Efficient Computation of 3-D Moments in Terms of an Object's Partition

Juan Humberto Sossa Azuela[1], Francisco Cuevas de la Rosa[2,*] and Héctor Benitez[1]

[1] Centro de Investigación en Computación del IPN
Av. Juan de Dios Bátiz esquina con M. Othón de Mendizábal
Colonia Nueva Industrial Vallejo, México, D. F. 07738, México
[2] Centro de Investigaciones en Óptica+
Apdo. Postal 1-948, León, Gto. México
{hsossa,fjcuevas}@cic.ipn.mx
hbenitez75@hotmail.com

Abstract. The method here proposed is based on the idea that the object of interest is first decomposed in a set of cubes under d_∞. This decomposition is known to form a partition. The required moments are computed as a sum of the moments of the partition. The moments of each cube can be computed in terms of a set of very simple expressions using the center of the cube and its radio. The method provides integral accuracy by applying the exact definition of moments over each cube of the partition. One interesting feature of our proposal is that once the partition is obtained, moment computation is faster than with earlier methods.

1 Introduction

The two-dimensional moment (for short 2D moment) of a 2D object R is defined as [1]:

$$M_{pq} = \iint_R x^p y^q f(x,y)\,dxdy \tag{1}$$

where $f(x,y)$ is the characteristic function describing the intensity of R, and $p+q$ is the order of the moment. In the discrete case, the double integral is often replaced by a double sum giving as a result:

$$m_{pq} = \sum\sum_R x^p y^q f(x,y) \tag{2}$$

with $f(x,y)$, *p and q* defined in equation 1, where $(x,y) \in \mathbf{Z}^2$.

---

* Francisco Cuevas is in a post-doctoral stay at the Centro de Investigación en Computación of the Instituto Politécnico Nacional.

T. Caelli et al. (Eds.): SSPR&SPR 2002, LNCS 2396, pp. 301-309, 2002.

The tri-dimensional geometric moment (for short 3D moment) of order $p+q+r$ of a 3D object is defined as [2]:

$$M_{pqr} = \iiint_R x^p y^q z^r f(x,y,z)\,dxdydz \tag{3}$$

where R is a 3D region. In the discrete case, the triple integral is often replaced by the triple sum giving as a result:

$$m_{pqr} = \sum_R \sum \sum x^p y^q z^r f(x,y,z) \tag{4}$$

with $f(x,y,z)$, $p,q\ y\ r$ defined in equation (3), where $(x,y,z) \in \mathbf{Z^3}$.

In the binary case, the characteristic function takes only values 1 or 0, assuming that for the volume of interest $f(x,y,z)=1$. When we replace this value in equation (4) we get the equation to compute the moments of order $(p+q+r)$ of a 3-D image R as

$$m_{pqr} = \sum_R \sum \sum x^p y^q z^r \tag{5}$$

with $(x,y,z) \in \mathbf{Z^3}$ y $p, q, r = 0,1,2,3,...$

The world around us is generally three-dimensional, and 3D shape information for an object can be obtained by computer tomographic reconstruction, passive 3D sensors, and active range finders. As 2D moments, 3D moments have been used in 3D image analysis tasks including movement estimation [3], shape estimation [4], and object recognition [2].

Several methods have been proposed to compute the 3D moments. In [6], Li uses a polyhedral representation of the object for the computing of its 3D moments. The number of required operations is a function of the number of edges of the surfaces of the polyhedral. The methods of Cyganski *et al.* [5], Li and Shen [7] and Li and Ma [8] use a voxel representation of the object. The difference among these methods is the way to compute the moments. Cyganski *et al.* uses the filter proposed in [9]. Li and Shen use a transformation based on Pascal triangle for the computation of the monomials and only additions are used for the computation of the moments. In the other hand, Li and Ma relate 3D moments with the so-called LT moments that are easier to evaluate. Although these methods allow to reduce the number of operations to compute the moments, they require a computation of $O(N^3)$. Recently, Yang *et al*. [10] propose to use the so called discrete divergence theorem to compute the 3D moments of an object. It allows a reduction in the number of operations to $O(N^2)$.

In this note we present an efficient method to compute the 3-D moments of a binary object in $\mathbf{Z^3}$. The method is an extension of the method recently introduced in [11] to compute the 2D moments of an object. It provides integral accuracy (see [12] for the details) on the values obtained by applying the original definition of moments (equation (3)) instead of that one using triple sums (equation (4) or (5)). This could not happen if equation (5) was used.

