# A Reformulation of the Gray-Level Image Geometric Moments Computation for Real-Time Applications*

Judit Martínez and Federico Thomas

Institut de Cibernètica (CSIC – UPC)
Diagonal 647, 2 planta
08028 Barcelona, Spain

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

## Abstract

*Geometric moments have been successfully used in many robotics applications involving pattern recognition and object pose determination; however, their computation is too expensive, which limits their use in real-time tasks such as visual servoing.*

*This paper describes a new formulation of the geometric moments computation of a 2-D grey-level image. It is shown that our final result is equivalent, from the complexity point of view, to the one provided in a very recently published paper by B. Li. In contrast to this recent approach, ours is surprisingly simple, leading to a very compact formulation of the problem. The key point has been to reformulate the problem in the continuous domain and adapt the final result to the discrete domain, instead of working it out directly in a discrete setting.*

*Finally, our method is compared with the direct approach showing that it efficiently reduces both addition and multiplication complexity, allowing high order moments be used in real time applications.*

## 1 Introduction

Two-dimensional moments have been widely used in computer vision. Typical examples of applications involving moments are pattern recognition, edge detection, orientation determination, image normalization, image reconstruction, texture classification, and visual servoing (see [9] for a recent survey).

The *geometric moment* of order $(m, n)$th of a 2-D function $f(x, y)$, which vanishes outside the domain $0 \leq x \leq a$, and $0 \leq y \leq b$, is defined as

$$\mu_{mn}\left(f(x,y)\right) := \int_0^b \int_0^a f(x,y) x^m y^n \, dxdy. \qquad (1)$$

This is a two-dimensional transformation of the function $f(x, y)$ with the basis functions represented by monomials $x^m y^n$. Choosing a polynomial basis function $m_m(x)m_n(y)$ where

$$m_m(t) = \sum_{k=0}^{m} c_k t^k \qquad (2)$$

results in a polynomial transformation of the form

$$\mu'_{mn}\left(f(x,y)\right) = \int_0^b \int_0^a f(x,y) m_m(x) m_n(y) dxdy. \quad (3)$$

Then, the new transform coefficients, $\mu'_{mn}$, can be computed as a linear combination of geometric moments. Two cases of interest arise when the polynomials are Legendre and Zernike polynomials (see [10] for details).

Some non-linear combinations of geometric moments lead to the so-called *moment invariants* [ 8, 7, 3]. They are used as parameters which remain invariant under scale, pose, contrast and reflection transformations. Other invariants, such as those obtained from generalized Fourier descriptors, can also be derived from geometric moments [5].

Thus, it is clear that geometric moments have attracted much interest from different points of views, and this justifies any effort to attain low complexity algorithms for their computation. Nevertheless, as it has

*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).

been recently pointed out [6], this computation still constitutes a challenge, especially for high orders. Actually, the direct implementation of (1) in software leads to very long processing times and is obviously not suitable for real-time applications.

Most of the efforts in the efficient computation of geometric moments have focused in the binary domain. Several fast algorithms have been described for binary images [4, 2]. They are based on simple techniques that decompose the different regions in the image into non-overlapping simpler regions such as one line thick rectangles or triangles.

In the gray-level domain, the first approach that avoids the direct evaluation of (1) was presented by M. Hatamian in [1], where the idea that a 2-D filter with separable impulse response $x^m y^n u(x)u(y)$ can be used to generate the $(m,n)$th-order moment of a digital image was exploited. Recently, B. Li has deepened on this idea [6], providing a new algorithm that compares favorably with the former one, especially for high-order moments. Although we follow a different approach, it will be shown in the next section that our result is equivalent to that from Li's.

This paper is organized as follows. Section II focuses on the computation of the geometric moments given a linear moment matrix, as defined in [6]. Section III describes an efficient algorithm to compute this latter matrix given an image. Section IV deals with the inverse problem, that is, given the moments of an image, it describes how to obtain its linear moment matrix. The operation complexity of the algorithm and its comparison to the direct approach is given in Section V. Finally, Section VI concludes giving some prospects for future research.

