Cross correlation function computation algorithm for video surveillance system.

Sadykhov R. Kh. [1,2], Lamovsky D. V. [2]

1) United Institute of Informatics Problems National Academy of Science of Belarus, The Laboratory of System Identification, 6, Surganov st., Minsk, 220012, Belarus, URL: http://lsi.bas-net.by

2) Belarusian State University of Informatics and Radioelecrtonics, Computer Department, 6, P.Brovka st., Minsk, 220013, Belarus, E-mail: rsadykhov@bsuir.unibel.by, lamovsky@tut.by

Abstract: *This paper describes a new algorithm to calculate cross-correlation function. We combined box filtering technique for calculation of cross correlation coefficients with parallel processing using MMX/SSE technology of modern general purpose processors. We have used this algorithm for real time optical flow estimation between frames of video sequence. Our algorithm was tested on real world video sequences obtained from the cameras of video surveillance system.*

***Keywords*:** cross-correlation, fast algorithm, MMX/SSE extensions, optical flow, video surveillance system.

I. INTRODUCTION

Feature matching is an important task in the area of computer vision. There are several different approaches for image correspondence estimation [1]. There is a group of so called area based methods [2] among them, and intensity values only are used to determine correspondence of the areas in this case. Three classes of metrics are commonly applied for area matching: cross correlation (CC), intensity differences (sum of absolute differences (SAD), sum of squared differences (SSD)), and rank [3] metrics. Cross correlation metrics is the standard statistical method for determining of similarity. This metrics is more robust than the others but computationally expensive.

Feature matching is used in many problems of computer vision. One of them is optical flow estimation. Optical flow is a two-dimensional vector field that represents velocities and their directions in each point of the image. It is used in many applications such as robot navigation, object tracking, video coding, and scene reconstruction. There are several different approaches to determine optical flow [4, 5]. Correlation based methods give more accurate results in the case of noised images with non rigid objects. The real world scenes obtained from the cameras of video surveillance system correspond to these conditions.

The main problem of most correlation based techniques is that they are computationally expensive. This fact doesn't allow using them in real time video processing. There are several ways to overcome this problem such as using optimal algorithms, application of special purposes hardware or using SIMD extension of modern general purposes processors.

Computationally optimal algorithms implement different approaches which allow to reduce computational cost for correlation. Fast Fourier transformation can be used for fast image matching algorithms [1, 5], and such technique is exploited in [6]. Approach known as box filtering technique [7] is also used for fast correlation calculation algorithm. Description of the algorithms that compute SAD (sum of absolute differences) metrics using box filtering technique can be found in [8, 9]. This technique in [10, 11] is used for fast NCC (normalize cross-correlation) metrics.

Using of special purposes hardware such as field programmable gate array (FPGA) chips is an excellent way to raise the performance of data processing. Several solutions can be discovered in [12, 13] for image and video processing. Using such hardware can lead to acceptable results, but it also requires additional efforts for special computational architecture realization.

Modern general purposes processors have significant computational power, moreover they have special sets of commands for accelerating multimedia and communication applications (we mean MMX and SSE extensions).

MMX and SSE appeared accordingly in 1996 and 1999 and since then all the popular general purposes processors support these technologies. These extensions exploits single instruction multiple data (SIMD) principle for parallel data processing. MMX and SSE are sets of new processor commands and data types. MMX allows parallel processing of byte, word, double word integer values. The main disadvantage is that MMX registers are in fact FPU (floating point unit) registers with other names. It means that it isn't possible to mix MMX instructions and FPU instructions in code. SSE is elaboration of MMX and operates with packed floating-point data. Also SSE includes 12 new instructions that extend the MMX instructions set and operate MMX registers. SSE2 is elaboration of both MMX and SSE. The key benefits of SSE2 are that MMX is extended to work with 128-bit data blocks, and the ability to support 64-bit floating-point values appeared.

As shown in [14] using such extensions may be not so advantageous as special hardware, but it can assist to reduce significantly the time consumption in comparison with non MMX/SSE implementation. Examples of using MMX/SSE for digital image processing and evaluation of its productivity are presented in [15, 16, 17, 18].

