A PC-Based Real-Time Computation of Moment Invariants*

Alrawi Mohammed , YANG Jie, ZHANG Feng-chao

(*Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University, Shanghai* 200030, *China*)
E-mail: jieyang@online.sh.cn
Received March 22, 2002; accepted May 27, 2002

Abstract: Moments and invariant moments are important features used in identification and inspection of industrial parts. It is necessary to compute geometric moment's values in real-time rate. The efficient computation of two-dimensional geometric moments on gray-level images is addressed in this paper. Despite the existence of many algorithms of fast computation of moments, it cannot be implemented for real-time computation to be run on a PC without the use of some special dedicated hardware tools. The reason beyond this is that those fast algorithms do reduce the complexity of computing but still one needs to use floating-point arithmetic operations in the computation process. To achieve real-time computation on a PC machine, what the algorithm suggested here is based on dividing the image into equally sized blocks. This algorithm works by computing local moments at each block using integer operations, then accumulating the total image moments with floating-point operations. With this computation scheme no approximation is used, it is an exact computation. Overcoming this overflow problem, however, is not straightforward without using some kind of transformation to each block. Hatamian's (improved) filter is used to compute those block moments (BLMs) efficiently. The experiments show that the algorithm presented in the paper has greatly reduced floating-point operations in fast computation of moments, and greatly improved the speed of the computation of moments. The new algorithm can be effectively used in real-time identification and inspection of complicated industrial parts.
Key words: industrial vision; fast computation of moment invariant; Hatamian's filter; parallel algorithm

Moment invariants are important features used in pattern recognition problems. They can be used to achieve translations, rotations and scales recognition of unoccluded objects. The identification and inspection of industrial parts that are randomly located on a moving conveyor belt is just a simple example of the applications of geometric moments. Practically, geometric moments were first introduced by Hu[1] 1961 and were used in OCR problem. Different kinds of moment's[2] have been used in many pattern recognition and image processing problems. Goals of using image moments may vary from OCR texture classification, aircraft identification, image analysis, edge detection, encoding etc. The methods of moments are one of the most reliable methods in invariance recognition problems.

Moments suffer from the heavy computation due to the monomials used in their computation. In addition, computing moments for real-time implementations can not be achieved without the use of some specially dedicated hardware. Several

* Supported by the National Natural Science Foundation of China under Grant No.30170274 (国家自然科学基金)

Alrawi Mohammed was born in 1966. He is a Ph.D. candidate at the Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University. His research interests are image understanding and computer vision. **YANG Jie** was born in 1964. He is a professor and doctoral supervisor at the Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University. His current research areas are object detection, recognition, data fusion and data mining, and medical image processing. **ZHANG Feng-chao** was born in 1972. He is a Ph.D. candidate at the Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University. His research interests are object detection, recognition, and computer vision.

algorithms were suggested for the fast computation of 2-D moments. Hatamian[3] suggested the most efficient fast algorithm to compute 2-D moments up to the 3rd order via digital filters. Even with the increased usage of high order moments in pattern recognition problems, fewer efforts were devoted to develop high order 2-D moment generators. There were some Hatamian-dialect fast moment algorithm[4~6]. Computations are slightly less than the original Hatamian algorithm. This was achieved via using an efficient digital filter output to geometric moment transformation scheme. In Ref.[6] the filter was changed slightly and computations were reduced slightly too. Computations can be performed, however, only for 2-D moments up to the 3rd order. For other kind of fast algorithms see Refs.[7~13]. For some recent applications and theories of moments see Refs.[14~16].

In this paper, a new efficient moment computation algorithm is presented. The algorithm is designed to improve the performance and increase the efficiency of any other fast algorithm. Due to the high range of moments which causes overflow and hence the requirement to use floating-point (FLP) arithmetic operations, the intrinsic numerical values of moments will be the key solution of the algorithm. Dividing the image into blocks is the scheme of the algorithm that enables the computation of moments at each block (BLMs) using fixed-point (FXP) arithmetic operations. Then a few FLP operations are required to accumulate the total image moments. It will be shown that with the use of this algorithm a teal-time computation of 2-D image moments can be attained.

