Fast and Exact Computation of Moments Using Discrete Green's Theorem

Luren Yang and Fritz Albregtsen
Image Processing Laboratory
Department of Informatics, University of Oslo, P.O.Box 1080 Blindern, N-0316 Oslo, Norway

Abstract

Green's theorem evaluates a double integral over the region of an object by a simple integration along the boundary of the object. It has been used in moment computation since the shape of a binary object is totally determined by its boundary. By using a discrete analogue of Green's theorem, we present a new algorithm for fast computation of geometric moments. The algorithm is faster than previous methods, and gives exact results. The importance of exact computation is discussed by examining the invariance of Hu's moments. A fast method for computing moments of regions in grey level image, using discrete Green's theorem, is also presented.

1 Introduction

Moments have been widely used in shape analysis and pattern recognition [1]–[9]. The $(p+q)$'th order moment of an image is defined as

$$m_{pq} = \int_y \int_x g(x,y) x^p y^q dx dy \tag{1}$$

where $g(x,y)$ is the intensity as a function of spatial position. The double integral is often replaced by a double summation in discrete images

$$m_{pq} = \sum_y \sum_x g(x,y) x^p y^q \tag{2}$$

In the most applications, moments of orders up to 3 are used. Higher order moments have also been applied [4]. Hu [1] proposed a set of moment invariants, which do not depend on translation, rotation, or scaling. Other types of moment invariants includes the affine moment invariants recently proposed by Flusser and Suk [8], which achieve invariance under general affine transformation.

In binary images, assuming the object has pixel value 1 and the background has pixel value 0, equation (2) becomes

$$m_{pq} = \sum_{(x,y)\in O} x^p y^q dx dy \tag{3}$$

where O denotes the region of the object.

To generate the moments directly by using (1) or (2) one has to perform a significant amount of computation. It requires additions and multiplications both of $O(N^2)$, where N is the vertical or horizontal size of an image. Since the value of the moments are large, integer words do not give enough precision, and therefore long integers or float numbers have to be used. The upper bound for a $(p+q)$'th order moment is given by Sadjadi and Hall [10],

$$m_{pq} \leq g_{max} \left[\frac{N^{p+1}-1}{p+1}\right] \left[\frac{N^{q+1}-1}{q+1}\right] \tag{4}$$

where g_{max} is the maximal grey level. Using long integer or float calculations further slow down the computation of moments. Thus, the computational cost limits the use of moments in on-line, and even in off-line applications.

Many algorithms have been developed to speed up the computation of moments by reducing the computational redundancy [11]–[27]. Some of them work for grey level images, some of them for binary images, and some of them for parallel computation or optical implementation. In this article, we propose two new methods for fast computation of moments by using a discrete analogue of Green's theorem. One of them is used to compute geometric moments from binary image; the other to compute moments of regions in grey level images.

2 State of the art

In order to speed up the computation of moments, the following techniques have been used.

1. Image filtering or transform: Hatamian [15] computed the moments by a causal spatial filter. The filtering needs only additions of $O(N^2)$ for 2D images and $O(N)$ for 1D signals. Hatamian developed an algorithm for computing the moments of grey level images. Fu *et al.* [21] found the relation between the moments and the coefficients of the Hadamard transform of an image. For a binary image, the 10 moments of orders up to three are totally determined by four projections of the image, and can be computed from the coefficients of the 1D Hadamard transform of the projections. The 1D Hadamard transform needs additions of $O(N \log_2 N)$.

2. Delta method: A moment of an object is the sum of the moments of all the vertical or horizontal line segments of the object. δ is used to denote the length of a line segment, hence the name of the method. The moments of a line segment can be expressed in a closed form. The method was first proposed by Zakaria *et al.* [18], and then improved by Dai *et al.* [19] and Li [20]. The method of Dai *et al.* is called integral method since the moments of line segments are computed by integration, instead of summation used by the delta method. Given y-line representation of an object, Li's algorithm requires about $6N$ multiplications and $17N$ additions to compute the 10 low order geometric moments for a convex object. The delta method is suitable for binary images represented by y-lines.

3. Computation via corner points: The geometric moments can be computed via corner points in the boundary of the object. The boundary between two corner points is a straight line. Strachan *et al.* [17], Leu [22], and Singer [23] computed the double integral over the object by summing up the integrals over some simple regions each containing one straight line boundary. The integral over such a simple region can be expressed in a closed form. Jiang and Bunke [24] used Green's theorem to transform the double integral to a single integral. The single integral along a straight line between two corner points can be expressed in a closed form. To compute the geometric moments via corner points implies additions and multiplications of $O(C)$, where C is the amount of the corner points. Since a large amount of computation are required for each corner point, the method is effective for objects with simple shape.

