*Fundamenta Informaticae 52 (2002) 1–11*
*IOS Press*


# A Simple Architecture for Computing Moments and Orientation of an Image*

**Sabyasachi Dey**†
*Texas Instruments Pvt. Ltd., Bangalore - 560 017, India,*
*dsabyasachi@ti.com*

**Bhargab B. Bhattacharya and Malay K. Kundu**‡
*Indian Statistical Institute, Calcutta - 700 108, INDIA*
*bhargab, malay@isical.ac.in*

**Tinku Acharya**§
*Intel Corporation, Chandler, AZ 85226, USA*
*acharya@ieee.org*

**Abstract.** In this paper, a new methodology for computing orientation of a gray-tone image by the method of moments is described. Orientation is an essential feature needed in many image processing and pattern recognition tasks. Computation of moments is accomplished by a new projection method that leads to a fast algorithm of determining orientation. Using a simple architecture, the proposed algorithm can be implemented as a special purpose VLSI chip. The hardware cost of the proposed design is significantly lower compared to that of the existing architecture.

**Keywords: Digital imaging, orientation, projection method, VLSI.**

## 1. Introduction

The task of recognizing orientation of an image has manifold applications to computer vision, pattern recognition, and image processing. Orientation information can be used to describe, recognize, and classify various objects in an image. For example, in remote sensing, this can be used to determine the

*This work was funded by a grant from Intel Corporation, USA (PO # CAC042717000)
†Address for correspondence: Texas Instruments Pvt. Ltd., Bangalore - 560 017, India
‡Address for correspondence: Indian Statistical Institute, Calcutta - 700 108, INDIA
§Address for correspondence: Intel Corporation, Chandler, AZ 85226, USA

direction of motion of an aircraft or a ship, to identify roads and rivers, to detect movement of a shoal of fish or a herd of livestock. This also helps in computing the directions of pressure gradients in a meteorological map for weather forecasting. Further, in industrial automation and robotics applications, orientation analysis plays an important role. For example, suppose a robot has to lift a moving object from a conveyer belt, with minimum stress. Computation of orientation and center of mass of the object is required to optimize the lift. Thus, efficient computation of orientation is needed for many online applications.

Several algorithms have been proposed in the literature for finding orientation. Among them, the methods based on determination of directed vein, convex hull, principal components transformation, and moments are important [9]. In this work, a new method for finding the orientation of an image is described based on computation of moments by the projection method. The direction of the principal axis of the given object defines its orientation. The principal axis is a line that passes through the center of mass of the object, around which the first order moment of the object is minimum. A naive algorithm for computing moments of an image requires several passes over the entire image and takes $O(N^2)$ time for an N×N image. Many applications demand fast real-time response that may not be achievable by a software system. For computing all moments, a few dedicated hardware architectures have been proposed in the recent past [1]. However, to determine orientation, computation of the moments only up to the second order is required. Hence, there is a need of a simple and application-specific hardware block capable of computing orientation quickly to meet the real-world demand. In this work, we propose a parallel algorithm for computing moments up to the second order by the projection method, and then design a very simple and regular VLSI architecture for its implementation. The hardware cost of the proposed design is significantly less compared to that of the existing architecture [1].

## 2. Preliminaries

The principal axis or axis of symmetry of an object can be found from the second order central moments. Let $\theta$ be the angle of the axis with x-axis ( see Fig. 1). It can be shown [9] that $\theta$ can be found by the following relation:

$$\theta = \frac{1}{2}\, tan^{-1}\, \frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \tag{1}$$

where $\mu_{11}, \mu_{20}$, and $\mu_{02}$ are central moments of third order given by the following equations:

$$m_{00} = \sum_x \sum_y f(x, y) \tag{2}$$

$$m_{10} = \sum_x \sum_y x f(x, y) \tag{3}$$

$$m_{01} = \sum_x \sum_y y f(x, y) \tag{4}$$

$$m_{11} = \sum_x \sum_y xy f(x, y) \tag{5}$$

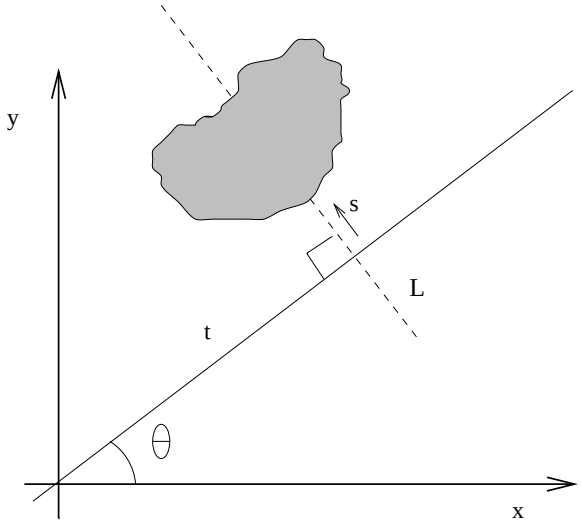


Figure 1. Projection method.

$$m_{20} = \sum_x \sum_y x^2 f(x,y) \tag{6}$$

$$m_{02} = \sum_x \sum_y y^2 f(x,y) \tag{7}$$

$$\mu_{11} = m_{11} - \frac{m_{10} m_{01}}{m_{00}} \tag{8}$$

$$\mu_{20} = m_{20} - \frac{m_{10}^2}{m_{00}} \tag{9}$$

$$\mu_{02} = m_{02} - \frac{m_{01}^2}{m_{00}} \tag{10}$$

where $f(x,y)$ is the intensity value at $(x,y)$ coordinates of the image.

In addition to this angle of orientation, we need to locate the center of gravity to define the principal axis properly.

The first order and second order central moments can be computed from the projections of the image on the x- and y-axes, and on the principal diagonal [3]. Consider a line through the origin inclined at an angle $\theta$ with the x-axis (see Fig. 1). Now construct a new line, intersecting the original line at right angle, through a point lying at a distance $t$ from the origin. Let the distance along the new line be designated as $s$. The integral of $f(x,y)$ along the new line gives the value of the projection. In other words,

$$p_\theta(t) = \int_L b(tcos\theta - ssin\theta, tsin\theta + scos\theta) ds$$

The integration is carried out over the portion of the line $L$ that lies within the image. The vertical projection $(\theta = 0)$, for example, is simply,

$$v(x) = \int_L f(x,y) dy,$$

while the horizontal projection ($\theta = \frac{\pi}{2}$) is,

$$h(y) = \int_L f(x,y)dx.$$

Now, since

$$A = \int \int_I f(x,y)dxdy,$$

we have

$$A = \int v(x)dx = \int h(y)dy.$$

More importantly,

$$\bar{x}A = \int \int_I xf(x,y)dxdy = \int xv(x)dx \tag{11}$$

and

$$\bar{y}A = \int \int_I yf(x,y)dxdy = \int yh(y). \tag{12}$$

Thus, the first moments of the projections are equal to the first moments of the original image.

To compute orientation we need the second moments. Two of these are easy to compute from the projections:

$$\int \int_I x^2 dxdy = \int x^2 v(x)dx$$

and

$$\int \int_I y^2 dxdy = \int y^2 h(y)dy.$$

We cannot, however, compute the integral of the product $xy$ from the two projections introduced so far. We can add the diagonal projection ($\theta = \pi/4$),

$$d(t) = \int f(\frac{1}{\sqrt{2}}(t-s), \frac{1}{\sqrt{2}}(t+s))ds.$$

Since,

$$\int \int_I \frac{1}{2}(x+y)^2 f(x,y)dxdy = \int \int_I t^2 f(s,t)dsdt = \int t^2 d(t)dt,$$

and

$$\int \int_I \frac{1}{2}(x+y)^2 f(x,y)dxdy = \int \int_I (\frac{1}{2}x^2 + xy + \frac{1}{2}y^2)f(x,y)dxdy.$$

We have,

$$\int \int_I xyf(x,y)dxdy = \int t^2 d(t)dt - \frac{1}{2}\int x^2 v(x)dx - \frac{1}{2}\int y^2 h(y)dy.$$

Thus, all the moments needed for computation of the angle of orientation in the image can be found from the horizontal, diagonal, and vertical projections.