1 A Note on the Design of a Hardware Accelerator for Real-Time Moment Computation: A Wave Front Array Approach

Isolated 2-D image recognition remains among the important issues in the industrial and other applications of vision. The most simple and efficient features are based on geometrical moments[1]. A new design of a hardware accelerator for real-time moment computation was suggested recently by Hung *et al*[17]. Many other sequential[2~5] and/or parallel algorithms[17,18] had been suggested in relation to this field of geometric moments computation. To this, we would like to give the following notes:

The work did not make use of (nor compare with) the flexible computation redundancy in which moments can be computed efficiently. Some fast algorithms, i.e. Hatamian's filter[2] method and its extensions[3,4] and recently[5] can be used to compute geometric moments free of multiplications. Multiplications were very time consuming and took large area size in Hung's *et al* design. Hatamian's algorithms enable reduction in memory, hardware size, and computation time. High-order moment generators using filter method are also available. Their performance to be run on a Pentium PC (using C language were a 64-bit double precision floating point format have been used) is nearly equivalent (in computation time) to the design of Hung's et al as shown in Fig.1.

Using the filter method, Hatamian designed a real-time moment algorithm with its single chip implementation[3]. Hatamian's chip is capable of computing 512×512 8-bits/pixel geometric image moments from M_{00} up to M_{33} at the speed of 30-frame/second that is more proper for real-time vision. The single chip is very flexible and can be used with a host computer. Using the same chip and the block moment algorithm (run on a host computer), it is possible to compute 1024×1024 8-bits/pixel geometric image moments from M_{00} up to M_{33} at the speed of 7-frames/second which is adequate for real-time vision problems too. In general, the number of additions required to generate 2D geometric moments up to the order (p,q) for $(p=q)$ is $\{(p+1)(N+1)^2+(N+1)(p+1)^2\}$ with a negligible amount of multiplications.

It is concluded that the moment processing elements (MPE) proposed in Hung's et al is not the best available choice nor the most efficient due to the high multiplications used. One of their aims, however, was to apply the most recent hardware design technology to direct computation of moments, the question is the optimum available algorithm. On the contrary, it is very consuming and the cost is still high. For a 512×512 8-bits/pixel image, it is possible to compute moments on Pentium-II(R) Intel MMX(TM) 350 MHz PC in real-time rate using Hatamian's filter method and its improved extensions. Moreover, Hatamian's filter can be used with an assistant block division algorithm as will be proposed here that enable the use of integer additions instead of the heavy floating point additions for the majority of operations. Hatamian's algorithm exists as a single chip, and the required block moment algorithm can be implemented on a host computer. Thus, not only a reduction of arithmetic operations is the goal but to perform some operations with integer arithmetic. This indicates an important fact that the development and selection of a good moment generation algorithm added to the rapid increase of computers speed will be as fast as and sometimes faster than hardware moment generators.

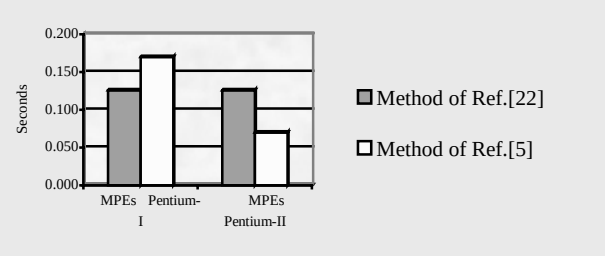

Fig.1 Performance comparison between Hung's *et al* moment processing elements MPEs (16 MPEs units are used and related estimated time are taken from Ref.[2]) and the filter method[3] to be run on a Pentium PCs. Moments are to be computed from M_{00} to M_{44} for a 512×512 8-bits/pixel image. Pentium-I (133 MHz) and Pentium-II (350 MHz) were used respectively

2 Definitions of 2-D Image Moments

Geometric moments were first introduced by Hu[1] to be applied in pattern recognition problems. When computed for a 2-D gray-level image it can be used to generate the so-called moment invariants. These invariants were used in many image processing and pattern recognition problems[9]. Other kinds of moments also exist, the most important is Zernike moments introduced by Teague[10], see also Ref.[11] for different kinds of moments (e.g. Legendre moments, pseudo Zernike moments, radial moments, complex moments, etc.) and their relations.

