A Recursive Updating Rule for Efficient Computation of Linear Moments in Sliding-Window Applications *

Judit Martínez, Ernesto Staffetti, and Federico Thomas
Institut de Cibernètica (CSIC – UPC)
Diagonal 647, 2 planta
08028 Barcelona

E-mail: {martinez,staffett,thomas}@ic.upc.es

Abstract

The computation of linear moment matrices, whose elements are defined as zeroth order integration values of an image, was recently introduced as a tool to reduce the computational cost required to obtain the geometric moments of an image. The main relevance of these matrices is twofold: on one hand, they can be efficiently obtained by means of accumulation filters, which only require additions; on the other one, their relation to geometric moments, as well as to discrete Fourier spectrum coefficients, allows the exchange and interpretation of many results from different areas of Image Processing and Pattern Recognition.

Taking into account the relevance of these matrices, a new recursive property that allows their efficient computation in sliding-window processes is presented here. First, a scalar recursive updating rule is formulated. It relates the value of each element of a linear moment matrix to those calculated in the previous location of the sliding-window. Then, this result is reformulated to obtain an explicit matrix formula.

The obtained recursive updating rule has a straightforward application in many different fields involving sliding-window processes in order to efficiently obtain local features related to geometric moments and discrete Fourier spectrum coefficients.

1. Introduction

Efficient computation of local features is a common requirement in many image processing applications. Typical fields include pattern recognition, edge detection, texture segmentation, etc. These local features are frequently obtained by centering a sliding-window, at any initial pixel of the image, and extracting them from the pixels inside it. Depending on whom the features belongs to, two possible shifting strategies are possible. On one hand, considering that they belong to each pixel inside the sliding-window, a non-overlapping shifting would define the next pixel to center the window. On the other one, considering the features as belonging only to one pixel, which uses to be the central one, an overlapping process would center the window on a pixel next to the initial one. In the first case a reduced number of pixels are analized; however, coarser processing results are obtained. In the second case a more precise analysis is obtained at the expenses of higher computational costs.

The linear moments of an image are the set of features used herein. They were first introduced in [3] and proved to be an efficient tool for geometric moment computation. Although these moments are related to Fourier spectrum as well as to geometric moments, as it has been recently proved in [4], their preferable usage will be conditioned by their recursive computation in a sliding-window process. The main point is that operations using all the pixels in an sliding-window are required only once. Then, when the window is shifted one pixel, either along the x or y axis, only the new pixels added to the process, called *incoming vector* for short, as well as the ones that go out of the window, the *outgoing vector*, should be considered in the computation. Two expressions, relating the elements of the new linear moment matrix to the one previously obtained and these two vectors, are derived in this work. The first one is an scalar recursive updating rule, and the second one corresponds to its explicit matrix counterpart.

This paper is organized as follow. Section II introduces the concept of linear moment matrix. Its relation to some

*This work has been partially supported by the Spanish CICYT under contract TAP 93-0451, and the HCM Program of the EC under contract No. ERBCHRXCT930086 (project HEROS).

other well known characteristics, i.e. geometric moments and discrete Fourier spectrum coefficients, is referenced. Section III states the scalar recursive updating rule in a sliding-window process, leading to a formulation that can be decomposed into three adding terms. Section IV states a matrix reformulation of this recursive rule. Computational costs derived from the application of both formulations are given in Section V. Finally, we conclude in Section VI.

2. The Linear Moment Matrix

The *linear moment matrix* associated with an image $\mathbf{I}[x,y]$, where $x = 1,\ldots,a$ and $y = 1,\ldots,b$, is defined as:

$$\mathbf{L} := \begin{pmatrix} \mathbf{I}_{x^1y^1}[a,b] & \ldots & \mathbf{I}_{x^1y^n}[a,b] \\ \vdots & & \vdots \\ \mathbf{I}_{x^my^1}[a,b] & \ldots & \mathbf{I}_{x^my^n}[a,b] \end{pmatrix}, \quad (1)$$

where $\mathbf{I}_{x^ky^l}[x,y]$ denotes the resulting image from the zeroth order integration of $\mathbf{I}[x,y]$ k times with respect to x and l times with respect to y.

Linear moments were first introduced in [3] where, using a rather involved mathematical formulation, they were used as an intermediate step to efficiently compute geometric moments. In [4], a simpler mathematical development led to the same connection, as well as a new connection with Fourier coefficients.