4. Green's theorem: Green's theorem [28] evaluates a double integral over a region by a single integration along the boundary of the region. It is important for moment computation since the shape of a binary object is totally determined by its boundary. Li and Shen [25] proposed a fast moment computation method by using Green's theorem. The moment kernel updating technique is used so that the algorithm needs only additions of $O(L)$, where L is the length of the boundary of the object. The method is effective, but not accurate, since Li and Shen used an approximation of Green's theorem in digital image lattice. There exists discrete Green's theorem [29] for exact computation. By using a discrete analogue of Green's theorem, Philips [26] proposed a method giving exact results. Unfortunately his method is not as effective as the method of Li and Shen. We proposed a method [27] which is as effective as the method of Li and Shen, and gives exact results. In this article, we introduce a new method, which is faster than previous methods, and achieves exact computation of geometric moments. We also apply discrete Green's theorem in grey level images, and present a method for fast computation of moments of grey level regions.

5. Contour following: Contour following [30] is to go through and label the object boundary pixels clockwise or anticlockwise. Fu *et al.* [21] used contour following to make the projections of a binary image. To use Li and Shen's method or compute the moments via corner points, contour following must be applied. To use the delta method, contour following can be applied but then the start point and the end point in a line segment must be paired. Contour following is an $O(N)$ algorithm to locate an object in a binary image. To locate the object by scanning the image or by recursively including the neighbor points from a seed point will need $O(N^2)$ operations. The main disadvantage of contour following is that holes of the object will not be considered. If an object contains holes, the moments of the holes have to be subtracted.

Different methods give different precision, depending on the size, shape and complexity of the object. Exact computation means to obtain results as if the moments were computed by a double sum as in equations (2) and (3). A good precision is important to achieve invariance for Hu's moments [1]. Dai *et al.* [19] evaluated the moment invariants of Hu computed by the exact evaluation of geometric moments and by an integral method, and found that the results of the integral method were more sensitive to the sampling. The integral method is better for shapes with only vertical and horizontal edges. In this case, the integral method gives the true value of the moments. For the 10 low order moments, the differences between the results of the integral method and the exact computation have been formulated by Dai *et al* [19]. A more thorough discussion of the precision is given in Section 6.

3 Discrete versions of Green's theorem

Green's theorem [28] relates a line integral around a simple closed plane curve C to an ordinary double integral over the plane region O bounded by C. Suppose that the curve C is piecewise smooth, and functions $M(x,y)$ and $N(x,y)$ are continuous and have continuous first-order partial derivatives in O, then

$$\oint_C M\,dx + N\,dy = \int\int_O f(x,y)dA \tag{5}$$

where $\oint_C$ denotes a line integral along C in anticlockwise direction, and

$$f(x,y) = \frac{\partial N(x,y)}{\partial x} - \frac{\partial M(x,y)}{\partial y} \tag{6}$$

Direct applyication of Green's theorem on a discrete region by changing the integrations in equation (5) to summations results in an approximation, i.e.,

$$\sum_C (M\Delta x + N\Delta y) \approx \sum_O f\Delta x\Delta y \tag{7}$$

Li and Shen [25] used this approximation to compute geometric moments. The results are poor when the object is small, or the shape of the object is complex.

There are different versions of discrete Green's theorem which relate a sum over a region to a sum along the boundary. One of the differences between them is in the definition of the boundary.

3.1 Tang's version

Tang [29] defines the boundary L in a natural way:

$$L = \{p | p \in O,\ \exists q \in N_4(p),\ q \notin O\} \tag{8}$$

where $N_4(p)$ denotes the set of the 4 neighbors of a pixel p. Tang's formula is

$$\sum_O f(x,y) = \sum_L (F_x(x,y)D_Y(x,y) + f(x,y)C_Y(x,y)) \tag{9}$$

where

$$F_x(x,y) = \sum_{i=0}^{x} f(i,y) \tag{10}$$

In Tang's paper, $D_Y(x,y)$ and $C_Y(x,y)$ are defined by the Freeman chain code representation of the boundary. They can also be defined by the coordinate (x,y) of the boundary point, as we give below

$$D_Y(x,y) = \begin{cases} 1 & (x-1,y) \in O,\ (x+1,y) \notin O \\ -1 & (x-1,y) \notin O,\ (x+1,y) \in O \\ 0 & \text{otherwise} \end{cases}$$

$$C_Y(x,y) = \begin{cases} 1 & (x-1,y) \notin O \\ 0 & \text{otherwise} \end{cases}$$

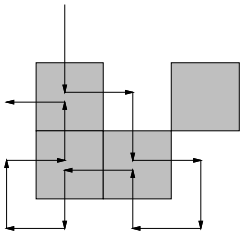

Figure 1: The simple bug contour following strategy.

