(x_\ell \, \Delta y_\ell - \frac{1}{2} \Delta x_\ell \, \Delta y_\ell\right)$$

$$m_{10} = \frac{1}{2} \sum_{\ell=1}^{n} \left(x_\ell^2 \, \Delta y_\ell - x_\ell \, \Delta x_\ell \, \Delta y_\ell + \frac{1}{3} \Delta x_\ell^2 \, \Delta y_\ell\right)$$

$$m_{01} = -\frac{1}{2} \sum_{\ell=1}^{n} \left(y_\ell^2 \, \Delta x_\ell - y_\ell \, \Delta x_\ell \, \Delta y_\ell + \frac{1}{3} \Delta x_\ell \, \Delta y_\ell^2\right)$$

$$m_{20} = \frac{1}{3} \sum_{\ell=1}^{n} \left(x_\ell^3 \, \Delta y_\ell - \frac{3}{2} x_\ell^2 \, \Delta x_\ell \, \Delta y_\ell + x \, \Delta x_\ell^2 \, \Delta y_\ell - \frac{1}{4} \Delta x_\ell^3 \, \Delta y_\ell\right) \qquad (28)$$

$$m_{11} = \frac{1}{2} \sum_{\ell=1}^{n} \left(x_\ell^2 \, y_\ell \, \Delta y_\ell - \frac{1}{2} x_\ell^2 \, \Delta y_\ell^2 - x_\ell \, y_\ell \, \Delta x_\ell \, \Delta y_\ell\right.$$

$$\left. + \frac{2}{3} x_\ell \, \Delta x_\ell \, \Delta y_\ell^2 + \frac{1}{3} y_\ell \, \Delta x_\ell^2 \, \Delta y_\ell - \frac{1}{4} \Delta x_\ell^2 \, \Delta y_\ell^2\right)$$

$$m_{02} = -\frac{1}{3} \sum_{\ell=1}^{n} \left(y_\ell^3 \, \Delta x_\ell - \frac{3}{2} y_\ell^2 \, \Delta x_\ell \, \Delta y_\ell + y_\ell \, \Delta x_\ell \, \Delta y_\ell^2 - \frac{1}{4} \Delta x_\ell \, \Delta y_\ell^3\right)$$

These formulas can be evaluated most efficiently if the terms of each sum are accumulated separately as the boundary is traversed. Let PMpq[k] represent the kth term, in the order written above, of moment m_{pq}. For example, PM10[2] = $x_\ell \Delta x_\ell \Delta y_\ell$ in Eq. (28). After completing the traversal, each PMpq[k] term is multiplied by its coefficient. These quantities are then combined to obtain the moments.

The chain code representation allows another simplification. Since Δx_ℓ and Δy_ℓ are limited to the values zero, one, and minus one, it is not necessary to perform the multiplications by Δx_ℓ and Δy_ℓ indicated in the formulas. If Δx_ℓ $(\Delta y_\ell) = 0$, then the terms containing Δx_ℓ (Δy_ℓ) can be ignored. Otherwise, Δx_ℓ and Δy_ℓ simply determine the sign of each term. The values of Δx_ℓ and Δy_ℓ are completely determined by the chain code number. We therefore treat each of the eight chain code directions as a separate case. Table 1 shows the eight cases for m_{00}, using the formula from Eq. (28).

Now, we must decide which set of formulas executes in the least amount of time, using the implementation strategy discussed above. Clearly, the formulas in Eq. (28) have more terms than those in Eq. (27). However, this does not tell the whole story, since the restrictions on Δx_ℓ and Δy_ℓ mean that each term is not necessarily computed every time. The probability that a term is computed equals the probability that its Δx_ℓ or Δy_ℓ factors are nonzero. Let π_k be the probability that the kth term is computed and let P(z) represent the probability that event z occurs. Then,

$$\pi_k = \begin{cases} 1, & \text{if there are no } \Delta x \text{ or } \Delta y \text{ factors present} \\ P(\Delta x \neq 0), & \text{if a } \Delta x \text{ but not a } \Delta y \text{ factor is present} \\ P(\Delta y \neq 0), & \text{if a } \Delta y \text{ but not a } \Delta x \text{ factor is present} \\ P(\Delta x \Delta y \neq 0), & \text{if both } \Delta x \text{ and } \Delta y \text{ factors are present} \end{cases} \tag{29}$$