So far we have restricted our discussions regarding moment calculation only to analog and continuous pictures. For a digital image, the equations for moments become:

$$v(x) = \sum_{L\| \ y-axis} f(x,y), \tag{13}$$

$$h(y) = \sum_{L\| \ x-axis} f(x,y), \tag{14}$$

$$d(t) = \sum f(\frac{1}{\sqrt{2}}(t-s), \frac{1}{\sqrt{2}}(t+s)), \tag{15}$$

$$m_{00} = \sum v(x), \tag{16}$$

$$m_{10} = \sum xv(x), \tag{17}$$

$$m_{01} = \sum yh(y), \tag{18}$$

$$m_{20} = \sum x^2 v(x), \tag{19}$$

$$m_{02} = \sum y^2 h(y), \tag{20}$$

$$m_{11} = \sum t^2 d(t) - \frac{1}{2}m_{20} - \frac{1}{2}m_{02}. \tag{21}$$

In the next section, we describe an algorithm for computing first and second order central moments by the method described above.

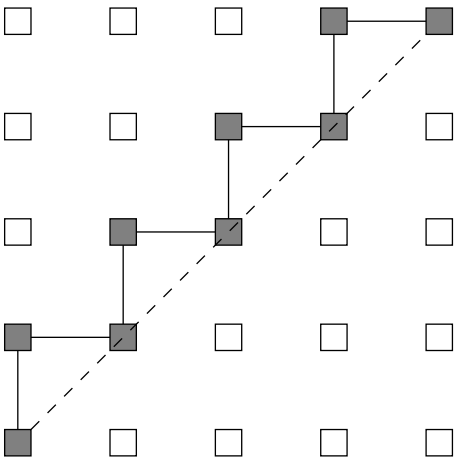

Figure 2. Staircase approximation of the diagonal

To compute the diagonal projection, we add the pixel values along each line L (Fig. 1) perpendicular to the diagonal. In order to accumulate the values on some point we must identify the point on the diagonal. In the case of a digital image, the diagonal and perpendicular lines do not always intersect on a pixel. To meet this requirement, we make slight approximation by considering a staircase diagonal instead of a line (Fig. 2).

## 3. Algorithm and Architecture

In this section, we describe an algorithm and an architecture for computing the orientation of a gray tone image. Consider an N×N image $\mathcal{I}$ as a 2-D matrix where the elements are from $\{0, 1\}$.

$$\mathcal{I} = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1N} \\ p_{21} & p_{22} & \cdots & p_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{NN} \end{pmatrix}$$

The architecture has N×N cells (processing elements or PE's), each of which performs very simple arithmetic operations and data movements. The cells are connected as a 2-D mesh architecture (Fig. 3). The algorithm accomplishes the task of finding second order moments in three phases. In phase 1, the cells are initialized, and at the end of phase 1, the cell $PE(i, j)$ has pixel data $p_{ij}$. Further, while the pixel data moves towards destined cell, the cells at the bottom boundary compute the vertical projection. In phase 2, horizontal and diagonal projections are computed. In phase 3, second order central moments are computed from the projection values. Each cell has four registers, $v, h, d,$ and $p$. For vertical data movement and for computing vertical projections, the register $v$ is used. Similarly, $h$ and $d$ are used for horizontal and diagonal data movements, and for computing horizontal and vertical projections. Each cell stores its own pixel data in register $p$. Vertical projections are computed in the cells on the bottom boundary. Horizontal projections are computed in the cells on the right boundary. Diagonal projections are computed in the cells on the staircase diagonal. The architectures of bottom boundary cells, left boundary cells, and cells on diagonals are shown in Fig. 4, Fig. 5, and in Fig. 6 respectively.

The algorithm for computing moments by the projection method is now described below.