In our paper we use MMX/SSE technology for developing fast box filtering technique based algorithm to perform cross correlation calculation and employ this algorithm for real time optical flow estimation.

II. FAST CROSS CORRELATION ALGORITHM

The main aim of our paper was reduction of computational costs for calculation cross correlation function. First of all we choose computationally optimal method. Box filtering technique [7] is used in this method for fast cross correlation between frames.

A. Box filtering technique

Let f and g be normalized intensity's of pixels on consecutive frames. Cross correlation between regions in these images can be calculated using:

$$c_{ij,d_x d_y} = \frac{\mathrm{cov}_{ij,d_x d_y}(f,g)}{\sqrt{\mathrm{var}_{ij}(f) * \mathrm{var}_{ij,d_x d_y}(g)}}, \qquad (1)$$

where

$$\mathrm{cov}_{ij,d_x d_y}(f,g) = \sum_{m=j-K/2}^{j+K/2} \sum_{n=i-L/2}^{i+L/2} f_{n,m} * g_{n+d_x,m+d_y}, \qquad (2)$$

$$\mathrm{var}_{ij}(f) = \sum_{m=j-K/2}^{j+K/2} \sum_{n=i-L/2}^{i+L/2} f_{n,m}{}^2, \qquad (3)$$

$$\mathrm{var}_{ij,d_x d_y}(g) = \sum_{m=j-K/2}^{j+K/2} \sum_{n=i-L/2}^{i+L/2} g_{n+d_x,m+d_y}{}^2, \qquad (4)$$

i, j – center of matching region on f; dx, dy – shift of matching region on g; K, L – matching region size.

First and foremost we must perform the normalization. It is essential for reduction of the influence of brightness and contrast changes between frames in cross-correlation based methods. Computing the normalization in every correlation window requires an additional processing stage. We perform it throughout the image instead of local for every window. It is admissible because we process frames obtained with small time interval and when conditions may be changed with low probability.

Equations (2), (3) and (4) are double sums and can be calculated using box filtering technique. They may be represented as:

$$Hsum_{ij} = \sum_{n=i-L/2}^{i+L/2} Vsum_j(n), \qquad (5)$$

where

$$Vsum_j(n) = \sum_{m=j-K/2}^{j+K/2} a_{n,m} * b_{n,m}, \qquad (6)$$

$Vsum_i(n)$ is the sum of products of pixel intensity values from column n of window with center in (i,j) on image a with respective values on image b.

The main idea of the technique is that for calculation $Vsum_j(n)$ it is necessary to update $Vsum_{j-1}(n)$. This means to subtract multiplication of $a_{n,(j-K/2-1)}$ and $b_{n,(j-K/2-1)}$ from it (leave sum because of shifting window) and to add multiplication of $a_{n,(j+K/2)}$ and $b_{n,(j+K/2)}$ to it (enter the sum). For calculation $Hsum_{ij}$ it is necessary to update the value of $Hsum_{(i-1)j}$ in the same way. Such method requires four addition-subtraction operations for new value calculation regardless of window size. The scheme of computation for reviewed technique is presented in Fig. 1.

The realization of the reviewed approach requires accurate calculation strategy. First of all it is necessary to exclude repeated calculation of the same value. For example, every value $f_{m,n}{}^2$ is used at least two times: when it is added to the sum and when it is subtracted from it. We allocate special buffers in memory for such values.

Our algorithm works with grayscale images. It requires byte per pixel in memory for frames. Intermediate values that have to be stored require double word per pixel because they are obtained by summation of multiplications of bytes. We assign memory for pairs of buffers to store products, intermediate sums and sought sums (3) and (4). We use pairs because the algorithm calculates (3) and (4) at one pass. Also we allocate memory for L*K buffers to store sought sums (2). In this case the algorithm does L*K passes in cycle to fill each buffer and uses previously allocated for calculation (3) buffers to store products and intermediate sums. Therefore the algorithm requires $N*M*(2*3+L*K)$ double words in memory for buffers.