The object is first partitioned into convex cubes which moments evaluation can be reduced to the computation of very simple formulae instead of using triple integrals. The desired 3D moments are obtained as the sum of the moments of each cube of the partition, given that the intersection among cubes is empty.

2 Moments of a Cube

In the last section we mentioned that to compute the desired moments of a 3D object, it should be first decomposed into a set of cubes. Then we also said that a set of simple expressions should be applied to get the desired values. In this section, this set of expressions is provided.

Depending on the definition of moments used, the set of expressions obtained might differ resulting in some differences. This situation was first studied in [12] and recently re-discussed in [13], both in the 2-D case. As stated in [13], if M_{pq} are the 2D moments obtained by means of equation (1) and m_{pq} those obtained in terms of equation (2) an error $\left|M_{pq} - m_{pq}\right|$ is introduced due to the approximations and numeric integration of $x^p y^q$ over each pixel. As we will next see, this also happens with 3D moments.

To derive the set of expressions needed to accurately compute the desired 3D moments, let us consider a cube centered in (X_c, Y_c, Z_c), with radius *t* and coordinates of its vertices in $(X_c - t, Y_c - t, Z_c - t)$, $(X_c + t, Y_c - t, Z_c - t)$, $(X_c - t, Y_c + t, Z_c - t)$, $(X_c - t, Y_c - t, Z_c + t)$, $(X_c + t, Y_c + t, Z_c - t)$, $(X_c + t, Y_c - t, Z_c + t)$, $(X_c - t, Y_c + t, Z_c + t)$ and $(X_c + t, Y_c + t, Z_c + t)$. The characteristic function of this block is

$$f(x,y,z) = \begin{cases} 1 & \text{if } (x,y,z) \in (a,b)\times(c,d)\times(e,f) \\ 0 & \text{otherwise} \end{cases}$$

with

$$\begin{aligned} a &= X_c - t - 0.5 \\ b &= X_c + t + 0.5 \\ c &= Y_c - t - 0.5 \\ d &= Y_c + t + 0.5 \\ e &= Z_c - t - 0.5 \\ f &= Z_c + t + 0.5 \end{aligned}$$

According to equation (3), the exact moments of a cube are given as

$$\begin{aligned} M_{pqr} &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q z^r f(x,y,z)\,dx\,dy\,dz \\ &= \frac{1}{(p+1)(q+1)(r+1)}\left[b^{p+1} - a^{p+1}\right]\cdot\left[d^{q+1} - c^{q+1}\right]\cdot\left[f^{r+1} - e^{r+1}\right] \end{aligned} \tag{6}$$

The reader can easily verify that the first 20 expressions for the moments are:

$$
\begin{aligned}
M_{000} &= (2t+1)^3 & M_{100} &= M_{000}X_c \\
M_{010} &= M_{000}Y_c & M_{001} &= M_{000}Z_c \\
M_{200} &= \frac{M_{000}}{3}\left(3X_c^2 + t(t+1) + 0.25\right) & M_{020} &= \frac{M_{000}}{3}\left(3Y_c^2 + t(t+1) + 0.25\right) \\
M_{002} &= \frac{M_{000}}{3}\left(3Z_c^2 + t(t+1) + 0.25\right) & M_{110} &= M_{100}Y_c = M_{000}X_cY_c \\
M_{101} &= M_{100}Z_c = M_{000}X_cZ_c & M_{011} &= M_{001}Y_c = M_{000}Y_cZ_c \\
M_{300} &= M_{000}X_c\left(X_c^2 + t(t+1) + 0.25\right) & & \\
M_{030} &= M_{000}Y_c\left(Y_c^2 + t(t+1) + 0.25\right) & & \\
M_{003} &= M_{000}Z_c\left(Z_c^2 + t(t+1) + 0.25\right) & & \\
M_{120} &= \frac{M_{000}}{3}X_c\left(3Y_c^2 + t(t+1) + 0.25\right) & & \\
M_{210} &= \frac{M_{000}}{3}Y_c\left(3X_c^2 + t(t+1) + 0.25\right) & & \\
M_{102} &= \frac{M_{000}}{3}X_c\left(3Z_c^2 + t(t+1) + 0.25\right) & & \\
M_{201} &= \frac{M_{000}}{3}Z_c\left(3X_c^2 + t(t+1) + 0.25\right) & & \\
M_{012} &= \frac{M_{000}}{3}Y_c\left(3Z_c^2 + t(t+1) + 0.25\right) & & \\
M_{021} &= \frac{M_{000}}{3}Z_c\left(3Y_c^2 + t(t+1) + 0.25\right) & & \\
M_{111} &= M_{000}X_cY_cZ_c & &
\end{aligned}
\tag{7}
$$