**Algorithm 3.1.** *Compute Orientation*

**begin algorithm**

**Phase 1:** /* Initialization */

1. Pixel data is fed to the bottom cell boundary;
2. Pixels values of each column are accumulated in the cells on the bottom boundary;
3. Pixel data is moved upwards;
4. At the end, $PE(i, j)$ has $p_{ij}$, and
   the bottom boundary cells contain vertical projections.

**Phase 2:** /* Projection computation */

1. Move pixel data towards left one cell at a time;
2. Move pixel data along the diagonal;
3. At the end, horizontal projections are in the left boundary cells;
   and diagonal projections are in the cells on the staircase diagonal.

**Phase 3:** /* Moment computation */

Step 1 through 3 do in parallel

*Step 1 :* /* Compute $m_{20}, m_{10}$, and $m_{00}$ */

Vertical projection values are moved towards left to compute the moments as in Eq. 16, Eq. 17, and Eq. 18.

*Step 2 :* /* Compute $m_{02}, m_{01}$ */

Horizontal projection values are moved towards the bottom to compute the moments as in Eq. 19, and Eq. 20.

*Step 3 :* /* Compute $\mu_{11}$ */

Diagonal projection values are moved along the staircase diagonal towards $PE(N, 1)$ to compute the moment in Eq. 21.

**Phase 4:**
**begin**

Compute and output the following:

$\mu_{11} = m_{11} - \frac{m_{10} m_{01}}{m_{00}}$;
$\mu_{20} = m_{20} - \frac{m_{10}^2}{m_{00}}$;
$\mu_{02} = m_{02} - \frac{m_{01}^2}{m_{00}}$;
Angle of orientation $\theta = \frac{1}{2} tan^{-1} \frac{2\mu_{11}}{\mu_{20} - \mu_{02}}$;
Center of gravity $(\bar{x}, \bar{y})$ :
$\bar{x} = \frac{m_{10}}{m_{00}}$;
$\bar{y} = \frac{m_{01}}{m_{00}}$;

**end**

**end algorithm**

The correctness of the algorithm is directly proved from the discussions in Section 2. The final output is available from $PE(N, 1)$, i.e., the cell on the bottom-left corner. Hence, all data I/O takes place only in the cells on the bottom boundary. Data movement takes place from top to bottom in phase 1, from right to left and diagonally in phase 2. There is no data movement towards right.

## 3.1. Algorithm Partition

In real-world applications, an image often has a very large size. It is not practical to build a VLSI architecture with a large number of processing elements to handle images with different sizes. At the

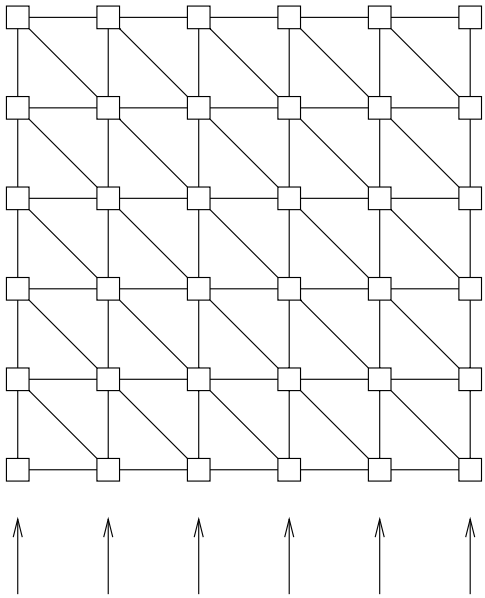


Figure 3. Architecture and cell organization for computing moments by projection method.

same time, it is not practical to design a new chip for each individual task of varying sizes. Therefore, a fixed-size VLSI architecture is highly needed.

If the image size is larger than the circuit size, then we have to partition it into smaller sub-images to fit them on the given architecture. Given an image of size $k \times l$, and a 2-D VLSI architecture of size $m \times n$, if $k, l$ are divisible by $m$ and $n$ respectively, we can easily partition the image and calculate the moment accordingly. Each component can be fed to the circuit one after another. The corresponding offset of the origin should be passed to all the cells. If $k$ ($l$) are not multiple of $m$ ($n$), then we can fill the extra columns and rows of the image with zeros, to make it a multiple of $m$ ($n$).

