Speeding-up NCC-based template matching using parallel multimedia instructions

Luigi Di Stefano, Stefano Mattoccia, Federico Tombari
Department of Electronics Computer Science and Systems (DEIS)
Viale Risorgimento 2, 40136 - Bologna, Italy
Advanced Research Center on Electronic Systems (ARCES)
Via Toffano 2/2, 40135 - Bologna, Italy
University of Bologna
Email: ldistefano@deis.unibo.it, smattoccia@deis.unibo.it, federicot@omniway.sm

***Abstract*— This paper describes the mapping of a recently introduced template matching algorithm based on the Normalized Cross Correlation (NCC) on a general purpose processor endowed with SIMD (Single Instruction Multiple Data) multimedia instructions. The algorithm relies on the Bounded Partial Correlation (BPC) technique, which consists in deploying a sufficient condition to detect unsatisfactory matching candidates at a reduced computational cost. First, we briefly describe the BPC technique and highlight the related expensive computations. Then, based on the analysis of the major SIMD multimedia instruction set extensions available nowadays, we define a processor-independent multimedia instruction set and show how to carry out the most expensive BPC calculations using these pseudo-instructions. Finally, we provide experimental results obtained mapping the proposed algorithm on a mainstream multimedia SIMD instruction set (i.e. MMX). We compare these results with those obtained with the *brute force* NCC algorithm. The results show that the BPC technique is suited for a parallel SIMD-style mapping and that its effectiveness can be significantly improved using the multimedia instructions available nowadays in most general purpose CPUs.**

I. Summary

Template matching is a fundamental task occurring in countless image analysis applications. As far as template matching is concerned, Normalized Cross Correlation (NCC) is often the adopted similarity measure (e.g. [1], [2], [3], [4], [5], [6]) due to its robustness with respect to photometric variations. Since with large size images and/or templates the matching process can be computationally very expensive, numerous techniques aimed at speeding up the basic algorithm have been proposed in literature [7], [8], [9]. However, these techniques imply a non exhaustive search process since they do not compare the full resolution image with the full resolution template at every search position and therefore can be trapped by local extremes resulting in wrong localization of the template. As for the Normalized Cross Correlation, we have shown recently [10], [11], [12] that it is possible to speed-up the exhaustive-search template matching process by the Bounded Partial Correlation (BPC) technique. In [12] BPC deploys a sufficient condition based on the Cauchy-Schwarz inequality to detect quickly unsatisfactory matching candidates.

Both BPC and the standard *brute force* NCC algorithm execute repetitive operations based on the *Multiply and Accumulate* (MAC) computations involved in the calculation of a dot-product term. If the pixels are relatively undersized with respect to the multimedia registries, the dot-product computation can be significantly accelerated with appropriate MAC instructions that parallelize the computation by carrying out several multiplications within a single instruction.

The paper is organized as follows: we first review the basic BPC technique, then we describe the most relevant aspects concerned with the mapping of the BPC algorithm on a parallel architecture. Then we provide experimental results comparing the performance of the scalar BPC technique with the performance of the same SIMD mapped algorithm. These results show that SIMD instructions are suited for the BPC algorithms allowing a significant further speed-up in NCC based template matching.

II. Review of the BPC technique

With Normalized Cross Correlation the template sub-image, T, is located into the image under examination, I, in order to determine the maximum of the NCC function (1). The dot-product term yielding the numerator of (1) represents the *cross correlation* between the template and the image, $C(x,y)$, its computation turning out to be the bottleneck in the evaluation of $\eta(x,y)$. In fact, the two terms appearing in the denominator represent the ℓ_2 norms of the sub-image under examination, $\|I(x,y)\|_2$, and of the template,$\|T\|_2$. The latter can be computed once at start up, the former can be obtained very efficiently using incremental calculation based on accumulating only 4 product terms per image point [13], [14], [15].