The two-dimensional moment for a function f are defined in terms of Rieman integral as

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p\, x^q\, f(x,y)\,\mathrm{d}x\,\mathrm{d}y\,, \tag{1}$$

where $p+q=0,1,2,\ldots$is the moment order. Also it is assumed that $f(x,y)$ is a piecewise continuous and has none-zero values only in the finite part of R^2, then moments of all orders exist. For a digital image the double integration is changed into a double summation, geometric moments will be given as

$$\mu_{pq} = \sum_{i=1}^{N}\sum_{j=1}^{M} i^p\; j^q\; f_{ij}\,, \tag{2}$$

where the image f is digitized into $N\times M$ samples. To achieve translation invariance central moments are used to move the origin of the image to the center of gravity as follows

$$\mu_{pq}^{Tr} = \sum_{i=1}^{N}\sum_{j=1}^{M} (i-\bar{i}_{10})^p\; (j-\bar{j}_{01})^q\; f_{ij}\;,$$

where the point $(\bar{i}_{10}, \bar{j}_{01})$ is the center of the gravity of the image, $\bar{i}_{10} = \mu_{10}/\mu_{00}$ and $\bar{i}_{01} = \mu_{01/}\mu_{00}$. While scale

changes can be normalized by: $\mu_{pq}^{\mathrm{Tr,Sc}} = \dfrac{\mu_{pq}^{\mathrm{Tr}}}{(\mu_{00})^{(p+q+2)/2}}$. Then moment invariants can be obtained from rotational invariance.

3 Numerical Ranges of Geometric Moments

To determine the threshold size of the image, some numerical estimation of the values of moments is developed here. Surely, the maximum value that a certain moment can have depends on the image size of the image and the maximum gray-level value, given as follows:

$$m_{pq}(\max) = f_{\max} \sum_{i=N_0}^{N_1} \sum_{j=M_0}^{M_1} i^p \; j^q \;, \tag{3}$$

where $f_{\max}$ is the maximum gray-level value, $(N_1-N_0)\times(M_1-M_0)$ is the block dimensions. If we assume the top-left block as the low order block, and the right-bottom block as the high order block, intermediate blocks increased from low to high order blocks. Lets assume moments to be calculated up to the third order (yielding ten moments). Then the values of $m_{30}(\max) = m_{03}(\max)$ would take the maximum value among all of the rest eight moments. Without loss of generality, let us assume that $(N_1-N_0)>(M_1-M_0)$ then the maximum value of the third order moment is given by

$$m_{30}(\max) = [\rho(N_1)^2 - \rho(N_0-1)^2][M_1 - M_0], \tag{4}$$

where $\rho(s) = s(s+1)/2$, S is a dummy variable. To keep the moment values within the range of the maximum value that the FXP word length adder can have (λ value), the following condition must hold ($m_{pq}(\max) < \lambda$). Table 1 demonstrates values of $m_{30}(\max)$ via different image sizes, assuming a square image of size N and $f_{\max}$=255. Its obvious that using a 32-bit word length its maximum value $\lambda = 4\,294\,967\,296$ is the best case. The threshold image size that matches this adder is 64.

Table 1 Maximum values that a third order moment m_{30} can have via image size changes for an 8-bits/pixel image

Image size	$m_{30}(\max)$
4	4 590
8	102 000
16	2 643 840
32	75 463 680
64	2 274 877 440
128	70 606 848 000
256	2 224 792 535 040
512	70 642 544 148 480

4 Block Moments of a 2-D Image

Considering the case where an image of size ($N\times N$) is divided into a number of equally sized blocks, then the total image moment of order ($p+q$) can be represented in terms of the moment value of each block. Decomposing the image into blocks that have the property of the top-left corner block can be of more great flexibility. Equally sized blocks are the optimum case since each block will have the maximum numerical range basing on the top-left corner block in the image. Let us define the block moment of order ($p+q$) and location (u,v) as

$$m_{pq}(u,v)=\sum_{i=u+1}^{u+N_B}\sum_{j=u+1}^{v+N_B} i^p\ j^q\ f_{ij}\ , \tag{5}$$

where $N_B \times M_B$ is the size of each block in the image, $\{u \in 0,1,2,\ldots,N-N_B\}$ and $\{v \in 0,1,2,\ldots,M-M_B\}$. In fact, (5) can be rewritten as

$$m_{pq}(u,v)=\sum_{i=1}^{N_B}\sum_{j=1}^{M_B} (i+u)^p\ (j+v)^q\ f_{i+u,j+v} \tag{6}$$

and using the binomial expansion we get

$$m_{pq}(u,v)=\sum_{r=0}^{p}\sum_{s=0}^{q}\binom{p}{r}\binom{q}{s}\ u^{p-r}\ v^{q-s}\ \mu_{rs}^{uv}\ , \tag{7}$$

where μ_{rs}^{uv} is given by

$$\mu_{rs}^{uv}=\sum_{i=1}^{N_B}\sum_{j=1}^{M_B} i^r\ j^s\ f_{i+u,\ j+v} \tag{8}$$