The reader can be easily verify that the same set of 20 expressions obtained through equation (5) is the following:

$$
\begin{aligned}
m_{000} &= (2t+1)^3 & m_{100} &= m_{000}X_c \\
m_{010} &= m_{000}Y_c & m_{001} &= m_{000}Z_c \\
m_{200} &= \frac{m_{000}}{3}\left(3X_c^2 + t(t+1)\right) & m_{020} &= \frac{m_{000}}{3}\left(3Y_c^2 + t(t+1)\right) \\
m_{002} &= \frac{m_{000}}{3}\left(3Z_c^2 + t(t+1)\right) & m_{110} &= m_{100}Y_c = m_{000}X_cY_c \\
m_{101} &= m_{100}Z_c = m_{000}X_cZ_c & m_{011} &= m_{001}Y_c = m_{000}Y_cZ_c \\
m_{300} &= m_{000}X_c\left(X_c^2 + t(t+1)\right) & & \\
m_{030} &= m_{000}Y_c\left(Y_c^2 + t(t+1)\right) & & \\
m_{003} &= m_{000}Z_c\left(Z_c^2 + t(t+1)\right) & &
\end{aligned}
\tag{8}
$$

$$m_{120} = \frac{m_{000}}{3} X_c \left(3Y_c^2 + t(t+1)\right)$$

$$m_{210} = \frac{m_{000}}{3} Y_c \left(3X_c^2 + t(t+1)\right)$$

$$m_{102} = \frac{m_{000}}{3} X_c \left(3Z_c^2 + t(t+1)\right)$$

$$m_{201} = \frac{m_{000}}{3} Z_c \left(3X_c^2 + t(t+1)\right)$$

$$m_{012} = \frac{m_{000}}{3} Y_c \left(3Z_c^2 + t(t+1)\right)$$

$$m_{021} = \frac{m_{000}}{3} Z_c \left(3Y_c^2 + t(t+1)\right)$$

$$m_{111} = m_{000} X_c Y_c Z_c$$

One might would like know which accuracy provides the proposed approach. It is know that for pixel or voxel represented objects, the computed moment values have mainly two types of accuracy. One of them is obtained by exactly performing the double or triple sum, given by equation (2) or (4). The another is obtained by assuming that a pixel is a square and a voxel is a cube, and computing the moments as an integral over the area covered by the small pixel squares, or the volume covered by the small cubes. None of the above approaches gives the true values of the moments. It is not possible to obtain the true moment values if digitalization is done. Our proposal provides integral accuracy. This was very well studied in [12] for the 2-D case.

One might think that because each cube has its own center located at (X_c, Y_c, Z_c), the summing of moments computed from different cubes of different centers is not possible. The summing is possible even if each cube has its own center. This is due to M_{pqr} are expressed in terms of t and one or more of the coordinates of the center. These last terms introduce the needed values to compensate the fact that each cube has its own center.

3 Discussion and Comparison

While equation (3) yields exact results, equation (5) provides some moments with small errors due to the zero-order approximation for numerical integration when using sums. We will always find $M_{pqr} \geq m_{pqr}$. The error $M_{pqr} - m_{pqr}$ depends directly on p, q and r. Your can easily verify that:

$$M_{200} - m_{200} = \frac{m_{000}}{12} \qquad M_{020} - m_{020} = \frac{m_{000}}{12} \qquad M_{002} - m_{002} = \frac{m_{000}}{12}$$

$$M_{300} - m_{300} = \frac{m_{100}}{4}. \qquad M_{030} - m_{030} = \frac{m_{100}}{4}. \qquad M_{003} - m_{003} = \frac{m_{100}}{4}.$$

On the other hand, for some moments both methods produce exact results:

$$M_{000} - m_{000} = 0 \qquad M_{100} - m_{100} = 0$$
$$M_{010} - m_{010} = 0 \qquad M_{001} - m_{001} = 0$$
$$M_{110} - m_{110} = 0 \qquad M_{101} - m_{101} = 0$$
$$M_{011} - m_{011} = 0 \qquad M_{111} - m_{111} = 0$$

On the main features of our method is that once the partition is obtained, moment computation is much faster than in the case of earlier methods. For this, let us take the next simple example. Let us suppose that the an object is composed of $N \times N \times N$ pixels, with *t* as its radius.