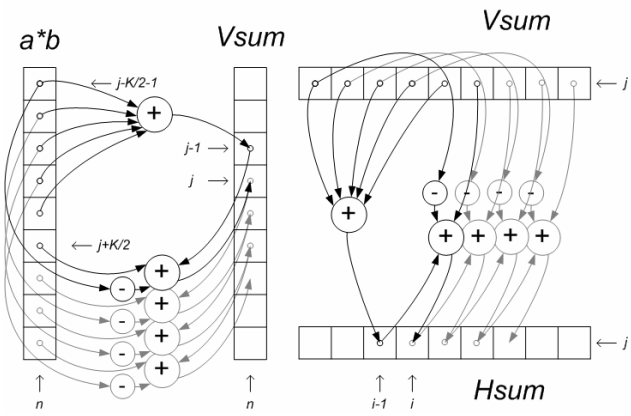

Fig. 1. Scheme of computation.

B. Algorithm

It is evident from Fig.1 that when *Hsum* is calculated in each pixel of the frame it is necessary to sum up L values of *Vsum* only once for every row. The calculation of *Vsum* requires summation of K products also only once for every column of the frame. This assertion leads to the next algorithm presented in pseudo code.

For x=1 **to** number of columns **do**

Compute $a*b$ for first K rows in column x

(without using cycle).

Compute *Vsum* in row 1 and column x by

summation K values obtained previous step

using (6).

End

For *y*=2 **to** number of rows **do**

For *x*=1 **to** number of columns **do**

Compute *a*b* in row *y* and column *x*.

Compute *Vsum* updating previous value *Vsum-1*

End

End

For *y=1* **to** number of rows **do**

Compute *Hsum* in column 1 and row *y* by summation *L* values *Vsum* using (5).

For *x*=2 **to** number of columns **do**

Compute *Hsum* updating previous value *Hsum-1*

End

End

C. Correlation coefficients calculation using SIMD extensions.

Te most computationaly expensive stage of cross correlation function calculation is computing of equation (1). It consists of three most complex operations: multiplication, division and square root computation. Fig. 2 shows computing strategy for calculation of equation (1) with using SSE commands. Organization of data storage together with SIMD parallelism allows computing four values per one pass in cycle. We achieved the largest rising in performance exactly at this stage.

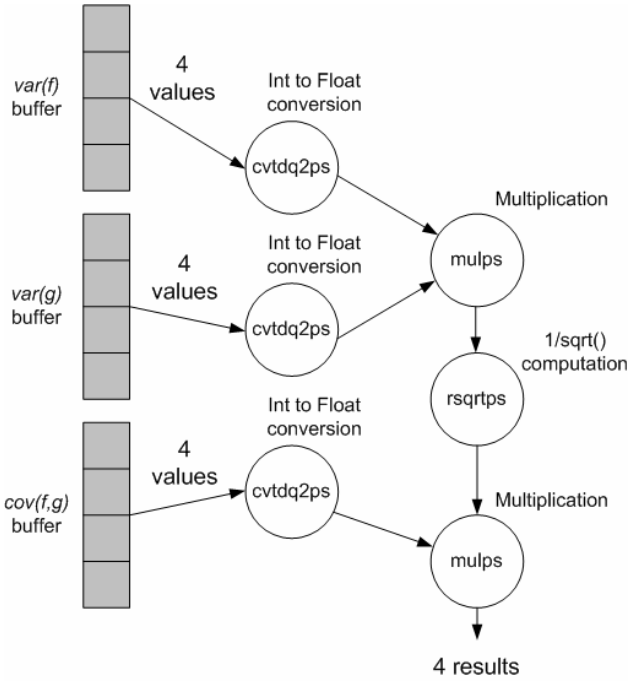

Fig. 2. Computing strategy to calculate equation (1).