$$\eta(x,y) = \frac{\sum_{j=1}^{N}\sum_{i=1}^{M} I(x+i,y+j)\cdot T(i,j)}{\sqrt{\sum_{j=1}^{N}\sum_{i=1}^{M} I(x+i,y+j)^2}\cdot\sqrt{\sum_{j=1}^{N}\sum_{i=1}^{M} T(i,j)^2}} \quad (1)$$

Suppose now that a function $\beta(x,y)$ exists such that $\beta(x,y)$ is an upper-bound for $C(x,y)$:

$$\beta(x,y) \geq C(x,y) = \sum_{j=1}^{N} \sum_{i=1}^{M} I(x+i, y+j) \cdot T(i,j), \quad (2)$$

by normalizing β we obtain an upper-bound for the NCC function:

$$\frac{\beta(x,y)}{\|I(x,y)\|_2 \cdot \|T\|_2} \geq \frac{C(x,y)}{\|I(x,y)\|_2 \cdot \|T\|_2} = \eta(x,y) \quad (3)$$

Then, indicating as η_{max} the current correlation maximum, if the following inequality is verified at point (x, y):

$$\frac{\beta(x,y)}{\|I(x,y)\|_2 \cdot \|T\|_2} < \eta_{max} \quad (4)$$

then the matching process can proceed with the next point without carrying out the calculation of $C(x, y)$, as that position is guaranteed not to correspond to the new correlation maximum. Hence, (4) is a sufficient condition for skipping the points that cannot improve the degree of matching with respect to the current maximum without calculating the actual cross correlation. Conversely, if (4) is not verified then it is necessary to compute $C(x, y)$, normalize it by the product $\|I(x,y)\|_2 \cdot \|T\|_2$ and check the *new maximum condition*:

$$\frac{C(x,y)}{\|I(x,y)\|_2 \cdot \|T\|_2} \geq \eta_{max} \quad (5)$$

This approach, referred to as BPC, allows for reducing significantly the number of operations required to carry out exhaustive template matching by deploying a suitable sufficient condition (i.e. equation (4)) where function β is derived from the Cauchy-Schwartz inequality [11].

III. THE SIMD MAPPING

The use of multimedia instructions allows for reducing the execution time of the BPC algorithm as well as of the *brute force* NCC algorithm. In particular, the most expensive computation involved in the evaluation of the NCC function is represented by the evaluation of the dot product term $C(x, y)$. As for BPC, when the sufficient condition (4) holds only a portion of the dot product term is calculated, otherwise this quantity must be integrated with the residual dot-product term (see [12] for more details).

The mapping of the *brute force* NCC algorithm and of the BPC algorithm with multimedia instructions [16] is obtained considering the case of images with 8-bit pixels. Therefore, the optimization of the algorithm by means of multimedia instructions is achieved by implementing the internal sums and multiplications (MAC) of the dot product term with appropriate parallel SIMD-style instructions. More precisely, the optimization is made row by row and a suitable procedure for indexing the correct lines and padding the vectors is adopted. In order to describe the proposed mapping we make use of machine-independent pseudo-code, considering the typical case of a general purpose processor with 64-bit data parallelism for SIMD instructions ([17], [18], [19], [20], [22], [23], [24]), though some recent machines provide a higher degree of parallelism (i.e 128 bits for Intel Pentium 4 and Motorola Power PC). The extension of the proposed implementation to 128-bit or higher data parallelism is straightforward.

Under these circumstances, we assume that the 64 bits stored in a multimedia register (or in memory) can be seen as eight bytes (B), four words (W), two double words (D) or one quadword (Q). Multimedia registers will be referred to as MR0, ..., MR5. The SIMD instructions working on multimedia data will be characterized by the prefix M, with the type of the data items processed in parallel specified through a suffix following the name of the operation. Hence, in our pseudo-language a generic SIMD parallel instruction will be expressed as:

```
M_(OP)_(T) op#1; op#2
```

where $OP \in \{AND, OR, NOT, ADD, SUB, ...\}$ encodes the operation and $T \in \{Q, D, W, B\}$ specifies the type of the native data items packed into operands *op♯1* and *op♯2*. Both operands can be multimedia registers or memory locations, and the first acts as source and destination.