## 3.2. Complexity Analysis

Algorithm 3 performs the task in three different phases, which are executed in sequential manner one after another. In phase 1, the loop iterates N times. Hence, it takes $O(N)$ time for execution. In phase 2, data movement takes place diagonally and from right to left. The main loop is executed N times, hence phase 2 takes $O(N)$ time. In phase 3, three steps are executed in parallel. Steps 1 and 2 take N time units to compute $\mu_{20}$ and $\mu_{02}$ respectively, while step 3 takes $2N - 1$ time units to complete the task. Hence, the overall time complexity is $O(N)$. Though we are using a 2-D mesh structure, only the cells on the bottom and left boundary, and on the staircase diagonal perform certain computations. Hence, only $O(N)$ cells should need multiplication and addition functions. Other cells are used just for data movement. For an N×N image having $h$ different intensity levels, the width of the arithmetic unit is given in Table 1. All other data path units in the circuit need very small chip area, and therefore, the overall area primarily depends on the number and size of the multiplier and adder cells.

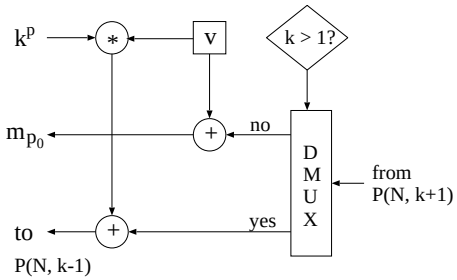


Figure 4. A cell on the bottom boundary.

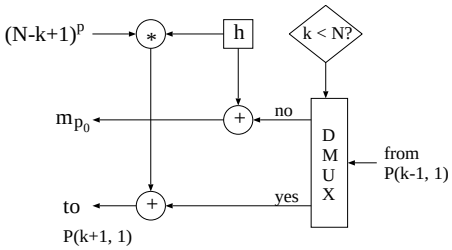


Figure 5. A cell on the left boundary.

**Table 1.** Size of the arithmetic unit

| Unit type | Size (in bits) |
|---|---|
| Adder | $4(\log_2 N + \log_2 h)$ |
| Multiplier | $4 \log_2 N$ |

## 4. Circuit cost

The algorithm has been implemented in C on a UNIX platform. Experimental results are very encouraging. The orientation of the object shown in Fig. 7 was found to be 150º. In Fig. 8 the object is a circle, hence it is symmetric with respect to all axes. Since the angle of orientation is being computed using moments, we define it as $90^o$ when the denominator in Eqn. 1 becomes zero for a totally symmetric object. Thus, the orientation of the circle (Fig. 8) is found to be 90º, which is indeed an axis of symmetry.

In the recent past, a VLSI architecture has been proposed [1] for computing regular and central moments. However, for computing orientation alone, the circuit is overly complex. Our proposed architecture is very simple, cost-effective and computes moments up to the second order, and hence can be used in application-specific systems where computation of orientation is required. It computes the moments up to the second order in $4N + k$ clock cycles, where $k$ is a constant factor. Table 2 shows the comparative hardware cost of the proposed design with that by Cheng et al. [1]. The adder and multiplier blocks are same in both the cases. Thus, for dedicated applications where orientation computation is

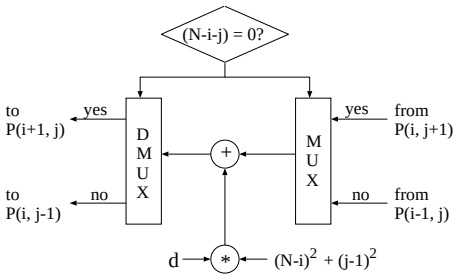


Figure 6. A cell on the staircase diagonal.