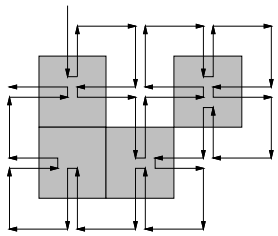

Figure 2: The backtracking contour following strategy.

3.2 Philips' version

Philips [26] defines the boundary in another way. Let $\partial O^+ = \{(x,y) | (x,y) \in O, (x+1,y) \notin O\}$ and $\partial O^- = \{(x,y) | (x,y) \notin O, (x+1,y) \in O\}$, the boundary ∂O is defined as $\partial O = \partial O^+ \cup \partial O^-$. Philips' formula is

$$\sum_O \nabla_x f(x,y) = \sum_{\partial O^+} f(x,y) - \sum_{\partial O^-} f(x,y) \tag{11}$$

where $\nabla_x f(x,y) = f(x,y) - f(x-1,y)$. Note that ∂O is not a closed boundary of the region O. It is the west and east boundary. A dual formula can be obtained if the north and south boundary is given.

3.3 Our new version

We associate the discrete Green's theorem with the contour following algorithm [30]. In contour following, a conceptual bug goes through all boundary points. Simple bug contour following should be used for 4-connected regions, and backtracking contour following should be used for 8-connected regions. During simple bug contour following, the bug makes a right turn when it is in the background, and makes a left turn when it is in the object, as illustrated in Fig. 1. In backtracking contour following, the bug makes a right turn when it is in the background, and returns back to the previous point when it is in the object. This is illustrated in Fig. 2.

A background-to-object transition can have four possible directions, 0, 1, 2, and 3, as illustrated in

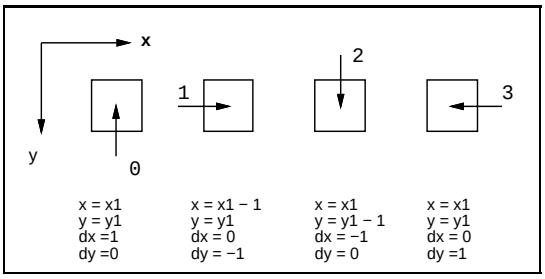

Figure 3: The four possible directions of background to object transition. The squares present the boundary points in the object, with coordinate (x, y).

Fig. 3. Thus, a transition is denoted by a quadruple $(x, y, \Delta x, \Delta y)$ which consists of a point coordinate (x, y) and $(\Delta x, \Delta y)$ giving directional information. The set of the transitions is defined as

$$\begin{aligned}
T_0 &= \{(x,y,\Delta x,\Delta y) | (x,y) \in O,\, (x,y+1) \notin O,\\
&\quad \Delta x = 1,\, \Delta y = 0\}\\
T_1 &= \{(x,y,\Delta x,\Delta y) | (x,y) \notin O,\, (x+1,y) \in O,\\
&\quad \Delta x = 0,\, \Delta y = -1\}\\
T_2 &= \{(x,y,\Delta x,\Delta y) | (x,y) \notin O,\, (x,y+1) \in O,\\
&\quad \Delta x = -1,\, \Delta y = 0\}\\
T_3 &= \{(x,y,\Delta x,\Delta y) | (x,y) \in O,\, (x+1,y) \notin O,\\
&\quad \Delta x = 0,\, \Delta y = 1\}\\
T &= T_0 \cup T_1 \cup T_2 \cup T_3
\end{aligned}$$

Note that one boundary coordinate can correspond to more than one transitions. In practical computation, only a triplet is to be registered under the contour following. As we will see later, only a triplet is involved in the computation.

By using contour following and the transition concept, we present a formula for the discrete Green's theorem

$$\sum_O f(x,y) = \sum_T F_x(x,y)\Delta y \tag{12}$$

where $F_x(x, y)$ is defined by equation (10). If we define $F_y(x, y) = \sum_{i=0}^{y} f(x, i)$, then we obtain a dual formula

$$\sum_O f(x,y) = \sum_T F_y(x,y)\Delta x \tag{13}$$

Assuming thay an object O consists of rows of points, and that the x-coordinates of the start point and the end point of the row r are $x_1(r)$ and $x_2(r)$, equation (12) can be proved as follows:

$$\begin{aligned}
\sum_O f(x,y) &= \sum_r \sum_{x=x_1(r)}^{x_2(r)} f(x,y)\\
&= \sum_r (F_x(x_2(r),y) - F_x(x_1(r)-1,y))\\
&= \sum_{T_3} F_x(x,y) - \sum_{T_1} F_x(x,y)\\
&= \sum_T F_x(x,y)\Delta y
\end{aligned}$$

Equation (13) can be proved in a similar way.