Table 1. Chain code values and an example of moment calculations

Chain code	0	1	2	3	4	5	6	7
Δx	1	1	0	-1	-1	-1	0	1
Δy	0	-1	-1	-1	0	1	1	1
PM00[1]=$x\Delta y$	0	-x	-x	-x	0	x	x	x
PM00[2]=$\Delta x\Delta y$	0	-1	0	1	0	-1	0	1

The eight chain code values are shown with corresponding values of Δx, Δy, and the terms PM00[1], PM00[2] used to compute m_{00}.

Groen and Verbeek [6] have calculated that, for an arbitrary chain-encoded figure, the probability of an even chain code link is 0.5858 and the probability of an odd link is 0.4142. Therefore, referring to Table 1,

$$P(\Delta x \neq 0) = P(\text{odd code}) + \frac{1}{2} P(\text{even code}) = 0.7071$$

$$P(\Delta y \neq 0) = P(\text{odd code}) + \frac{1}{2} P(\text{even code}) = 0.7071 \qquad (30)$$

$$P(\Delta x \Delta y \neq 0) = P(\text{odd code}) = 0.4142$$

Let T_k be the execution time of PMpq[k], M the execution time for one multiplication, and A the execution time for one additión. Let μ_k be the number of multiplications needed to calcualte PMpq[k]. Then the expected execution time for this term, $E(T_k)$, is given by

$$E(T_k) = \pi_k \ (\mu_k \ M + A) \qquad (31)$$

Table 2 gives π_k, μ_k, and $E(T_k)$ for every term in Eqs. (27) and (28). The total expected execution time E(T) is just the sum of the $E(T_k)$ over all terms in the formula. The result of adding the $E(T_k)$ in Tables 2 and 3 reveals that

For Eq. (27), $E(T) = 5M + 12.4852A$

For Eq. (28), $E(T) = 3.9497M + 10.3343A$ (32)

Therefore, we will use equation (28) to calculate the moments from a chain-encoded boundary.

Table 2. Computational requirements for Equations (27)

m_{pq}	k	PMpq[k]	π_k	μ_k	$\pi_k\mu_k$
m_{00}	1	A_ℓ	1	0	0
m_{10}	1	$A_\ell x$	1	1	1
	2	$A_\ell \Delta x$	0.7071	0	0
m_{01}	1	$A_\ell y$	1	1	1
	2	$A_\ell \Delta y$	0.7071	0	0
m_{20}	1	$A_\ell x^2$	1	1[a]	1
	2	$A_\ell x\Delta x$	0.7071	0[a]	0
	3	$A_\ell \Delta x^2$	0.7071	0	0
m_{11}	1	$A_\ell xy$	1	1	1
	2	$A_\ell x\Delta y$	0.7071	0[a]	0
	3	$A_\ell y\Delta x$	0.7071	0[a]	0
	4	$A_\ell \Delta x\Delta y$	0.4142	0	0
m_{02}	1	$A_\ell y^2$	1	1	1
	2	$A_\ell y\Delta y$	0.7071	0[a]	0
	3	$A_\ell \Delta y^2$	0.7071	0	0
Total			12.0710		5

The expected number of additions is $.4142 + \varepsilon\pi_k = 12.4852$.[b]

The expected number of multiplications is $\varepsilon\pi_k\mu_k = 5$.

[a]All or part of an indicated product is computed for a previous term; e.g., $PM20[1] = A_\ell x^2 = (A_\ell x)x = PM10[1]\cdot x$.

[b]There is one extra addition required to compute $A_\ell = x\Delta y - y\Delta x$ which is performed with probability $P(\Delta x\Delta y \neq 0) = .4142$.