The number of operations required by one of the fastest methods (for example the method of Yang, Albregtsen and Taxt, [10]) to compute all the moments of order $(p+q+r)$ up to some *K*, let say *K=3* from a discrete image of $N \times N \times N$ pixels is: $2KN^2$ multiplications and $\left(\frac{1}{2}K^2 + \frac{7}{2}K + 3\right)N^2$ additions (for the details, refer to [10]).

The number of operations required by our proposal **once the partition has been obtained** will depend basically on the radius *t* of the object: 26*t* multiplications and 10*t* additions.

4 A Method to Compute the Desired Object Moments

To compute the desired moments we could use the same idea already used in [11], this is :

1. Decompose the object into the union of disjoint cubes.
2. Compute the geometric moments for each of these cubes, and
3. Obtain the final moments as a sum of the moments computed for each cube.

The key problem to apply this idea is how to obtain the desired partition, i.e. the union of disjoint cubes. For this we can use the same morphological approach used in [11] (extended to the 3D case). According to [11] there two main variants to compute the desired moments:

4.1 Method Based on Iterated Erosions

The following method to compute the geometric moments of a 3D object $R \subset Z^3$, using morphological operations is an extension of the one described in [11]. It s composed of the following steps:

1. Initialize 20 accumulators $C_i=0$, for i=1,2,...,20, one for each geometric moment.
2. Make $A=R$ and $B=\{(\pm a, \pm b, \pm c) / a,b,c \in \{-1,0,1\}\}$, *B* is a 3x3x3 pixel neighborhood in $\mathbf{Z}^3$.
3. Assign $A \leftarrow A \ominus B$ iteratively until the next erosion results in ∅ (the null set). The number of iterations of the erosion operation before set ∅ appears, is the

radius r of the maximal cube completely contained in the original region R. The center of this cube is found in set A just before set $\varnothing$ appears.

4. Select one of the points of A and given that the radius r of the maximal cube is known, we use the formulae derived in the last section to compute the moments of this maximal cube, the resulting values are added to the respective 20 accumulators, C_i, for 1,2,3,...,20.
5. Eliminate this ball from region R, and assign this new set to R.
6. Repeat steps 2 to 5 with the new R until it becomes $\varnothing$.

The method just described gives us as a result the true values of the geometric moments of order $(p+q+r) \leq 3$, using only erosions and the formulae developed in Section 2.

4.2 Method Based on Iterated Erosions and Parallel Processing

This method is a brute force method. A considerable enhancement can obtained if steps 4 and 5 are replaced by:

1. Select those points in A at a distance among them greater than $2t$ and use the formulae given by equation (7), to compute the geometric moments of these maximal cubes, and add these values to the respective accumulators.
2. Eliminate the maximal cubes from region R, and assign this new set to R.

The enhancement consists in processing all maximal cubes of the same radius in just a step, coming back to the iterated erosions until the value of the radio t is to be changed. At this step it is very important to verify that the eliminated cubes do not intersect with those just eliminated, for one of the important conditions is that the set of maximal cubes forms a partition of the image. Thus one has to guarantee that these maximal cubes be disjoint sets.

5 Results

Suppose we use the proposed two variants described in the last section to compute the desired object's moments. Because both variants are not designed to work in a conventional computer, the processing times are only significant to compare the method eliminating a cube at the time against the method eliminating, at the same step, all the non-intersecting maximal cubes at the same time.

Both variants were tested on several hundreds of images. All of them are binary and $101 \times 101 \times 101$ voxel sized. These images were obtained by generating at random P touching and overlapping cubes of different sizes inside the $101 \times 101 \times 101$ cubical image. At the beginning all the locations of the $101 \times 101 \times 101$ cube are zero.