4 Computing geometric moments of binary objects

By using the discrete analogue of Green's theorem, we present a fast method to compute geometric moments, defined by equation (3). Substituting $f(x, y) = x^p y^q$ into (12), we have

$$m_{pq} = \sum_T \sum_{i=0}^{x} i^p y^q \Delta y =$$

$$\sum_T \left(\frac{x^{p+1} y^q}{p+1} + \frac{x^p y^q}{2} + \sum_{j=2}^{p} \frac{1}{j} C_p^{j-1} B_j x^{p-j+1} y^q \right) \Delta y \tag{14}$$

where C_p^{j-1} is a combination number and B_j is the j'th Bernoulli number. Let

$$u_{ij} = \sum_T x^{i+1} y^j \Delta y \tag{15}$$

Then equation (14) becomes

$$m_{pq} = \frac{u_{p\,q}}{p+1} + \frac{u_{p-1,q}}{2} + \sum_{j=2}^{p} \frac{1}{j} C_p^{j-1} B_j\, u_{p-j,q} \tag{16}$$

This is our formula for geometric moment computation. The moment m_{pq} is a linear combination of $u_{j\,q}$ for $j = 0, \cdots, p$. The 10 low order moments, which are often used in applications, can be computed as

$$\begin{bmatrix} m_{0q} \\ m_{1q} \\ m_{2q} \\ m_{3q} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 1/6 & 1/2 & 1/3 & 0 \\ 0 & 1/4 & 1/2 & 1/4 \end{bmatrix} \begin{bmatrix} u_{0q} \\ u_{1q} \\ u_{2q} \\ u_{3q} \end{bmatrix}$$

In our earlier algorithm [27], we proposed to compute the monomials $x^i y^j$ for each transition registered under contour following, and u_{ij}, as defined in (15), are computed by accumulation. $x^i y^j$ can be computed by updating, as proposed by Li and Shen [25], so that only additions are required.

Now we propose a more effective method to compute u_{ij}. Suppose an image consists of scan-lines $l(y)$, and a set of transitions in this scan-line is $T(y)$

$$T(y) = \{(x,y,\Delta x,\Delta y)|(x,y,\Delta x,\Delta y) \in T, (x,y) \in l(y)\} \quad (17)$$

We define $v_i(y)$ as

$$v_i(y) = \sum_{T(y)} x^{i+1}\Delta y \quad (18)$$

We compute $v_i(y)$ for $i = 0,\cdots,p$ and all y-coordinates. This can be implemented by using $p+1$ arrays of size N. During contour following, the array entries are updated.

Comparing equation (15) with (18), we have

$$u_{ij} = \sum_y v_i(y)y^j \quad (19)$$

which means that u_{ij} is the j'th moment of a 1D signal $v_i(y)$. A fast algorithm for computing moments of a 1D signal was proposed by Hatamian [15]. Letting $v_i^0(y)$ be the result of Hatamian filtering of $v_i(y)$, we have

$$v_i^0(y) = \sum_{k=N}^{y} v_i(k) \quad (20)$$

Applying the Hatamian's filter recursively, we obtain

$$v_i^j(y) = \sum_{k=N}^{y} v_i^{j-1}(k) \quad (21)$$

Then the 1D moments u_{ij} are linear combinations of $v_i^j(1)$. For $j \leq 3$ we have

$$\begin{bmatrix} u_{i0} \\ u_{i1} \\ u_{i2} \\ u_{i3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 2 & 0 \\ 0 & 1 & -6 & 6 \end{bmatrix} \begin{bmatrix} v_i^0(1) \\ v_i^1(1) \\ v_i^2(1) \\ v_i^3(1) \end{bmatrix}$$

Hatamian filtering of a 1D signal requires N additions.

Our new method to compute the geometric moments can be implemented in three steps:

1. The contour following is applied, and a triplet $(x,y,\Delta y)$ is recorded for each transition from the background to object. Then the array entries $v_i(y)$ are accumulated for $i = 0,\cdots,p$, see equation (18)

2. Apply Hatamian filtering to $v_i(y)$ for $i = 0,\cdots,p$ and obtain $v_i^j(y)$ for all i and j.

3. Compute the moment m_{pq} as a linear combination of $v_i^j(1)$.

Li [20] also computed geometric moments by 1D moment computation. However our method is more effective since we do not need to determine the start, and the stop point, and the length of each line segment. For the computation of the 10 low order moments, it requires, in addition to the operations for contour following, 3 multiplications and 4 additions for each T_1 and T_3 transition in the first step. The second step needs $10N$ additions. The last step needs only a few multiplications and additions. So totally it needs about $10N + 8S$ additions and $6S$ multiplications in addition to the contour following, where S is the amount of line segments of the object.