We will assume that, thanks to appropriate passages of variables, the routine disposes of 4 already initialized global parameters:

- *assindex* : address of the 32-bit index that contains the number of iterations to be done;
- *tot* : address of the 64-bit accumulator that contains the sum of products computed so far;
- *gi* : pointer to the first element of the current row of the sub-image
- *gt* : pointer to the first element of the current row of the template

Load and store operations between memory and multimedia registers, as well as data-transfers between multimedia registers, will be handled by the (M_MOV op♯1, op♯2) instruction, which transfers 64 bits from the source operand (*op♯2*) to the destination operand (*op♯1*). Both can be a memory location or a multimedia register. This is the only instruction of our pseudo-language that does not require specification of the type of the data items packed into the operands, as it works only with quadwords.

Once loaded the index of the iterations on 32-bit register R0 and the accumulator on a multimedia 64-bit register:

```
MOV R0, [assindex]
M_MOV MR2, [tot]
```

the iterative cycle can start:

```
_loop:

 ; the counter gets decremented
 DEC R0
 ; 8 elements of the sub_image
 ; are loaded on MR0
 MOV R1,[gi]
 M_MOV MR0,[R1+8*R0]

 ; 8 elements of the template
 ; are loaded on MR1
 MOV R1,[gt]
 M_MOV MR1, [R1+8*R0]
```

As the native data are byte items, representing pixels both of the sub-image and of the template, MAC instructions need to work with word items in order to preserve possible overflows derived by multiplications between bytes. SIMD extensions generally contain instructions used to pack and unpack data within multimedia registers, mostly to allow casts from different data types. Using these instructions the iterative cycle can be divided in 2 parts, where first the 4 higher then the 4 lower bytes are unpacked in 4 words, whose higher 8 bits are set to 0.

We define two groups of instructions: one to unpack the lower 32 bits (L) of the register, the other to unpack the higher (H) 32 bits. Hence, the generic unpacking instruction will be expressed as (M_UNPCK_(P)_(T) op♯1) where $P \in (H; L)$ is used to denote unpacking of the higher or lower 32 bits, $T \in (D; W; B)$ specifies the original data type and $op\sharp1$ the multimedia register to be unpacked. For example, the following instruction unpacks the lower 4 bytes within MR0 into four words:

```
M_UNPCK_L_B MR0
```

with the most significant bytes of the resulting word data items set to zero. The next section of code is relative to the unpacking operations performed on the data loaded in the previous part:

```
 ; the 8 image elements are
 ; stored on MR4 for later use
 M_MOV MR4,MR0

 ; the higher 4 bytes are unpacked
 M_UNPCK_H_B MR0

 ; the 8 template elements are
 ; stored on MR5 for later use
 M_MOV MR5, MR1

 ; the higher 4 bytes are unpacked
 M_UNPCK_H_B MR1
```

We define the MAC instruction applied on word items as (M_MAC_W op♯1, op♯2), where W specifies that the original data type is word, where $op\sharp1$ and $op\sharp2$ specify the 2 word operands and $op\sharp1$ the destination register. After executing the 4 multiplications between correspondent items, the instruction stores into the destination register 2 doublewords that represent, respectively, the sums of the 2 higher and the 2 lower multiplications (see Fig. 1).

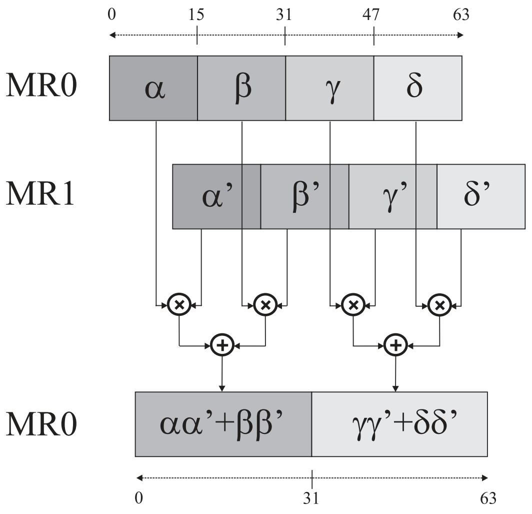

Fig. 1. *The parallel MAC instruction.*

```
 ; MAC of the higher 4 elements
 M_MAC_W MR0, MR1

 ; update of the accumulator
 M_ADD_D MR2, MR0
```