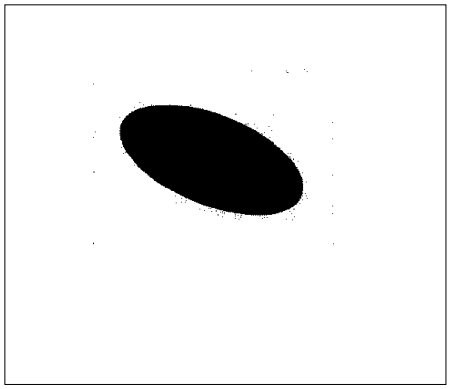

Figure 7. An ellipse (orientation $151^o$).

required, the proposed architecture is more suitable and cost-effective.

**Table 2.** Comparative hardware cost

| Component type | Architecture of [1] | Proposed architecture |
|---|---|---|
| multiplier | $3.N^2$ | $18.N$ |
| adder | $3.N^2$ | $12.N$ |
| 2-1 multiplexer | $2.N^2$ | nil |
| 2-input AND gate | $11.N^2$ | $8.N^2$ |
| NOT gate | $3.N^2$ | nil |

## 5. Conclusion

In this paper, we have described a new technique for computing momemnts and the angle of orientation of an object in a gray level image. The method is simple, cost-effective and can easily be implemented with a regular VLSI architecture.

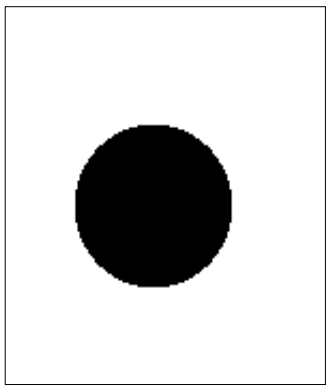

Figure 8. A circle (orientation $90^o$).

# References


[1] Cheng H.D., Wu C.Y., and Hung D.L.,"VLSI for Moment Computation and its Application to Breast Cancer Detection", *Pattern Recognition*, Vol. 31, No. 9, pp. 1391-1406, 1998.

[2] Gonzalez R.C. and Woods R.E., Digital Image Processing. *Addison-Wesley*, Reading, Massachusetts, pp. 504-506, 1993.

[3] Horn B.K.P., Robot Vision. *The MIT Press*, Masachhusetts, 1991.

[4] Hu M.K.,"Visual Pattern Recognition by Moment Invariant", *IRE Trans. Info. Theory*, Vol. IT-8, pp. 179-187, 1962.

[5] Kung H.T,"Why Systolic Architectures", *IEEE Transactions on Computers*, pp. 12-17, July 1982.

[6] Kung H.T. and Leiserson C.E.,"Systolic Arrays", *Sparse Matrix Proc.*, Society for Industrial and Applied Mathematics, pp. 256-282, 1979.

[7] Leiserson C.E., Area Efficient VLSI Computation. *MIT Press*, 1983.

[8] Luo D., Macleod J.E.S., Leng X., and Smart P., "Orientation analyses of particles in soil microstructures", *Geotechnique*, Vol.24, pp.97-107 , 1992.

[9] Luo D., Pattern Recognition and Image Processing. *Horwood Publishing*,1998.

[10] Mead C.A. and Rem M.,"Cost and Performance of VLSI Computing Structures", *IEEE Journal of Solid-State Circuits*, SC-14, No. 2, pp. 455-462, 1979.

[11] Pratt W.K., Digital Image Processing. *John Wiley & Sons.*, 1978.

[12] Rosenfeld A., Kushner T., Wu A.Y.,"Image Processing on ZMOB", *IEEE Transactions on Computers*, Vol. C-31, pp. 943-951, 1982.

[13] Rosenfeld A. and Kak A.C., Digital Picture Processing. *Academic Press*, Vols. 1 and 2, 1982.

[14] Tarjan R.E.,"Complexity of Combinatorial Algorithms", *SIAM Review*, 20, pp. 451-491, 1978.

[15] Ullman J.D., Computational Aspects of VLSI. *Computer Science Press*, 1984.