Table 3. Computational requirements for Equations (28)

m_{pq}	k	PMpq[k]	π_k	μ_k	$\pi_k\mu_k$
m_{00}	1	$x\Delta y$	0.7071	0	0
	2	$\Delta x\Delta y$	0.4142	0	0
m_{10}	1	$x^2\Delta y$	0.7071	1	0.7071
	2	$x\Delta x\Delta y$	0.4142	0	0
	3	$\Delta x^2\Delta y$	0.4142	0	0
m_{01}	1	$y^2\Delta x$	0.7071	1	0.7071
	2	$y\Delta x\Delta y$	0.4142	0	0
	3	$\Delta x\Delta y^2$	0.4142	0	0
m_{20}	1	$x^3\Delta y$	0.7071	1[a]	0.7071
	2	$x^2\Delta x\Delta y$	0.4142	0[a]	0
	3	$x\Delta x^2\Delta y$	0.4142	0	0
	4	$\Delta x^3\Delta y$	0[b]	0	0
m_{11}	1	$x^2y\Delta y$	0.7071	1[a]	0.7071
	2	$x^2\Delta y^2$	0.7071	0[a]	0
	3	$xy\Delta x\Delta y$	0.4142	1	0.4142
	4	$x\Delta x\Delta y^2$	0.4142	0	0
	5	$y\Delta x^2\Delta y$	0.4142	0	0
	6	$\Delta x^2\Delta y$	0.4142	0	0
m_{02}	1	$y^3\Delta x$	0.7071	1[a]	0.7071
	2	$y^2\Delta x\Delta y$	0.4142	0[a]	0
	3	$y\Delta x\Delta y^2$	0.4142	0	0
	4	$\Delta x\Delta y^3$	0[b]	0	0
Total			10.3343		3.9497

The expected number of additions is $\Sigma\pi_k = 10.3343$.

The expected number of multiplications is $\Sigma\pi_k\mu_k = 3.9497$.

[a]All or part of an indicated product is computed for a previous term; e.g., $PM20[1] = x^3\Delta y = x(x^2\Delta y) = x\cdot PM10[1]$.

[b]These terms are not computed, since for Δx, $\Delta y\varepsilon\{-1,0,1\}, \Delta x\Delta y = \Delta x^3\Delta y = \Delta x\Delta y^3$. Only $PM00[2] = \Delta x\Delta y$ is computed.

V. THE ALGORITHMS

The first algorithm presented here computes moments from arbitrary polygonal boundary curves, using the formulas in equation (27). The following information is needed as input:

NVERT — the number of vertices in the polygon.

X[i], Y[i], i = 0, 1, 2, ..., NVERT — ordered lists of the x and y coordinates, respectively, of the vertices of the polygon. For convenience, X[0] = X[NVERT] and Y[0] = Y[NVERT]. The coordinates are labelled by the convention shown in Fig 3. (0,0) is the upper lefthand corner of the image; the +x axis points to the right, and the +y axis points downward.

MAXORD — the highest order moment to calculate

In addition, it assumed that the procedure BCOEFF(m,n) which calculates the binomial coefficient $\binom{m}{n}$ has been declared.

The output of this algorithm is a list of all (p + q)-th order moments where $p + q \le$ MAXORD. The moments are stored in lexicographical order in the array MOMENT as follows:

$$m_{00}, m_{10}, m_{01}, m_{20}, m_{11}, m_{02}, \ldots, m_{0MAXORD}$$

All entries of the array MOMENT are assumed to be initialized to zero.

```
BEGIN
  INTEGER L
  FOR L:=1 STEP 1 UNTIL NVERT DO
```

```
BEGIN     COMMENT traverse boundary;
  INTEGER K,M;
  REAL DELTAX,DELTAY,AL;

  DELTAX:=X[L] - X[L-1];
  DELTAY:=Y[L] - Y[L-1];
  AL:=X[L]*DELTAY - Y[L]*DELTAX;
  MOMENT[1]:=MOMENT[1] + AL;
  K:=1;   COMMENT K indexes the array MOMENT;
  FOR M:=1 STEP 1 UNTIL MAXORD DO
  BEGIN     COMMENT calculate moments of order M;
    INTEGER P,Q;
    FOR P:=M STEP -1 UNTIL 0 DO
    BEGIN    COMMENT select next p,q;
      INTEGER I,SI;   REAL MPQL;
      Q:=M-P;
      MPQL:=0;  K:=K+1;  SI:=1;
      FOR I:=0 STEP 1 UNTIL P DO
      BEGIN     COMMENT outer sum;
        INTEGER J,SJ,PI
        PI:=BCOEFF(P,I);
        SI:=-SI;  SJ:=SI;
        FOR J:=0 STEP 1 UNTIL Q DO
        BEGIN     COMMENT inner sum;
          INTEGER EXP:   REAL T;
          COMMENT initialize partial product for this term;
          T:=SJ:=-SJ;
```

```
                FOR EXP:=I STEP -1 UNTIL 1 DO T:=T*DELTAX;
                FOR EXP:=J STEP -1 UNTIL 1 DO T:=T*DELTAY;
                FOR EXP:=P-I STEP -1 UNTIL 1 DO T:=T*X[L];
                FOR EXP:=Q-J STEP -1 UNTIL 1 DO T:=T*Y[L];
                MPQL:=MPQL + (PI*BCOEFF (Q,J)*T)/I+J-1);
              END;        COMMENT end of FOR-J loop;
            END;        COMMENT end of FOR-I loop;
            MOMENT[K]:=MOMENT[K] + AL*MPQL;
          END;        COMMENT end of FOR-P loop;
        END;        COMMENT end of FOR-M loop;
      END;        COMMENT end of boundary;
      BEGIN   COMMENT scale moments by 1/(p+q+2)
        INTEGER K,M,I;
        K:=0
        FOR M:=0 STEP 1 UNTIL MAXORD DO
          FOR I:=0 STEP 1 UNTIL M DO
          BEGIN  K:=K+1;  MOMENT[K]:=MOMENT[K]/(M+2) END; COMMENT M=P+Q;
      END;
    END;
```

The second algorithm computes specific low order moments from chain-encoded curves. It needs the following information:

X,Y — the coordinates of the starting point of the chain. The coordinate labelling convention described above is used.

L — the number of links in the chain. This is the number of segments that make up the boundary curve.

CC[i], i = 1, ..., L - the list of chain code representing the curve, CC[i] ε(0, ..., 7) as shown in Fig. 2. The chain code is ordered so that the curve is traversed in a clockwise direction.

This algorithm generates a list of low order moments stored in the array MOMENT as follows:

m_{00}, m_{10}, m_{01}, m_{20}, m_{11}, m_{02}.

```
BEGIN
  INTEGER ARRAY PM00[1:2],PM10,PM01,PM20,PM02[1:3],PM11[1:6];
  INTEGER I,X2,X3,Y2,Y3,XY,X2Y;

  COMMENT initialize sums;
  PM00[1]:=PM00[2]:=0;
  FOR I:=1 STEP 1 UNTIL 3 DO
    PM10[I]:=PM01[I]:=PM20[I]:=PM11[I]:=PM02[I]:=0;
  PM11[4]:=PM11[5]:=PM11[6]:=0;

  FOR I:=1 STEP 1 UNTIL L DO
    CASE CC[I] OF
    BEGIN COMMENT case number corresponds to chain code value;

    [0] BEGIN COMMENT Dx=1, Dy=0;
        X:=X+1;
        Y2:=Y*Y;  Y3:=Y2*Y;
        PM01[1]:=PM01[1] + Y2;
        PM02[1]:=PM02[1] + Y3        END;
```

```
[1]  BEGIN COMMENT Dx=1, Dy=-1;

     X:=X+1;  Y:=Y-1;

     X2:=X*X;  X3:=X2*X;  Y2:=Y*Y;  Y3:=Y2*Y;  XY:=X*Y;
                                              X2Y:=X*XY;

     PM00[1]:=PM00[1] - X;

     PM00[2]:=PM00[2] - 1;

     PM10[1]:=PM10[1] - X2;

     PM10[2]:=PM10[2] - X;

     PM10[3]:=PM10[3] - 1;

     PM01[1]:=PM01[1] + Y2;

     PM01[2]:=PM01[2] - Y;

     PM01[3]:=PM01[3] + 1;

     PM20[1]:=PM20[1] - X3;

     PM20[2]:=PM20[2] - X3;

     PM20[3]:=PM20[3] - X;

     PM11[1]:=PM11[1] - X2Y;

     PM11[2]:=PM11[2] + X2;

     PM11[3]:=PM11[3] - XY;

     PM11[4]:=PM11[4] + X;

     PM11[5]:=PM11[5] - Y;

     PM11[6]:=PM11[6] + 1;

     PM02[1]:=PM02[1] + Y3;

     PM02[2]:=PM02[2] - Y2;

     PM02[3]:=PM02[3] + Y      END;

[2]  BEGIN COMMENT Dx=0, Dy=-1;

     Y:=Y-1;

     X2:=X*X;  X3:=X2*X;  X2Y:=X2*Y;
```

```
    PM00[1]:=PM00[1] - X;
    PM10[1]:=PM10[1] - X2;
    PM11[1]:=PM11[1] - X2Y;
    PM11[2]:=PM11[2] + X2;
    PM20[1]:=PM20[1] - X3      END;

[3] BEGIN COMMENT Dx=-1, Dy=-1;
    X:=X-1;  Y:=Y-1;
    X2:=X*X;  X3:=X2*X;  Y2:=Y*Y;  Y3:=Y2*Y;  XY:=X*Y;
                                              X2Y:=X*XY;
    PM00[1]:=PM00[1] - X;
    PM00[2]:=PM00[2] + 1;
    PM10[1]:=PM10[1] - X2;
    PM10[2]:=PM10[2] + X;
    PM10[3]:=PM10[3] - 1;
    PM01[1]:=PM01[1] - Y2;
    PM01[2]:=PM01[2] + Y;
    PM01[3]:=PM01[3] - 1;
    PM20[1]:=PM20[1] - X3;
    PM20[2]:=PM20[2] + X2;
    PM20[3]:=PM20[3] - X;
    PM11[1]:=PM11[1] - X2Y;
    PM11[2]:=PM11[2] + X2;
    PM11[3]:=PM11[3] + XY;
    PM11[4]:=PM11[4] - X;
    PM11[5]:=PM11[5] - Y;
    PM11[6]:=PM11[6] + 1;
    PM02[1]:=PM02[1] - Y3;
```

```
     PM02[2]:=PM02[2] + Y2;
     PM02[3]:=PM02[3] - Y      END;

[4]  BEGIN COMMENT Dx=-1, Dy=0;
     X:=X-1;
     Y2:=Y*Y;  Y3:=Y2*Y;
     PM01[1]:=PM01[1] - Y2;
     PM02[1]:=PM02[1] - Y3      END;

[5]  BEGIN COMMENT Dx=-1, Dy=1;
     X:=X-1;  Y:=Y+1;
     X2:=X*X;  X3:=X2*X;  Y2:=Y*Y;  Y3:=Y2*Y;  XY:=X*Y;
                                             X2Y:=X*XY;
     PM00[1]:=PM00[1] + X;
     PM00[2]:=PM00[2] - 1;
     PM10[1]:=PM10[1] + X2;
     PM10[2]:=PM10[2] - X;
     PM10[3]:=PM10[3] + 1;
     PM01[1]:=PM01[1] - Y2;
     PM01[2]:=PM01[2] - Y;