## 2 The Linear Moment Matrix

Expression (1) can be rewritten as:

$$\mu_{mn}\left(f(x,y)\right) = \int_0^b y^n dy \int_0^a x^m f(x,y) dx \qquad (4)$$

Thus, integrating the inner term by parts, we get:

$$\mu_{mn}\left(f(x,y)\right) = a^m \int_0^b y^n f_{x^1}(a,y) dy - m\mu_{(m-1)n}\left(f_{x^1}(x,y)\right). \qquad (5)$$

where $f_{x^i y^j}(x,y)$ denotes the resulting image from the integration of $f(x,y)$ $i$ times with respect to $x$ and $j$ times with respect to $y$.

The integration by parts of the resulting integral term leads to:

$$\mu_{mn}\left(f(x,y)\right) = a^m b^n f_{x^1y^1}(a,b) - a^m n \int_0^b y^{n-1} f_{x^1y^1}(a,y) dy - m\mu_{(m-1)n}[f_{x^1}(x,y)]. \qquad (6)$$

Then, if this operation is repeated up to the elimination of the integral term, and the resulting moments recursively expressed in terms of lower order moments, it can be easily checked that:

$$\mu_{mn}\left(f(x,y)\right) = \sum_{i=0}^{m} a^{m-i} \frac{m!}{(m-i)!} \sum_{j=0}^{n} (-1)^{i+j} b^{n-j} \frac{n!}{(n-j)!} f_{x^{(i+1)}y^{(j+1)}}(a,b). \qquad (7)$$

Moreover, this result can be further simplified if expressed in matrix terms as follows:

$$\mu_{mn} = \mathbf{a}_m^t \begin{pmatrix} f_{x^1y^1}(a,b) & \dots & f_{x^1y^{n+1}}(a,b) \\ \vdots & & \vdots \\ f_{x^{m+1}y^1}(a,b) & \dots & f_{x^{m+1}y^{n+1}}(a,b) \end{pmatrix} \mathbf{b}_n, \qquad (8)$$

where

$$\mathbf{a}_m^t = a^m m!\left(\frac{1}{m!}, \dots, \frac{(-a)^{-i}}{(m-i)!}, \dots, (-a)^{-m}\right) \qquad (9)$$

and

$$\mathbf{b}_n^t = b^n n!\left(\frac{1}{n!}, \dots, \frac{(-b)^{-j}}{(n-j)!}, \dots, (-b)^{-n}\right) \qquad (10)$$

are constant vectors, i.e. independent from the contents of the image, so that, for a given application, they can be precomputed. Thus, we can concentrate ourselves on the efficient computation of:

$$\mathbf{L} := \begin{pmatrix} f_{x^1y^1}(a,b) & \dots & f_{x^1y^{n+1}}(a,b) \\ \vdots & & \vdots \\ f_{x^{m+1}y^1}(a,b) & \dots & f_{x^{m+1}y^{n+1}}(a,b) \end{pmatrix}. \qquad (11)$$

A deeper insight reveals that matrix $\mathbf{L}$ is equivalent to the *linear moment matrix*, as defined in [6]. Actually, Hatamian's, Li's and our method only differ in the way the geometric moments are obtained from this matrix. Hatamian did not provide a general expression for this. His results were limited to third order moments. Li, using a rather involved mathematical machinery, obtained a computatively more efficient method but he also failed to obtain an explicit expression for the elements of $\mathbf{a}_m$ and $\mathbf{b}_n$, which had to be computed by multipliying a variable sequence of matrices.