The 3 multiplications for each transition are used to compute x^i from x. In some computational environment, addition is much faster than multiplication. In this case we can use Li and Shen's updating technique [25], requiring only additions, to compute x^i, since x is changed at most by 1 from one transition to the next. Using the updating, 10 additions are used instead of 3 multiplications.

5 Computing moments of grey level region

Hatamian [15] proposed a fast algorithm to compute moments of grey level images, defined by equation (2). Sometimes we do not need to compute the moments of the whole image, but some regions of the image, as defined below

$$m_{pq} = \sum_O g(x,y)x^p y^q \quad (22)$$

where O denotes a region. The zeroth order moment of a region is the correlation of the grey level function and the region.

We present a method to compute (22) by using the discrete version of Green's theorem. Substituting $f(x,y) = g(x,y)x^p y^q$ into (12), we have

$$m_{pq} = \sum_T \sum_{k=0}^{x} g(k,y)k^p y^q \Delta y \quad (23)$$

Let

$$G^p(x,y) = \sum_{k=0}^{x} g(k,y)k^p \quad (24)$$

Equation (23) becomes

$$m_{pq} = \sum_T G^p(x,y)y^q \Delta y \quad (25)$$

Let

$$v_p(y) = \sum_{T(y)} G^p(x,y)\Delta y \tag{26}$$

where $T(y)$ are the transitions in scan-line $l(y)$ as defined by equation (17). Then we have

$$m_{pq} = \sum_y v_p(y) y^q \tag{27}$$

We see that the moment m_{pq} can be computed as the q'th order moment of a 1D signal $v_p(y)$.

Our method to compute the moments given by equation (22) is stated as follows: First compute $G^p(x,y)$ from $g(x,y)$, then apply contour following. For each transition, the values of $v_p(y)$, defined by equation (26) and implemented as an array, are accumulated. Then we apply Hatamian filtering on $v_p(y)$ and obtain $v_p^j(y)$ for $j = 0, \cdots, q$. The moment m_{pq} is then a linear combination of $v_p^j(y)$.

To compute the 10 low order moments, we need to compute $G^i(x,y)$ for $i = 0, \cdots, 3$. This requires $4N^2$ additions and $3N^2$ multiplications. These multiplications can, if desired, be changed to additions by applying Li and Shen's updating. To accumulate $v_i(y)$ we need 4 additions for each T_1 and T_3 transition. Hatamian filtering of $v_i(y)$ needs $10N$ additions.

If the moments are to be computed in more than one regions in the image, the values of $G^i(x,y)$ are computed only once. Thereafter only about $10N + 8S$ additions are required for each region. Thus, we gain more computational advantage if there are more regions.

6 The precision of the computation

The standard two-dimensional discrete moments m_{pq} of $f(x,y)$

$$m_{pq} = \sum_y \sum_x g(x,y) x^p y^q \tag{28}$$

will vary for a given shape depending on the spatial position of the object. Translation invariance is obtained using the central moments

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q g(x,y) \tag{29}$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}}, \qquad \bar{y} = \frac{m_{01}}{m_{00}} \tag{30}$$

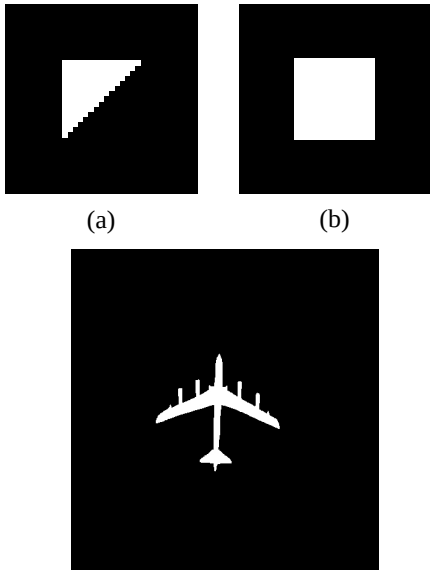

Figure 4: Three test images.