It has been proved that each element of $\mathbf{L}$ can be expressed in terms of the original image, based on its definition as zeroth order integration values at point $[a,b]$ (see [4] for details):

$$\mathbf{L}[k,l] = \sum_{r=1}^{a}\sum_{s=1}^{b} \binom{b-s+k-1}{b-s} \mathbf{I}[s,r] \binom{a-r+l-1}{a-r}. \quad (2)$$

It can also be proved that the linear moments can be related to the geometric moments and Fourier coefficients in a straightforward way by matrix products. Moreover, they can be efficiently obtained by means of accumulative filters, that only requires additions (again, see [4] for more details). Figure 1a represents this idea, which essentially consists on integrating each raw of the image m times using filter $H_x(z)$ (*fig.* 1b) and integrating the last column of the integrated image n times using $H_y(x)$ (*fig.* 1c).

3. Scalar Recursive Updating Rule

It can be seen, directly from figure 1, that $\mathbf{L}[k,l]$ can be obtained from previous integrated images, either $\mathbf{I}_{x^{k-1}y^l}[x,y]$,

$$\mathbf{L}[k,l] = \sum_{j_1=1}^{b} \mathbf{I}_{x^{k-1}y^l}[a,j_1], \quad (3)$$

or $\mathbf{I}_{x^ky^{l-1}}[x,y]$,

$$\mathbf{L}[k,l] = \sum_{i_1=1}^{a} \mathbf{I}_{x^ky^{l-1}}[i_1,b]. \quad (4)$$

Likewise, $\mathbf{I}_{x^{k-1}y^l}[x,y]$ and $\mathbf{I}_{x^ky^{l-1}}[x,y]$ can be obtained from the previous integrated images $\mathbf{I}_{x^{k-2}y^l}[x,y]$, $\mathbf{I}_{x^{k-1}y^{l-1}}[x,y]$ and $\mathbf{I}_{x^ky^{l-2}}[x,y]$. Then, $\mathbf{L}[k,l]$ can be expressed as:

$$\mathbf{L}[k,l] = \sum_{j_2=1}^{b}\sum_{j_1=1}^{j_2} \mathbf{I}_{x^{k-2}y^l}[a,j_1], \quad (5)$$

if l integrations with respect to y have already been obtained, or:

$$\mathbf{L}[k,l] = \sum_{i_2=1}^{a}\sum_{i_1=1}^{i_2} \mathbf{I}_{x^ky^{l-2}}[i_1,b], \quad (6)$$

if k integrations with respect to x have already been obtained, or:

$$\mathbf{L}[k,l] = \sum_{j_1=1}^{b}\sum_{i_1=1}^{a} \mathbf{I}_{x^{k-1}y^{l-1}}[i_1,j_1], \quad (7)$$

if one integration in both directions is carried out.

If these operations are iteratively repeated and only one-direction integrations are considered, it can be checked that

$$\mathbf{L}[k,l] = \sum_{j_k=1}^{b}\sum_{j_{k-1}=1}^{j_k}\cdots\sum_{j_1=1}^{j_2} \mathbf{I}_{x^0y^l}[a,j_1], \quad (8)$$

or

$$\mathbf{L}[k,l] = \sum_{i_l=1}^{a}\sum_{i_{l-1}=1}^{i_l}\cdots\sum_{i_1=1}^{i_2} \mathbf{I}_{x^ky^0}[i_1,b]. \quad (9)$$

Both expressions, (8) and (9), will be convenient to obtain a recursive updating rule when sliding a window in either the positive x or y direction, respectively.

Let $W[i,j]$ be an odd-dimensioned window centered at point $[x,y]$, where $i = 1, \ldots, w_a$ and $j = 1, \ldots, w_b$. The linear moment matrix associated with this region, using (8) or (9) is:

$$\mathbf{L}^x[k,l] = \sum_{j_k=1}^{w_b}\sum_{j_{k-1}=1}^{j_k}\cdots\sum_{j_1=1}^{j_2} \mathbf{I}_{x^0y^l}[x+\frac{w_a}{2}, y-\frac{w_b}{2}+j_1] \quad (10)$$