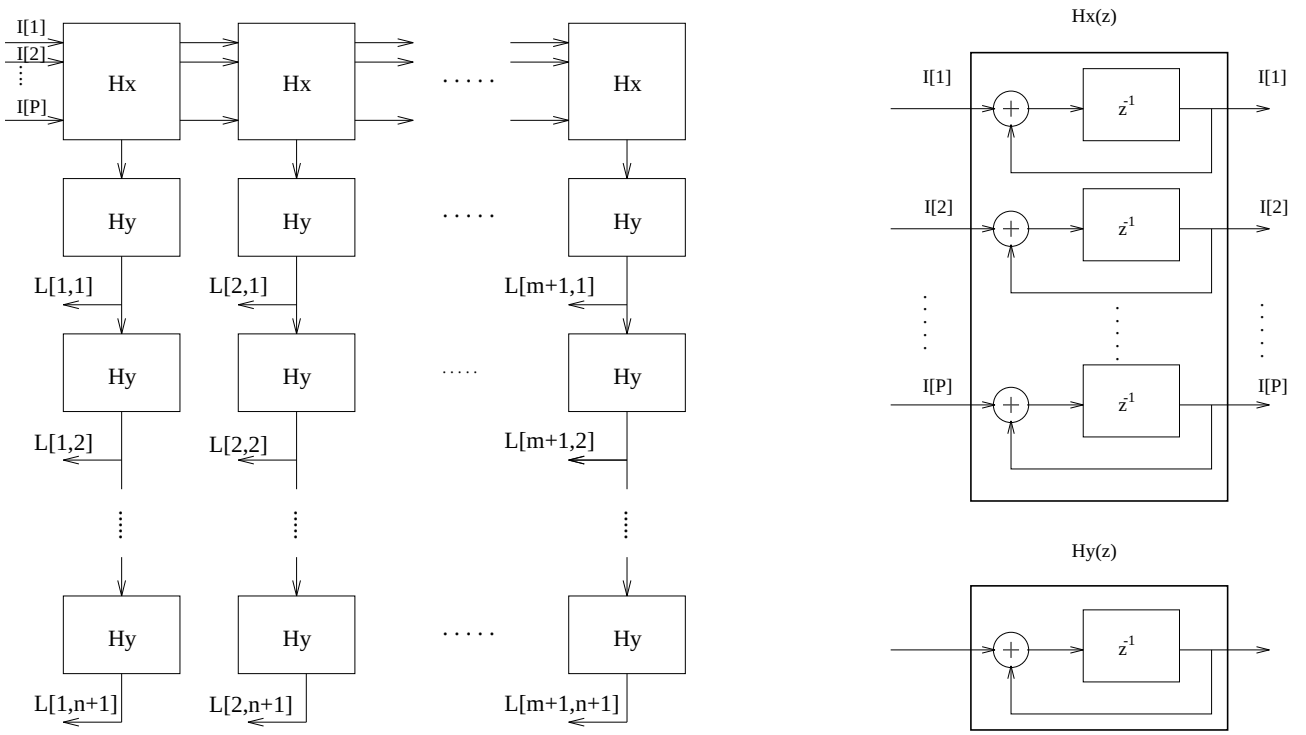



**Fig. 1** *Determination of the elements of* **L**, *directly from the original image, using only accumulation filters.*

# 3 Computation of L

The formulation of the problem in the continuous domain has allowed us to obtain a simple formulation in a straightforward way. Now, this has to to be adapted to the discrete domain, so that, instead of dealing with a two dimensional function $f(x, y)$ we deal with a two dimensional array, $\mathbf{I}[x, y]$, $x = 1, \ldots, P$, $y = 1, \ldots, Q$, or image.

To obtain **L**, the image has to be integrated and evaluated at $[P, Q]$, a number of times that depends on the order of the moment required.