and $f_{i+u,j+v}$ is the pixel value at the location $(i+u,j+v)$. Having block moments as those defined in (7) we can get total image moments by accumulating all block moments of a certain order, this may be accomplished as

$$m_{pq}=\sum_{I=0}^{N_b-1}\sum_{J=0}^{M_b-1} m_{pq}(u,v)\ , \tag{9}$$

where $u=(IN_B)$, $v=(JM_B)$, $N_b=N/N_B$, and $M_b=M/M_B$. Substituting (7) into (9) and changing the order of summations yields:

$$m_{pq}=\sum_{r=0}^{p}\sum_{s=0}^{q}\sum_{I=0}^{N_b-1}\sum_{J=0}^{M_b-1}\binom{p}{r}\binom{q}{s}(IN_B)^{p-r}\ (JM_B)^{q-s}\ \mu_{rs}^{uv}\ . \tag{10}$$

A more useful representation is as follows

$$m_{pq}=\sum_{r=0}^{p}\sum_{s=0}^{q}\binom{p}{r}\binom{q}{s}(N_B)^K\ (M_B)^L\ (\mu_{rs})_{KL}\ , \tag{11}$$

where $K=p-r$, $L=q-s$. The value of the accumulated moment is given by

$$(\mu_{rs})_{KL}=\sum_{I=0}^{N_b-1}\sum_{J=0}^{M_b-1} I^K\ J^L\ \mu_{rs}^{uv}\ , \tag{12}$$

Equation (11) is the final formula of computing 2-D image moments. As can be seen, moments computed in (8) are used as the input data to compute other kind of moments given in (12), which is used finally to compute the required 2-D image moments through (11). Therefore, computational complexity is reduced. Using (11), up to the third order total accumulated moments are given by

$$\begin{aligned}
m_{00}&=(\mu_{00})_{00}\\
m_{01}&=M_B(\mu_{00})_{01}+(\mu_{01})_{00}\\
m_{10}&=N_B(\mu_{00})_{10}+(\mu_{10})_{00}\\
m_{11}&=N_BM_B(\mu_{00})_{11}+N_B(\mu_{01})_{10}+M_B(\mu_{10})_{01}+(\mu_{11})_{00}\\
m_{02}&=M_B^2(\mu_{00})_{02}+2M_B(\mu_{01})_{01}+(\mu_{02})_{00}\\
m_{20}&=N_B^2(\mu_{00})_{20}+2N_B(\mu_{10})_{10}+(\mu_{20})_{00}\\
m_{12}&=N_B\ M_B^2(\mu_{00})_{12}+M_B^2(\mu_{10})_{02}+2N_B\ M_B(\mu_{01})_{11}+2M_B(\mu_{11})_{01}+N_B(\mu_{02})_{10}+(\mu_{12})_{00}\\
m_{21}&=N_B^2\ M_B(\mu_{00})_{21}+N_B^2(\mu_{01})_{20}+2N_B\ M_B(\mu_{10})_{11}+2N_B(\mu_{11})_{10}+M_B(\mu_{20})_{01}+(\mu_{21})_{00}\\
m_{03}&=M_B^3(\mu_{00})_{03}+3M_B^2(\mu_{01})_{02}+3M_B(\mu_{02})_{01}+(\mu_{03})_{00}\\
m_{30}&=N_B^3(\mu_{00})_{30}+3N_B^2(\mu_{10})_{20}+3N_B(\mu_{20})_{10}+(\mu_{30})_{00}
\end{aligned} \tag{13}$$

It is obvious that most of the high dynamic moment values were taken out of the original moment computation scheme, these high values are computed in the above equation (as well as the total moment accumulation process).

In addition, computations using the improved digital filter via FXP arithmetic on (8) then, FLP arithmetic will be used with (12) for the moment accumulation process.

5 Experimental Results

A real-time 2-D moment generating algorithm and its single chip implementation was realized by Hatamian[3]. That chip is capable of computing image moments of sizes less than or equal to 512×512 8-bits per pixel. Due to the fixed length integer adders used, a real-time PC based computation of moments is not straightforward, however, without the use of our BLMs algorithm.