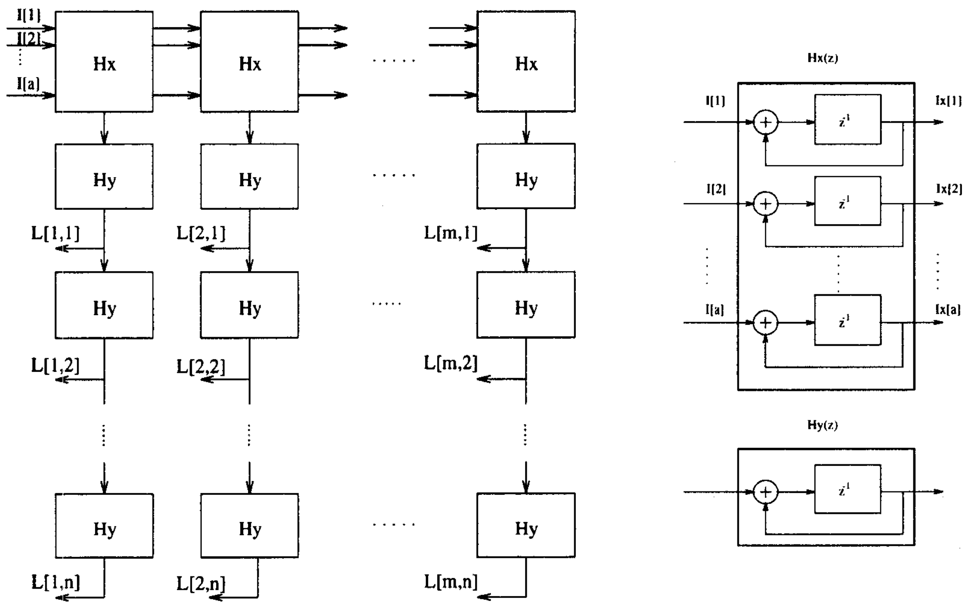

Fig. 1 *Determination of the elements of* **L**, *directly from the original image, using only accumulation filters.*

or

$$\mathbf{L}^{y}[k,l]=\sum_{i_l=1}^{w_a}\sum_{i_{l-1}=1}^{i_l}\cdots\sum_{i_1=1}^{i_2}\mathbf{I}_{x^k y^0}[x-\frac{w_a}{2}+i_1,y+\frac{w_b}{2}], \quad (11)$$

respectively. The linear moment matrix obtained when sliding the window $W[i,j]$ one pixel along the positive x direction is:

$$\mathbf{L}^{x+1}[k,l]=$$
$$\sum_{j_k=1}^{w_b}\sum_{j_{k-1}=1}^{j_k}\cdots\sum_{j_1=1}^{j_2}\mathbf{I}_{x^0 y^l}[x+\frac{w_a}{2},(y+1)-\frac{w_b}{2}+j_1] \quad (12)$$

or

$$\mathbf{L}^{y+1}[k,l]=$$
$$\sum_{i_l=1}^{w_a}\sum_{i_{l-1}=1}^{i_l}\cdots\sum_{i_1=1}^{i_2}\mathbf{I}_{x^k y^0}[(x+1)-\frac{w_a}{2}+i_1,y+\frac{w_b}{2}], \quad (13)$$

when sliding it along the y axis.

A simple algebraic manipulation of (12) allows us to express it using only three adding terms:

$$\begin{aligned}\mathbf{L}^{x+1}[k,l] &= \mathbf{L}^{x}[k,l]+\\ &\quad \mathbf{L}^{x+1}[k-1,l]-\\ &\quad \mathbf{b}[k]\cdot\mathbf{I}_{x^0 y^l}[x+\frac{w_a}{2},y-\frac{w_b}{2}], \end{aligned}\quad (14)$$

where

$$\mathbf{b}[k]=\sum_{j_k=1}^{w_b}\sum_{j_{k-1}=1}^{j_k}\cdots\sum_{j_2=1}^{j_3}1=\begin{pmatrix} w_b+k-2\\ k-1\end{pmatrix}, \quad (15)$$

with $k=1,\ldots,m$, and $\mathbf{I}_{x^0y^l}[x+\frac{w_a}{2},y-\frac{w_b}{2}]$ being the last component of vector $\mathbf{I}[i,y-\frac{w_b}{2}]$, i ranging from $(x-\frac{w_a}{2})$ to $(x+\frac{w_a}{2})$, integrated l times.