The original method takes on average 150 seconds to compute all moments of order $(p+q+r) \leq 3$; while the enhanced method requires only about 25 seconds onto 233 Mhz PC based system to compute the same moments.

6 Conclusions and Present Research

In this note an extended version of the recently proposed method in [11] to compute accurately the 3D geometric moments for a object has been presented. Initially, the object is partitioned in a set of convex cubes whose moment evaluation can be reduced to the computation of very simple formulae.

These expressions were derived from the original definition of moments given by equation (3). This gives more accurate values for the moments. This would not happen if equation (5) would be used. An error is introduced due to zero-order approximation and numeric integration of $x^p y^q z^r$ over each voxel. The resulting shape moments are finally obtained by addition of the moments of each cube forming the partition, giving that the intersections are empty.

As implemented until now the proposed approach is very slow, for and image of $100 \times 100 \times 100$ voxels, 150 seconds in the case of the first variant and 25 in the case of the second variant.

To make our proposal really competitive with classical sequential algorithms we need a better way the obtain the desired partition. Apparently, the fast distance transform (see [14]) could be an excellent option. In this case, the idea here would be to first decompose the image into a set of disjoint cubes by means of the fast tri-dimensional distance transform, which would provide the necessary information of all the maximal cubes covering the image. We would then apply the simple formulae given by equation (7) to obtain the exact moments for each cube. We would finally get the final desired moments of the image by summing the partial results from all the cubes.

One of the huge advantages of the fast distance transform is that it can be efficiently programmed in a sequential machine. At this moment, we are working on the development of a suitable algorithm.

Acknowledgments

The authors would like to thank the CIC-IPN and the CONACYT under project 34880-A for their economical support to develop this work.

References

1. M. K. Hu, Visual pattern recognition by moment invariants, IRE Transactions on Information Theory, 179-187, 1962.
2. C. H. Lo and H. S. Don, 3-D moment forms: Their construction and application to object identification and positioning, IEEE Transactions on Pattern Analysis and Machine Intelligence, 11:1053-1064, 1989.
3. S. C. Pei and L. G. Liou, Using moments to acquire the motion parameters of a deformable object without correspondences, Image Vision and Computing, 12:475-485, 1994.

4. J. Shen and B. C. Li, Fast determination of center and radius of spherical surface by use of moments, in Proceedings of the 8th Scandinavian Conference on Image Analysis, Tromso, Norway, pp. 565-572, 1993.
5. D. Cyganski, S. J. Kreda and J. A. Orr, Solving for the general linear transformation relating 3-D objects from the minimum moments, in SPIE Intelligent Robots and Computer Vision VII, Proceedings of the SPIE, Vol. 1002, pp. 204-211, Bellingham, WA, 1988.
6. B. C. Li, The moment calculation of polyhedra, Pattern Recognition, 26:1229-1233, 1993.
7. B. C. Li and J. Shen, Pascal triangle transform approach to the calculation of 3D moments, CVGIP: Graphical Models and Image Processing, 54:301-307, 1992.
8. B. C. Li and S. D. Ma, Efficient computation of 3D moments, in Proceedings of 12 the International Conference on Pattern Recognition, Vol 1, pp. 22-26, 1994.
9. Z. L. Budrikis and M. Hatamian, Moment calculations by digital filters, AT&T Bell Lab. Tech. J. 63:217-229, 1984.
10. L. Yang and F. Albregtsen and T. Taxt, Fast computation of three-dimensional geometric moments using a discrete divergence theorem and a generalization to higuer dimensions, CGVIP: Graphical models and image processing, 59(2):97-108, 1997.
11. H. Sossa, C. Yañez and J. L Díaz, Computing geometric moments using morphological erosions, Pattern Recogntition, 34(2), 2001.
12. M. Dai, P. Baylou and M. Najim, An efficient algorithm for computation of shape moments from run-length codes or chain codes, Pattern Recognition, 25(10):1119-1128, 1992.
13. J. Flusser, Refined moment calculation using image block representation, IEEE Transactions on Image Processing, 9(11):1977-1978, 2000.
14. J. D. Díaz de León and J. H. Sossa, Mathematical Morphology based on linear combined metric spaces on Z^2 (Part I): Fast distance transforms, Journal of Mathematical Imaging and Vision, 12:137-154, 2000.