In the discrete domain, it can be easily shown that **L** can be expressed directly in terms of the original image as $\mathbf{L} = \mathbf{W}_1\ \mathbf{I}\ \mathbf{W}_2$, where $\mathbf{W}_1$ and $\mathbf{W}_2$ are constant matrices. Although this formulation might be of interest for some applications, **L** can also be obtained, much more efficiently, using accumulation filters, so that only additions will be required. Figure 1a represents this idea, which essentially consists on integrating each raw of the image $(m + 1)$ times using filter $H_x(z)$ (*fig.* 1b) and integrating the last column of the integrated image $(n + 1)$ times using $H_y(x)$ (*fig.* 1c).

# 4 The Inverse Problem

If we define the matrix of moments as

$$\mathbf{M}_{mn} := \begin{pmatrix} \mu_{00} & \cdots & \mu_{0n} \\ \vdots & & \vdots \\ \mu_{m0} & \cdots & \mu_{mn} \end{pmatrix}, \tag{12}$$

then

$$\mathbf{M}_{mn} = \mathbf{A}\ \mathbf{L}\ \mathbf{B}, \tag{13}$$

where **A** is an $(m+1) \times (m+1)$ lower triangular matrix and **B**, an $(n+1) \times (n+1)$ upper triangular matrix obtained from (9) and (10), respectively. It can be checked that

$$\mathbf{A}[i, j] = \begin{cases} (-1)^{j+1} \begin{pmatrix} i-1 \\ i-j \end{pmatrix} a^{i-j}(j-1)!, & i \geq j \\ 0 \quad \text{otherwise} \end{cases}, \tag{14}$$

and

$$\mathbf{B}[i, j] = \begin{cases} (-1)^{i+1} \begin{pmatrix} j-1 \\ j-i \end{pmatrix} b^{j-i}(i-1)!, & j \geq i \\ 0 \quad \text{otherwise} \end{cases}. \tag{15}$$

Given a matrix of moments, $\mathbf{M}_{mn}$, the problem of computing the corresponding linear moment matrix satisfying (13), that is $\mathbf{A}^{-1}\mathbf{M}_{mn}\mathbf{B}^{-1}$, is here defined as the inverse problem. The gaussian inversion algorithm allows us to obtain the following expressions for $\mathbf{A}^{-1}$ and $\mathbf{B}^{-1}$:

$$\mathbf{A}^{-1}[i,j] = \begin{cases} (-1)^{j+1} \begin{pmatrix} i-1 \\ j-1 \end{pmatrix} \frac{a^{i-j}}{(i-1)!}, & i \geq j \\ 0 \quad \text{otherwise} \end{cases}, \tag{16}$$

and

$$\mathbf{B}^{-1}[i,j] = \begin{cases} (-1)^{i+1} \begin{pmatrix} j-1 \\ i-1 \end{pmatrix} \frac{b^{j-i}}{(j-1)!}, & j \geq i \\ 0 \quad \text{otherwise} \end{cases}. \tag{17}$$

This leads to an alternative approach to solve the *problem of moments*: Given a set of moments up to a given order, find an image that best fit these moments according to a given error function. The usual way to solve this problem consists in describing the unknown image as a sum of bidimensional orthogonal polynomials. Then, the problem becomes that of obtaining the coefficients of these polynomials [10].

# 5 Complexity

To compare the complexity of our algorithm in terms of number of products and additions to the direct algorithm, we consider a square image of size $P$. Then, the discrete counterpart of (1) becomes:

$$\mu_{mn} = \sum_{i=1}^{P}\sum_{j=1}^{P} i^m j^n \mathbf{I}[i,j] = \sum_{i=1}^{P} i^m \sum_{j=1}^{P} j^n \mathbf{I}[i,j]. \tag{18}$$

This suggests the following matrix representation:

$$\mu_{mm} = \begin{pmatrix} 1^m & \cdots & P^m \end{pmatrix} \mathbf{I} \begin{pmatrix} 1^n \\ \vdots \\ P^n \end{pmatrix}. \tag{19}$$