The boundary values $\mathbf{L}^{x+1}[0,l]$, $l=1,\ldots,n$, are:

$$\mathbf{L}^{x+1}[0,l]:=\mathbf{I}_{x^0y^l}[x+\frac{w_a}{2},(y+1)+\frac{w_b}{2}]\ ,\ l=1,\ldots,n. \quad (16)$$

A similar development leads the following expression for $\mathbf{L}^{y+1}[k,l]$:

$$\begin{aligned}\mathbf{L}^{y+1}[k,l] &= \mathbf{L}^{y}[k,l]+\\ &\quad \mathbf{L}^{y+1}[k,l-1]-\\ &\quad \mathbf{a}[l]\cdot\mathbf{I}_{x^k y^0}[x-\frac{w_a}{2},y+\frac{w_b}{2}], \end{aligned}\quad (17)$$

where

$$\mathbf{a}[l]=\sum_{i_l=1}^{w_a}\sum_{i_{l-1}=1}^{i_l}\cdots\sum_{i_2=1}^{i_3}1=\begin{pmatrix} w_a+l-2\\ l-1\end{pmatrix}, \quad (18)$$

with $l=1,\ldots,n$ and boundary values

$$\mathbf{L}^{y+1}[k,0]:=\mathbf{I}_{x^ky^0}[(x+1)+\frac{w_a}{2},y+\frac{w_b}{2}]\ ,\ k=1,\ldots,m. \quad (19)$$

Expressions (14) and (17), that we call *scalar updating rule* can also be expressed directly in terms of $\mathbf{L}^x$ and $\mathbf{L}^{x+1}$, or $\mathbf{L}^y$ and $\mathbf{L}^{y+1}$, as shown in the next section.

4. An Explicit Matrix Reformulation

Let $\mathbf{L}[k]$ denote the kth row vector of matrix $\mathbf{L}$ and $\mathbf{v}_l^x$ an l-dimensional vector whose elements are the last component of the outgoing vector integrated up to order l. Likewise, the corresponding integrated values of the incoming vector are denoted by $\mathbf{v}_l^{x+1}$.

Then, (14) can be expressed as:

$$\begin{pmatrix} \mathbf{L}^{x+1}[1] \\ \mathbf{L}^{x+1}[2] \\ \vdots \\ \mathbf{L}^{x+1}[m] \end{pmatrix} = \begin{pmatrix} \mathbf{L}^{x}[1] \\ \mathbf{L}^{x}[2] \\ \vdots \\ \mathbf{L}^{x}[m] \end{pmatrix} + \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & & 0 & 0 \\ 0 & 1 & 0 & \vdots & 0 \\ \vdots & & & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{L}^{x+1}[1] \\ \mathbf{L}^{x+1}[2] \\ \vdots \\ \mathbf{L}^{x+1}[m] \end{pmatrix} - \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \cdot \mathbf{v}_l^x + \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \cdot \mathbf{v}_l^{x+1}. \quad (20)$$

Hence, the updating rule can be reformulated as follows:

$$\mathbf{L}^{x+1} = \mathbf{U}_m \cdot \mathbf{L}^x - \mathbf{U}_m \cdot \mathbf{b}^t \cdot \mathbf{v}_l^x + \mathbf{U}_m \cdot \mathbf{d}^t \cdot \mathbf{v}_l^{x+1}, \quad (21)$$

where $\mathbf{U}$ is an $m \times m$ lower triangular matrix, whose elements are all 1, and $\mathbf{d}$ is an m-dimensional vector whose first component is 1 and all others are 0.

A similar reasoning, when the window is shifted along the y-direction, leads to:

$$\mathbf{L}^{y+1} = \mathbf{L}^y \cdot \mathbf{U}_n^t - \mathbf{v}_k^{y^t} \cdot \mathbf{a}^t \cdot \mathbf{U}_n^t + \mathbf{v}_k^{y+1^t} \cdot \mathbf{d}^t \cdot \mathbf{U}_n^t. \quad (22)$$

5. Computational Cost

We will first evaluate the computational cost of the scalar updating rule (14). Each shifting in the positive x direction will require n boundary values, $\mathbf{L}^{x+1}[0, l], l = 1, \ldots, n$, which involves n zeroth order integrations of a w_a-dimensional vector. Each integration requires $(w_a - 1)$ additions. Thus, the total number of additions required to obtain the whole boundary values is $n(w_a - 1)$. Moreover, n zeroth order integrations are required from the outgoing vector, leading to $n(w_a - 1)$ more additions. Since matrix $\mathbf{L}$ has $m \times n$ elements, each one obtained by means of 2 additions and 1 multiplication, the required number of additions and multiplications are $2n(w_a - 1) + 2mn$ and mn, respectively.