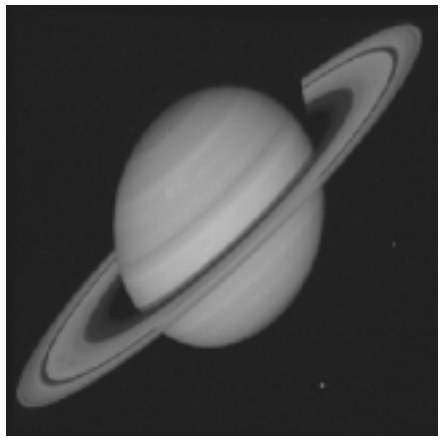

Fig.2 The 512×512 sized image used in the BLMs computation and comparison

The proposed algorithm was used with the (latest and the most efficient) improved digital filter[6] to compute moments up to the third order, then computing the feature set of invariant moments[1] of the image shown in Fig.2.

The software was written in C language. A 512×512 8 bits/pixel image was divided into a number of equally sized blocks. Each block was of size (N_B=64). FXP arithmetic was used to compute block Moments at each block. For comparison purposes, the improved digital filter was used directly on the image (i.e. without block decomposition). The machine used in computation was Pentium II (350 MHz). As shown in Table 3, the proposed algorithm is nearly three times faster than the improved filter of Hatamian[6], and twenty two times faster than the direct computation. From Table 3 one can see that using the proposed computation scheme, the real-time computation a 512×512 8-bits/pixel image moments is possible. A more detailed complexity of computation analysis is shown in Table 2. Moreover, moment invariants of this image (Fig.2) are shown in Table. 4.

Table 2 Comparison of theoretical computation complexity of regular moments

Algorithm	No. of additions	No. of Multiplications
Straightforward	$(10N^2)_{\text{float}}$	$(20N^2)_{\text{float}}$
Hatamian's improved filter[8]	$(4N^2+10N+12)_{\text{float}}$	0
Suggested block algorithm+ Hatamian's improved filter	$N_b^2(4N_B^2+10N_B+12)_{\text{int}}+(20N_b^2+37N_b+45)_{\text{float}}$	$(25)_{\text{float}}$

Note that $N_b=N/N_B$, "float" refers to floating-point operations, "int" refers to integer operations.

Table 3 Comparison of execution time and speedup factor for a 512×512 8-bits/pixel image

Algorithm	Execution time in (s)	Speed up factor
Straightforward	0.634	1
Hatamian's improved filter[8]	0.085	7.46
Suggested block algorithm + Improved filter	0.028	22.59

6 Conclusions

The methods of moments have taken an increased concern due to their concise applications in image processing and pattern recognition. This work suggests a new block-moments-algorithm for the real-time computation of image moments based on dividing the image into equally sized blocks. The method works by calculating moments of each block followed then by the accumulation of total image moments. Despite the fact that

Table 4 Log values of moment invariants for the image used in the test computed using the suggested algorithm

Reference image (512×512)	Rotated 45°	Rotated 180°
6.165 8	6.164 9	6.165 2
17.422	17.377	17.432
22.620	22.622	22.619
42.496	42.497	42.499
47.255	47.239	47.235
32.459	32.456	32.458
47.661	48.354	47.255

the suggested algorithm has the advantage of computing moments of large size images in the sequential mode, the flexibility in parallelism is of more great importance. The computational advantage stems from the reduction in the word length to enable the use of a FXP arithmetic adder, hence reducing the number of FLP arithmetic needed. This may result in a high accuracy as well as high-speed computation.

Numerical analysis has been presented to be able of determining the threshold image size via moment orders and to avoiding overflow. Obviously, if the block size is increased the range of moment values will be increased too. Also, the highest the moment order yields the highest numerical value. Moments at the lowest order block (at top-left corner which starts from the point "1,1") take the minimum values among all other blocks. Moment values at the highest order block (right-bottom block near the end-point "*N*,*N*") takes the highest values that may be in the numerical order of the whole image. The latest is the major problem that was solved in this paper. It was shown that all blocks can be computed within the properties of the lowest order block (which means making use of the maximum block size). We also concluded that, for a typical model where 8-bits/pixel image moments are to be computed up to the third order using a 32-bit word length adder the threshold block size is 64. A suggestion is that the block moment algorithm can be used in high speed/precision texture classification, where the textural images are divided into blocks and moments and their invariants are computed at each block. Then the feature vector may include those invariants in addition to invariants obtained from the accumulated moments (the total moments of the texture). Although, the method can be used for industrial inspection of damaged/undamaged object identification where moments for each object are computed as a function to some set of concentric circles centered at the gravity center of the image.