Then, when considering moments up to order $(m,m)$, we have that

$$\mathbf{M}_{mm} = \mathbf{V}^t \, \mathbf{I} \, \mathbf{V}, \tag{20}$$

where $\mathbf{V}$ is a Vandermonde matrix of the form:

$$\mathbf{V} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 2^2 & & 2^m \\ \vdots & & & & \vdots \\ 1 & P & P^2 & \cdots & P^m \end{pmatrix}, \tag{21}$$

which can be precalculated. Thus, the global cost in the direct case corresponds to the cost of computing (20), neglecting the exponentials factors involved in $\mathbf{V}$. The required number of multiplications and additions are:

$$P^2 m + P m^2 \tag{22}$$

and

$$P(P-1)m + (P-1)m^2 \tag{23}$$

respectively.

In our method, as well as in [6], the number of multiplications is independent from the size of the image. The only multiplications arise when computing $\mathbf{A}\,\mathbf{L}\,\mathbf{B}$. Since $\mathbf{A}$ and $\mathbf{B}$ are triangular matrices the required number of multiplications and additions are:

$$(m+1)^2(m+2) \tag{24}$$

and

$$m^2(m+1) \tag{25}$$

respectively.

Taking into account the cost of obtaining $\mathbf{L}$, using the accumulation filters, the total number of additions is

$$m^2(m+1) + P^2(m+1) + P(m+1). \tag{26}$$

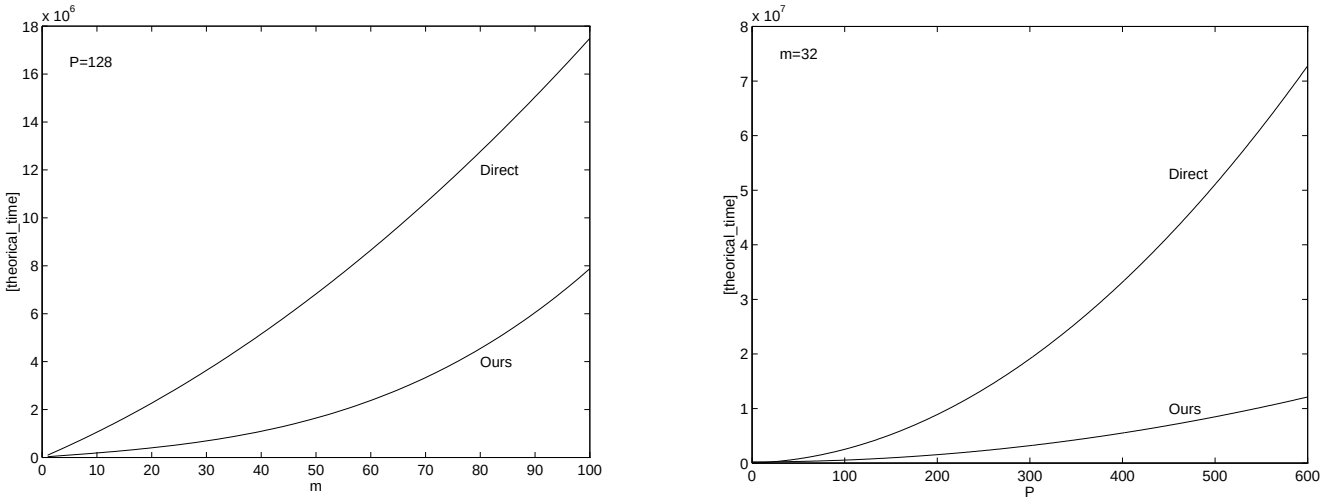
The following table compiles these results:

| *Method* | *Additions* | *Multiplications* |
|---|---|---|
| Direct | $P(P-1)m + (P-1)m^2$ | $P^2 m + P m^2$ |
| Ours | $P^2(m+1) + P(m+1) + m^2(m+1)$ | $(m+1)^2(m+2)$ |

Figure 2 contains the theoretical time comparison between the direct method and ours. The comparison is done for different orders of the moments (*fig.* 2a) and different sizes of the image (*fig.* 2b). We have considered the very favorable case in which a multiplication operation takes five times the time required for an addition.