D. Optical flow estimation.

We implemented simple correlation based method to estimate optical flow, and applied cross correlation metrics for matching regions on consecutive frames of video sequence. Our matching strategy uses correlation window to estimate correspondence of current pixel in surrounding area on previous frame. Optical flow in each point of frame is determined by maximum coefficient in sight area and size of this area depends on the size of the correlation window.

We realized the simple filter to reduce dynamic noise that appears because of discretization of analog video image. Filtration is done by comparison difference of pixel intensity values on consecutive frames with threshold. The pixel value is changed when the mentioned difference exceeds defined value. We use this approach before correlation coefficients calculation to obtain more accurate result.

III. RESULTS

A. Experimental system

All calculations described in this paper were performed under FreeBSD 5.0 on a processor Pentium 4 2.8 GHz with 512MB RDRAM. Performance evaluation was done with the frames obtained from analog video camera of surveillance system. Discretization of analog video was carried out by frame grabber with BT-878 chip. It allows to obtain digital video sequences with maximum size 768x576 (PAL) or 640x480 (NTSC) and frequency 25 fps.

B. Performance

The developed algorithm was employed for correlation coefficients computation in images with size 320*240 pixels. Computational cost of our algorithm in millions processor cycles is presented in Table 1. Correlation window size is 3*3 points in this case.

Table 1. Computational cost in millions processor cycles for C implementation and C/MMX/SSE implementation.

Algorithm stage	C	C/ MMX/ SSE	Profit
Dynamic noise reduction	3.5	0.4	8.75
Vsum computation for (3), (4)	4.4	2.8	1.57
Hsum computation for (3), (4)	3.7	2.1	1.76
Vsum computation for (2)	2.5	1.7	1.47
Hsum computation for (2)	2.1	1.1	1.9
Correlation coefficients calculation and maximum coefficients determination	284	16.7	17.6

Total time consumptions for different window sizes are represented in Table 2.

Table 2. Achieved time in ms. for C implementation and C/MMX/SSE implementation.

Window size (L*K)	Number of coefficients	C	C/ MMX/SSE
3*3	9	119	14
5*5	25	379	42

As evident from the results, increasing in performance does not depend on correlation window size. It is so because we use box filtering technique whose productivity is invariant from window size. Factor of productivity rising is approximately 8.5 times in both cases.

To determine the accuracy of the developed algorithm obtained coefficients were used for optical flow estimation according to the method described in section 2.4. Fig.3 shows initial frames and obtained optical flow maps. These maps were obtained by using correlation window with size 3*3 pixels. Time consumptions for optical flow computation method are represented in Table 3.

IV. CONCLUSION

We have developed fast algorithm to calculate cross correlation function with application MMX and SSE (SSE2) extensions of modern general purposes processors. Obtained results have shown high efficiency of such extensions in the field of computer vision. Our algorithm can be used in other correlation based techniques such as stereo matching, feature detection, object tracking and others.

Table 3. Achieved time in ms. for optical flow estimation.

Frame size	3*3	5*5
640*480	85 (10 fps)	159 (6 fps)
320*240	19 (52 fps)	45 (22 fps)
160*120	4 (250 fps)	7 (142 fps)

We have used our algorithm for optical flow estimation. The experiments allow us to calculate optical flow between frames of video sequence in real time with high frame rate.