References:

[1] Hu, M.K. Visual pattern recognition by moment invariants. IRE Transactions on Information Theory, 1962,8(2):179~187.

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

[3] Hatamian, M. A real-time two-dimensional moment generating algorithm and its single chip implementation. IEEE Transactions on Acoustics, Speech, and Signal Processing, 1986,34(6):546~533.

[4] Li, B.C. High order moment computation of gray-level images. IEEE Transactions on Image Processing, 1995,4(4):502~505.

[5] Abdul-Hameed, M.S. High order multi-dimensional moment generating algorithm and the efficient computation of Zernike moments. In: Proceedings of the 1997 International Conference on Acoustics, Speech, and Signal Processing, ICASSP'97. Germany: Munich, 1997. 3061~3064.

[6] Wong, W.H., Siu, W.C. Improved digital filter structure for the fast moments computation. Proceedings of the IEE on Vision, Image and Signal Processing, 1999,146(2):73~79.

[7] Dai, M., *et al*. An efficient algorithm for the computation of shape moments from run-length codes or chain codes. Pattern Recognition, 1992,25(10):119~1128.

[8] Yang, L., Albregsten, F. Fast and exact computation of Cartesian geometric moments using discrete Green's theorem. Pattern Recognition, 1996,29(7):1061~1073.

[9] Yang, L., Albregsten, F., Taxt, T. Fast computation of three-dimensional moments using a discrete divergence theorem and a generalization to higher dimensions. Graphical Models and image Processing, 1997,59(2):97~108.

[10] Li, B.-C., Shen, J. Pascal triangle transforms approach to the calculation of 3-D moments. Graphical Models and Image Processing, 1992,54(4):301~307.

[11] Cyganski, D., Kreda, S. J., Orr, J.A. Solving for the general linear transformation relating 3-D objects from the minimum moments. SPIE Intelligence Robots and Computer Vision V. II, Proceedings of the SPIE. Vol 1002. Bellingham, WA: SPIE, 1988. 204~211.

[12] Li, B.-C., Shen, J. Fast computation of moment invariants. Pattern Recognition, 1991,24(8):807~813.

[13] Spiliotis, I.M., Mertsios, B.G. Real-Time computation of two-dimensional moments on binary images using image block representation. IEEE Transactions on Image Processing, 1998,7(11).

[14] Wang, L., Healey, G. Using Zernike moments for the illumination and geometry invariant classification of multispectral texture. IEEE Transactions on Image Processing, 1998,7(2):196~203.

[15] Mamistvalov, G. *N*-Dimensional moment invariants and conceptual mathematical theory of recognition *n*-dimensional solids. IEEE Transactions on PAMI, 1998,20(8):819~831.

[16] Liao, S.X., Pawlak, M. On image analysis by moments. IEEE Transactions on Pattern Recognition and Machine Intelligence, 1996,18(3):254~266.

[17] Hung, D.L., Cheng, H.D., Kamyong, S.S. Design of a hardware accelerator for real-time moment computation: a wave front array approach. IEEE Transactions on Industrial Electronics, 1999,46(1):18~25.

[18] Liu, Wen-tai, Chen, Su-Shing, Ralph Cavin. A bit-serial VLSI architecture for generating moments in real-time. IEEE Transactions on Systems, Man, and Cybernetics, 1993,23(2):539~546.

基于 PC 的不变矩实时计算算法

Alrawi Mohammed, 杨 杰, 张凤超

(上海交通大学 图像处理与模式识别研究所,上海 200030)

摘要: 矩和不变矩是工业部件识别和检测的重要特征.几何矩的值必须实时计算.介绍了灰度图像二维几何矩的高效计算.尽管存在许多矩快速计算算法,但不能在没有特殊硬件工具的微机上实时计算.原因是这些快速算法虽减少了计算复杂性,但在计算过程中仍需要大量浮点运算.为了实现在微机上的实时计算,提出的算法将图像分成相同大小的块,每图像块运用定点运算计算各自矩,然后运用浮点运算计算整个图像的矩.这种计算模式不需要近似而是精确计算,然而对于每个图像块不采用变换不容易克服溢出问题,在高效计算各图像块矩过程中使用了改进的 Hatamian 滤波器.实验结果表明,提出的算法大大减少了浮点运算次数,大大提高了图像矩计算速度.该算法可有效应用于复杂工业部件的实时识别和检测.

关键词: 工业视觉;不变矩快速算法; Hatamian 滤波器;并行算法

中图法分类号: TP391 文献标识码: A