A similar reasoning applied to (17) leads to $2m(w_b - 1) + 2mn$ additions and mn multiplications.

The number of additions required to obtain the initial linear moment matrix is $mw_a(w_b - 1) + mn(w_a - 1)$, as stated in [4].

Taking into account the whole image, $\mathbf{I}[x, y]$ where $x = 1, \ldots, a$ and $y = 1, \ldots, b$, the total number of additions and multiplications are:

$$a(b-1)[2n(w_a - 1) + 2mn]+$$
$$(a-1)[2m(w_b - 1) + 2mn]+$$
$$[mw_a(w_b - 1) + mn(w_a - 1)]$$

and $(ab - 1)mn$, respectively.

Now, the computational cost of the explicit matrix reformulation, (21), is evaluated. In this case, the required number of additions and multiplications is:

$$2n(w_a - 1) + 2mn + \frac{(m-1)(m-2)}{2}$$

and mn, respectively. Table I compiles all these complexity results.

Although the computational cost is higher when using the matrix formulation, it could be convenient for specific vectorial implementations.

Finally, note that the computational cost of the global recursive calculation of the linear moment matrix is $O(abm^2)$, in front of the global cost required to obtain it directly, which is $O(abm^3)$. It clearly comes up the advantage of the recursive implementation.

6. Conclusions

Most image analysis techniques use a window over a region to derive a description vector. Examples of this can be found within many texture segmentation algorithms [1, 5], optic flow computation [2], etc. The considered window may be placed at different positions in an image to detect regions with similar description vectors. Many approaches use a sliding odd-dimensioned window. This approach has been recognized to be computationally expensive since a description vector is obtained for each pixel of the image. Alternatively, other approaches simply use perfectly aligned windows to reduce computational overhead. The obtained results are obviously worse than those using the former approach, since all pixels inside one of those window are assumed to have the same description vector.

	Additions	*Multiplications*
L_initial_matrix	$mw_a(w_b-1)+mn(w_a-1)$	-
x_recursive_formulation	$2n(w_a-1)+2mn$	mn
y_recursive_formulation	$2m(w_b-1)+2mn$	mn
Global cost of the recursive formulation	$2a(b-1)(w_a-1)n+2(a-1)(w_b-1)m+$ $+2mn(2ab+w_a-3)+mw_a(w_b-1)$	$mn(ab-1)$
x_matrix_formulation	$2n(w_a-1)+2mn+\frac{m(m-1)}{2}$	mn
y_matrix_formulation	$2m(w_b-1)+2mn+\frac{n(n-1)}{2}$	mn
Global cost of the matrix formulation	$2a(b-1)(w_a-1)n+2(a-1)(w_b-1)m+$ $+2mn(2ab+w_a-3)+mw_a(w_b-1)+$ $+\frac{am(m-1)(b-1)+n(n-1)(a-1)}{2}$	$mn(ab-1)$

Table I *Complexity results*

Geometric moments and discrete Fourier spectrum coefficients, which have been typically used as description vectors, can be easily obtained from linear moments through simple matrix calculations. Thus, it seems reasonable not only focus on their efficient computation – as described in [4]–, but also on their direct use as descriptors – as assumed here.

If linear moments are used as descriptors, it is shown that the performance of the sliding-window approach can be greatly improved. The key point of this improvement has been to realize that the formulation of the linear moment matrix, in terms of only one-direction integrations, leads to an efficient recursive updating rule.

7. References

[1] Y-Q. Chen, M.S. Nixon and D.W. Thomas, "Statistical Geometrical Features for Texture Classification," *Pattern Recognition*, Vol. 28, pp. 537-552, 1995.

[2] S. Ghosal and R. Mehrotra, "Robust Optical Flow Estimation," *Proc. of the 1st IEEE Int. Conf. on Image Processing*, Part. 2, pp. 780-784, 1994.

[3] B. Li, "High-Order Moment Computation of Grey-Level Images," *IEEE Trans. on Image Processing*, Vol. 4, No. 4, pp. 502-505, April 1995.

[4] J. Martínez and F. Thomas, "A Reformulation of the Gray-Level Image Geometric Moments Computation for Real-Time Applications," *IEEE Proc. of the Int. Conf. on Robotics and Automation*, Minneapolis, April 1996.

[5] B.J. Super and A.C. Bovik, "Shape from Texture Using Local Spectral Moments," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 4, pp. 333-343, 1995.