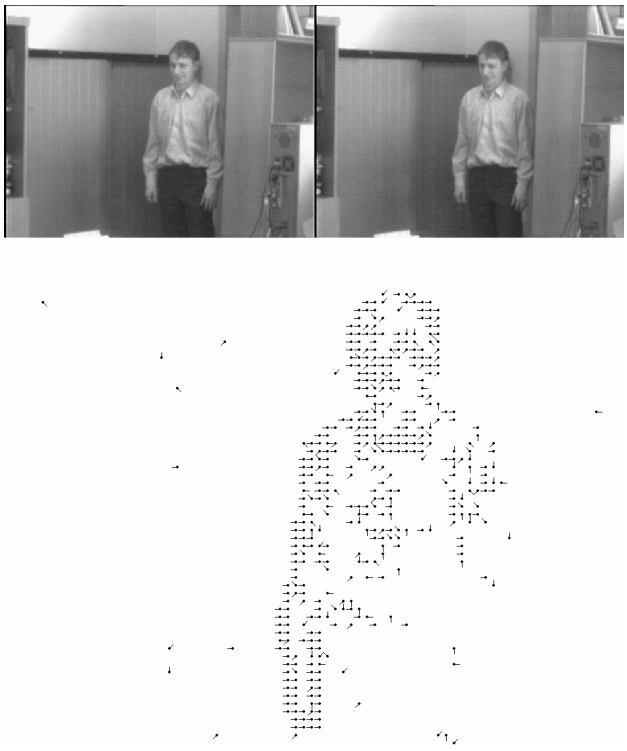

Fig. 3. Optical flow estimation procedure result.

REFERENCES

[1] B. Zitova J. Flusser Image registration methods: a survey", *Image and Vision Computing*, 21 (11) (2003). pp. 977-1000

[2] W. K. Pratt Correlation techniques of image registration, *IEEE Transactions on Aerospace and Electronic Systems,* 10 (1974). pp. 353–358.

[3] R. Zabih J. Woodfill Non-Parametric Local Transforms for Computing Visual Correspondence, *Proceedings 3rd European Conf. Computer Vision*, Stockholm, 1994, pp. 150-158.

[4] J. L. Barron D. J. Fleet S. S. Beauchemin Performance of optical flow techniques, *International Journal of Computer Vision*, 12(1) (1994), pp. 43-77.

[5] S. S. Beauchemin J. L. Barron Computation of optical flow, *ACM Computing Surveys*, 27 (3) (1995). pp. 433-467.

[6] J. Weng. Image matching using the windowed Fourier phase *International Journal of Computer Vision*, 11 (3) (1993). pp. 211-236.

[7] M. J. McDonnel Box-filtering techniques, *Computer Graphics and Image Processing*, 17 (3) (1981). pp. 65-70.

[8] H. Hirschmüller P. R. Innocent J. Garibaldi Real-time correlation-based stereo vision with reduced border errors, *International Journal of Computer Vision*, 47 (1-3) (2002). pp. 229 – 246.

[9] K. Muhlmann D. Maier J. Hesser R. Manner Calculating dense disparity maps from color stereo images, an efficient implementation, *International Journal of Computer Vision*, 47 (1-3) (2002). pp. 79 – 88.

[10] C. Sun Fast algorithms for stereo matching and motion estimation, *Proceedings of Australia-Japan Advanced Workshop on Computer Vision*, Adelaide, Australia, September 2003, pp.38-48.

[11] C. Sun Fast optical flow using 3d shortest path techniques, *Image and vision computing*, 20 (13/14) (2002). pp. 981-991.

[12] J. Germano R. Baptista L. Sousa Configurable platform for real time video processing and vision systems, *Proceedings of XX Conference on Design of Circuits and Integrated Systems (DCIS'05)*, Lisbonne, Portugal, 2005.

[13] R.Andraka A dynamic hardware video processing platform, *Proceedings of Conference on Reconfigurable Technology for Rapid Product Development and Computing*,November 1996, pp. 90-99.

[14] S. Persa P.P. Jonker Evaluation of two real time image processing architectures, *Proceedings of 6th Annual Conf. of the Advanced School for Computing and Imaging (ASCI 2000)*, Lommel, Belgium, June 2000, pp. 387-392.

[15] J. Skoglund M Felsberg Fast image processing using SSE2, *Proceedings of the SSBA Symposium on Image Analysis*, Malmö, March, 2005.

[16] V. Kravtchenko Using MMX technology in digital image processing, Technical Report and Coding Examples TR-98-13, Department. of Computer Science. The University of British Columbia.

[17] G. Conte S. Tommesani F. Zanichelli The long and winding road to high-performance image processing with MMX/SSE, *Proceedings of the In Fifth IEEE International Workshop on Computer Architecture for Machine Perception*, Padova, Italy, September 2000, p. 302.