One relevant feature of our formulation is that it can be easily described as an incremental process. Actually, this is carried out by simply adding two new vectors $a^t_{m+1}$ and $b_{m+1}$ to matrix $A$ and $B$ respectively, and iterating the image filtering process.

Increasing the maximum moment order from $\mu_{mm}$ to $\mu_{(m+1)m}$ results in integrating the image once more in the $x-$direction and evaluating at $[P,P]$ the $(m+1)$ integrals, of this latter image, with respect to the $y-$direction. This is equivalent to add one more column of filters to the rightmost part of the array shown in *fig.* 1a). Thus, $P^2 + P(m+1)$ additions are required to update $\mathbf{L}$. If the order is increased from $\mu_{mm}$ to $\mu_{m(m+1)}$, one more row of filters must be added at the bottom of the array. Then, only $(m+1)P$ additions would be required.




**(a)** **(b)**

**Fig. 2** *Time comparisons between the direct method and ours.*

The only information we must store to proceed with this incremental formulation is the last column of last integrated images with respect to $y$ and the last integrated image with respect to $x$ (see *fig.* 1a to follow this reasoning). This amounts to $P^2 + (m+1)P$ total memory locations.

# 6 Conclusions

In this paper, we have proposed a new formulation for the grey-level image geometric moments computation whose most relevant peculiarity is its simplicity without efficiency detrimental. It has been proved to be equivalent, from the complexity point of view, to Li's algorithm, the most efficient one known until now.

The presented algorithm is under implementation and the preliminary results fit the theoretical expectations. Hopefully, these results would be included in the final paper.

The moments problem is a point for future research that deserves further attention. It has been shown that, given a matrix of moments, the corresponding linear moment matrix can be computed. The efficient computation of the original image from this matrix, without relying on the computation of orthogonal polynomial series, would have important applications to decimation and interpolation of images, non-linear filtering and even image segmentation.

# References


[1] M. Hatamian, "A Real-Time Two-Dimensional Moments Generating Algorithm and Its Single Chip Implementation," *IEEE Trans. On Acoustics, Speech and Signal Processing*, Vol. ASSm-34, No. 3, pp. 546-553, June 1986.

[2] X.Y. Jiang y H. Bunke, "Simple and Fast Computation of Moments," *Pattern Recognition*, Vol. 24, No. 8, pp. 801-806, 1991.

[3] C.-Y. Lee, and D.B. Cooper, "Structure from Motion: A Region Based Approach Using Affine Transformations and Moment Invariants," *IEEE Int. Conf. on Robotics and Automation*, pp. 120-127, 1993.

[4] J.-G. Leu, "Computing a Shape's Moments from its Boundary," *Pattern Recognition*, Vol. 24, No. 10, pp. 949-957, 1991.

[5] B. Li, and S.D. Ma, "On the Relation between Region and Contour Representations," $12^{th}$ *IAPR Int. Conference on Pattern Recognition*, Jerusalem, Israel, October 1994.

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

[7] V. Markandey, and R.J.P. deFigueiredo, "Robot Sensing Techniques Based on High-Dimensional Moment Invariants and Tensors," *IEEE Trans. on Robotics and Automation*, Vol. 8, No. 2, pp. 186-194, 1992.

[8] T.H. Reiss, "The Revised Fundamental Theorem of Moment Invariants," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 8, pp. 830-834, 1991.

[9] R.J. Prokop, and A.P. Reeves, "A Survey of Moment-Based Techniques for Unoccluded Object Representation and Recognition," *CVGIP: Graphical Models and Image Processing*, Vol. 54, No. 5, pp. 438-460, 1992.

[10] M.R. Teague, "Image Analisys Via the General Theory of Moments," *J. Opt. Soc. Amer.*, Vol. 70, pp. 920-930, August 1980.