     PM01[3]:=PM01[3] - 1;
     PM20[1]:=PM20[1] + X3;
     PM20[2]:=PM20[2] - X2;
     PM20[3]:=PM20[3] + X;
     PM11[1]:=PM11[1] + X2Y;
     PM11[2]:=PM11[2] + X2;
     PM11[3]:=PM11[3] - XY;
     PM11[4]:=PM11[4] - X;
```

```
     PM11[5]:=PM11[5] + Y;
     PM11[6]:=PM11[6] + 1;
     PM02[1]:=PM02[1] - Y3;
     PM02[2]:=PM02[2] - Y2;
     PM02[3]:=PM02[3] - Y      END;

[6]  BEGIN COMMENT Dx=0, Dy=1;
     Y:=Y+1;
     X2:=X*X;  X3:=X2*X;  X2Y:=X2*Y;
     PM00[1]:=PM00[1] + X;
     PM10[1]:=PM10[1] + X2;
     PM11[1]:=PM11[1] + X2Y;
     PM11[2]:=PM11[2] + X2;
     PM20[1]:=PM20[1] + X3      END;

[7]  BEGIN COMMENT Dx=1, Dy=1;
     X:=X+1;  Y:=Y+1;
     X2:=X*X;  X3:=X2*X;  Y2:=Y*Y;  Y3:=Y2*Y;  XY:=X*Y;
                                              XY2:=X*XY;
     PM00[1]:=PM00[1] + X;
     PM00[2]:=PM00[2] + 1;
     PM10[1]:=PM10[1] + X2;
     PM10[2]:=PM10[2] + X;
     PM10[3]:=PM10[3] + 1;
     PM01[1]:=PM01[1] + Y2;
     PM01[2]:=PM01[2] + Y;
     PM01[3]:=PM01[3] + 1;
```

```
            PM20[1]:=PM20[1] + X3;

            PM20[2]:=PM20[2] + X2;

            PM20[3]:=PM20[3] + X;

            PM11[1]:=PM11[1] + X2Y;

            PM11[2]:=PM11[2] + X2;

            PM11[3]:=PM11[3] + XY;

            PM11[4]:=PM11[4] + X;

            PM11[5]:=PM11[5] + Y;

            PM11[6]:=PM11[6] + 1;

            PM02[1]:=PM02[1] + Y3;

            PM02[2]:=PM02[2] + Y2;

            PM02[3]:=PM02[3] + Y      END

       END;

    COMMENT now combine partial sums to get the moments

      The moments are computed in the order M00, M10, M01, M20,

      M11, M02;

    MOMENT[1]:= PM00[1] - PM00[2]/2;

    MOMENT[2]:= (3*PM10[1] - 3*PM10[2] + PM10[3])/6;

    MOMENT[3]:=-(3*PM01[1] - 3*PM01[2] + PM01[3])/6;

    MOMENT[4]:= (4*PM20[1] - 6*PM20[2] + 4*PM20[3] - PM00[2])/12;

    MOMENT[5]:= (12*PM11[1] - 6*PM11[2] - 12*PM11[3] + 8*PM11[4]
                            + 4*PM11[5] - 3*PM11[6])/24;

    MOMENT[6]:=-(4*PM02[1] - 6*PM02[2] + 4*PM02[3] - PM00[2])/12

  END;
```

REFERENCES

1. M. Hu, "Visual Pattern Recognition by Moment Invariants," IRE Trans. Information Theory, IT-8, 1962, pp. 179-187.

2. R. Wong and E. Hall, "Scene Matching With Invariant Moments," Computer Graphics and Image Processing, Vol. 8, pp. 16-24, 1978.

3. R. Eskenazi and J. Wilf, Low Level Processing for Real-Time Image Analysis, Publication 79-79, Jet Propulsion Laboratory, Pasadena, California, Sept. 1979.

4. H. Freeman, "Computer Processing of Line-Drawing Images," Computing Surveys, Vol. 6, pp. 57-97, 1974.

5. H. Flanders, Differential Forms with Applications to the Physical Sciences, Academic Press, New York, 1963.

6. F. Groen and P. Verbeek, "Freeman-Code Probabilities of Object Boundary Quantized Contours," Computer Graphics and Image Processing, Vol. 7, pp. 391-402, 1978.