Scaling invariant central moments are obtained by the normalization

$$\eta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^\gamma}, \quad \gamma = \frac{p+q}{2} + 1, \quad p+q \geq 2 \tag{31}$$

A set of seven combinations of the second and third order normalized central moments, invariant to translation, rotation and scale change, due to Hu [1], are often cited:

$$\begin{aligned}
\phi_1 &= \eta_{20} + \eta_{02} \\
\phi_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
\phi_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
\phi_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
\phi_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12}) \\
&\quad \left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] \\
&\quad +(3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) \\
&\quad \left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \\
\phi_6 &= (\eta_{20} - \eta_{02}) \left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \\
&\quad +4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
\phi_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12}) \\
&\quad \left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] \\
&\quad +(3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03}) \\
&\quad \left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]
\end{aligned}$$

Strict invariance is achieved by assuming a continuous image function and a double integral over the region. Due to the digital nature of the image, the

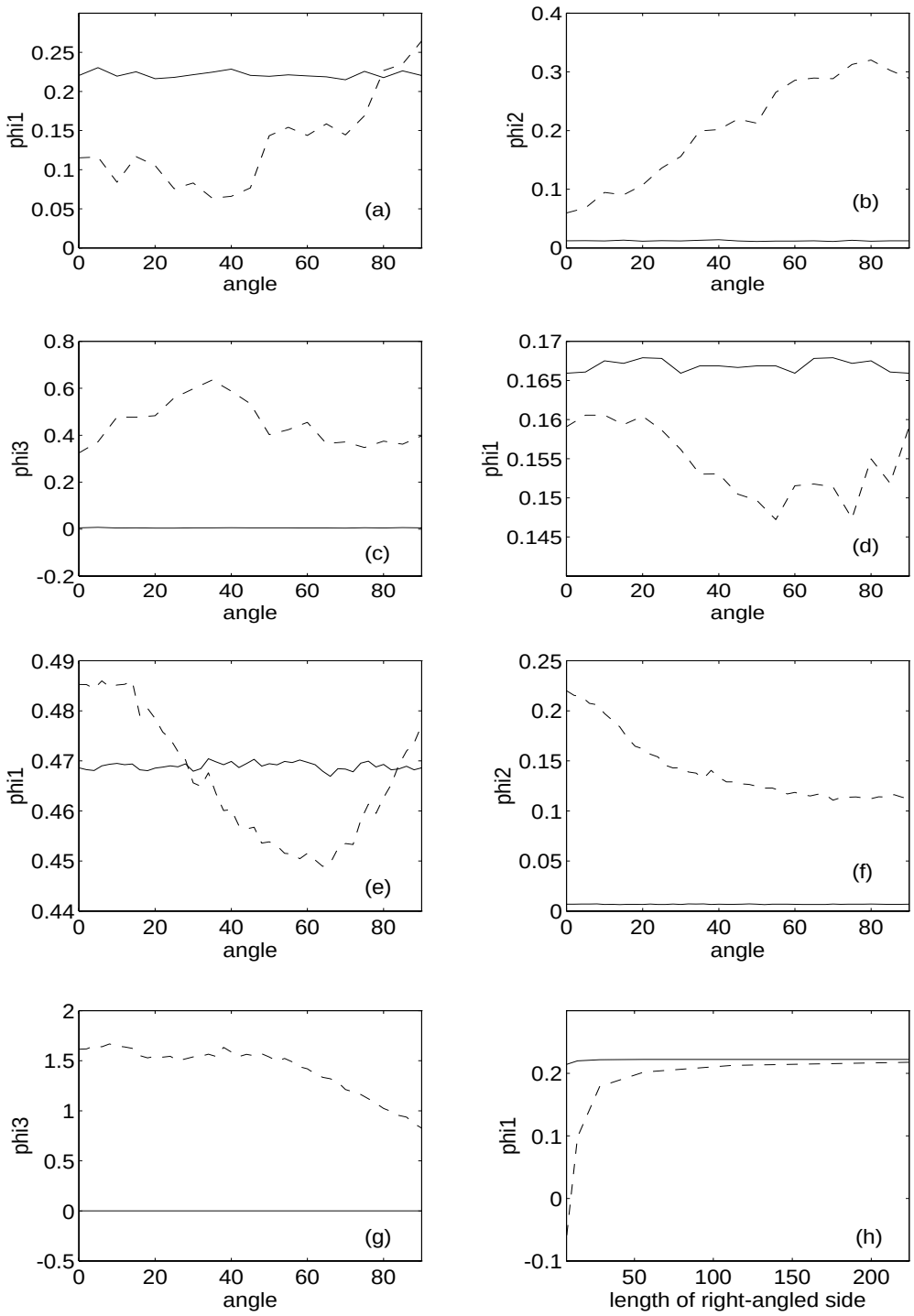

Figure 5: Invariant moments as computed by the new method (solid line) and by the method of Li and Shen (dashed line): (a) ϕ_1 of the triangle, rotation: $0° - 90°$, (b) ϕ_2 of the triangle, rotation: $0° - 90°$, (c) ϕ_3 of the triangle, rotation: $0° - 90°$, (d) ϕ_1 of the square, rotation: $0° - 90°$, (e) ϕ_1 of the aircraft, rotation: $0° - 90°$, (f) ϕ_2 of the aircraft, rotation: $0° - 90°$, (g) ϕ_3 of the aircraft, rotation: $0° - 90°$, (h) ϕ_1 of the triangle, length of the right-angled side: 7–224 pixels.

moment invariants computed digitally are not strictly invariant. The precision of the computation of the seven invariant moments of Hu are often quoted, as discussed and tested by Gonzalez and Woods [31]. It can be shown [32] that the set of moment combinations is still invariant under image translation if they are computed by the exact computation. But under digital processing, the invariant moments are expected not to be strictly invariant under image rotation and scale changes.

The method of Li and Shen, using an approximation of Green's theorem, is not accurate for small objects or objects with complex shape. By using the new exact method, the precision is largely improved. Three binary images, as shown in Fig. 4, are used to test the precision of the new method and the method of Li and Shen, in respect to the invariance of Hu's invariant moments. Fig. 4(a) shows a test image of a triangle with 120 pixels. Fig. 4(b) shows a test image of a square with 225 pixels. They are small objects. Fig. 4(c) shows a test image of an aircraft with 7086 pixels, which is complex in shape.

While the objects are rotated around their centroids clockwise from 0 to 90 degrees, the invariant moments of the three objects are computed by both methods. The results are shown in Fig. 5(a)–(g). The invariant moments computed by the new method are represented by the solid lines, while that computed by the method of Li and Shen are represented by the dashed lines. We see that the values computed by the new method are very stable under image rotation, while the values computed by the method of Li and Shen vary much.

We have shown the test results of the first three invariant moments of the triangle and the aircraft, and the first invariant moment of the square. For other invariant moments of these figures, the results are similar.

For the triangle and the square, we can compute the true values of the invariant moments analytically: ϕ_1 of the triangle is about 0.2222, ϕ_2 of the triangle is about 0.0123, ϕ_3 of the triangle is about 0.0055, and ϕ_1 of square is about 0.1667. Thus we see that the values computed by the new method are close to the true values.

Also, the new method is much less sensitive to sampling than the method of Li and Shen. This is demonstrated by comparing the invariant moment ϕ_1 of the triangles of different sizes. The triangle of the largest size (right angle side = 224 pixels) is first created, and is then successively down sampled by a factor of 2. The smallest triangle has a right angle side of 7 pixels. The results of the new method and the method of Li and Shen applied on these triangles are shown in Fig. 5(h). We see that the new method is much more stable under scale changes, and thus may be very useful in scale space applications of invariant geometric moments.

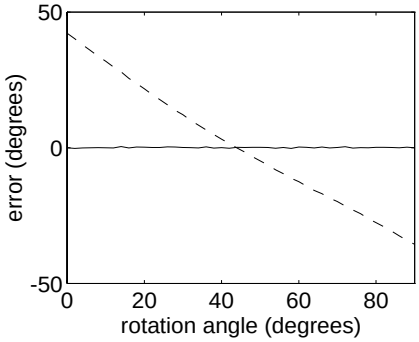

Figure 6: Error ($\Delta\theta = \hat{\theta} - \theta$) in estimated orientation ($\hat{\theta}$) as a function of true rotation angle (θ) of the aircraft object in Fig. 4, for the new method (solid line) and the method of Li and Shen (dotted line).

An alternative way of achieving rotation invariance is to estimate the object orientation angle, given by

$$\hat{\theta} = \frac{1}{2} \tan^{-1} \left[\frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \right] \tag{32}$$

and compute the normalized central moments in a rotated coordinate system. The error in the estimated orientation, as obtained by the two different methods, is given in Fig. 6 as the aircraft object is rotated around its centroid. We see that the new method gives an insignificant orientation error compared to the error introduced by the method of Li and Shen.

7 Conclusion

By using a discrete analogue of Green's theorem, we propose a new method to compute the geometric moments of binary images. Compared to previous methods, the new method is faster, and gives exact results as if the moments were computed by a double sum over the object area. Exact computation is important to achieve invariance of Hu's moments, specially for small and complex objects.

We also apply the discrete Green's theorem in grey level images, and propose a fast method to compute moments of regions in grey level images. The method is particularly effective when the moments are to be computed in a number of regions in an image.

Acknowledgments

The authors wish to thank Tor Lønnestad and Per Grøttum, the Department of Informatics, University of Oslo, for their constructive comments.

References

[1] M.-K. Hu, Visual pattern recognition by moment invariants, *IRE Trans. Information Theory* **8**, 179–187(1962).

[2] S. A. Dudani, K. J. Breeding, and R. B. McGhee, Aircraft identification by moment invariants, *IEEE Trans. Computers* **26**, 39–46(1977).

[3] R. Y. Wong, and E. L. Hall, Scene matching with invariant moments, *Comput. Graph. Image Process.* **8**, 16–24(1978).

[4] A. P. Reeves, and A. Rostampour, Shape analysis of segmented object using moments, *Proc. PRIP,* pp.171–174(1981).

[5] A. P. Reeves, M. L. Akey, and O. R. Mitchell, A moment based two-dimensional edge operator, *Proc. CVPR,* pp.312–317(1983).

[6] S. O. Belkasim, M. Shridhar, and M. Ahmadi, Pattern recognition with moment invariants: a comparative study and new results, *Pattern Recogn.* **24**(12), 1117–1138(1991).

[7] R. J. Prokop, and A. P. Reeves, A survey of moment-based techniques for unoccluded object representation and recognition, *CVGPR: Graphical Models and Image Processing* **54**(5), 438–460(1992).

[8] J. Flusser, and T. Suk, Pattern recognition by affine moment invariants, *Pattern Recogn.* **26**(1), 167–174(1993).

[9] C.-C. Chen, Improved moment invariants for shape discrimination, *Pattern Recogn.* **26**(5), 683–686(1993).

[10] F. A. Sadjadi, and E. L. Hall, Numerical computations of moment invariants for scene analysis, *Proc. PRIP,* pp.181–187(1978).

[11] D. Casasent, and D. Psaltis, Hybrid processor to compute invariant moments for pattern recognition, *Opt. Lett.* **5**(9), 395–397(1980).

[12] M. R. Teague, Optical calculation of irradiance moments, *Applied Opticals* **19**(8), 1353–1356(1980).

[13] A. P. Reeves, Parallel algorithms for real-time image processing, *Multicomputer Image Processing Algorithms and Programs,* pp.7–18, Academic Press(1982).

[14] A. P. Reeves, A parallel mesh moment computer, *Proc. 6th ICPR,* pp.465–467(1982).

[15] M. Hatamian, A real-time two-dimensional moment generating algorithm and its single chip implementation, IEEE Trans. ASSP **34**(3), 546–553(1986).

[16] K. Chen, Efficient parallel algorithms for the computation of two-dimensional image moments, *Pattern Recogn.* **23**(1/2), 109–119(1990).

[17] N. J. C. Strachan, P. Nesvadba, and A. R. Allen, A method for working out the moments of a polygon using an integration technique, *Pattern Recogn. Lett.* **11** 351–354(1990).

[18] M. F. Zakaria, L. J. Vroomen, P. J. A. Zsombor-Murray, and J. M. H. M. van Kessel, Fast algorithm for the computation of moment invariants, *Pattern Recogn.* **20**(6), 639–643(1987).

[19] M. Dai, P. Baylou, and M. Najim, An efficient algorithm for computation of shape moments from run-length codes or chain codes, *Pattern Recogn.* **25**(10), 1119–1128(1992).

[20] B.-C. Li, A new computation of geometric moments, *Pattern Recogn.* **26**(1), 109–113(1993).

[21] C.-W. Fu, J.-C. Yen, and S. Chang, Calculation of moment invariants via Hadamard transform, *Pattern Recogn.* **26**(2), 287–294(1993).

[22] J.-G. Leu, Computing a shape's moments from its boundary, *Pattern Recogn.* **24**(10), 949–957(1991).

[23] M. H. Singer, A general approach to moment calculation for polygons and line segments, *Pattern Recogn.* **26**(7), 1019–1028(1993).

[24] X. Y. Jiang, and H. Bunke, Simple and fast computation of moments, *Pattern Recogn.* **24**(8), 801–806(1991).

[25] B.-C. Li, and J. Shen, Fast computation of moment invariants, *Pattern Recogn.* **24**(8), 807–813(1991).

[26] W. Philips, A new fast algorithm for moment computation, *Pattern Recogn.* **26**(11), 1619–1621(1993).

[27] L. Yang, and F. Albregtsen, Fast computation of invariant geometric moments: a new method giving correct results, *Submitted to 12 th ICPR,* Jerusalem, Israel (1994).

[28] C. H. Edwards, Jr., and D. E. Penny, *Calculus and analytic geometry, 3/e,* pp.859–866. Prentice Hall, (1982).

[29] G. Y. Tang, A discrete version of Green's theorem, *Proc. PRIP,* pp.144–149(1981).

[30] W. K. Pratt, *Digital image processing, 2/e,* pp.623–625. Wiley-Interscience, (1991).

[31] R. C. Gonzalez, and R. E. Woods, *Digital image processing,* Addison-Wesley, (1992).

[32] C.-H. Teh, and R. T. Chin, On digital approximation of moment invariants, *Comput. Vision Graph. Image Process.* **33**, 318–326(1986).