Once the first 4 pixels of both the template and the image have been multiplied and summed between each other, and then added to the current accumulator value, it is possible to adopt a similar procedure to compute the remaining 4 pixels that have been previously stored into the registers named $MR4$ and $MR5$:

```
 ; the lower 4 elements of the
 ; sub-image and template are unpacked
 M_UNPCK_L_B MR4
 M_UNPCK_L_B MR5

 ; MAC of the lower 4 elements
 M_MAC_W MR4, MR5

 ; update of the accumulator
 M_ADD_D MR2, MR4
```

At the end of the iterative cycle, the counter value is tested to verify if there are other pixels in the current row that is being computed. In case no more pixels are left, the accumulator *tot* can be updated with a memory access.

```
 ; testing if there are more
 ; elements on the row
 CMP R0, 0
```

```
JNE _loop

; final update of the accumulator
; with the current total
M_MOV [tot], MR2
```

The described cycle can be easily deployed by the BPC (or standard NCC) algorithm to perform a row-by-row correlation between the pixels belonging to the image and to the template. As it can be seen, the cycle has been optimized in order to require the minimum number of memory accesses for each computed row (i.e. 2 load operations and 1 store operation). As it can be easily pointed out, working with bigger registries would mean processing more pixels in the same time, hence increasing the speed up of the correlation. In case a greater parallelism can be handled by the processor, the code can be simply modified by increasing the number of pixels that are being computed during each of the two stage of the algorithm.

IV. Experimental results

The experimental results have been obtained mapping the *brute force* NCC and BPC algorithms into the MMX multimedia instruction set extension [18] and running the benchmarks on a Linux workstation based on a AMD Thunderbird 900 MHz processor. Table I shows the speed-up with respect to the C-coded *brute force* NCC algorithm yield respectively by the SIMD-coded *brute force* NCC algorithm (NCC_{SIMD}), the C-coded BPC algorithm (BPC_C) and the SIMD-coded BPC algorithm (BPC_{SIMD}). These measures have been obtained on three images called as "albert", "pcb3", "plants" that belong to the data set used in [10] and are shown in Figures 2, 3 and 4.

Image	NCC_{SIMD}	BPC_C	BPC_{SIMD}
pcb3	2.78	2.21	5.66
plants	2.50	2.54	5.96
albert	2.51	3.28	7.40

TABLE I
MEASURED SPEED-UPS ON AMD THUNDERBIRD 900 MHZ CPU

As shown by the table, the proposed SIMD implementation is effective in speeding-up NCC-based and BPC-based template matching at least by a factor of 2. The use of both BPC and multimedia SIMD instructions can dramatically accelerate exhaustive NCC-based template matching process, with measured speed-ups ranging from 5.66 to 7.40. It is also worth pointing out that the SIMD mapping results more effective with the NCC standard algorithm: this is due to the fact that the sufficient condition (4) embodied in the BPC algorithm allows for skipping several MAC operations involved in the whole dot product computation, with these operations representing the bulk of the proposed SIMD optimization.

V. Conclusion

We have described the mapping of the BPC algorithm on a general purpose processor by means of multimedia instructions. The BPC technique represents a novel technique capable of rapidly rejecting mismatching positions thanks to the deployment of a sufficient condition based on the Cauchy-Schwarz inequality. The related mapping has been described by means of a processor-independent SIMD instruction set, and it can also be used to optimize any algorithm that makes use of the NCC function. Experimental results show that the benefit of the original BPC technique can be effectively improved using the described mapping.

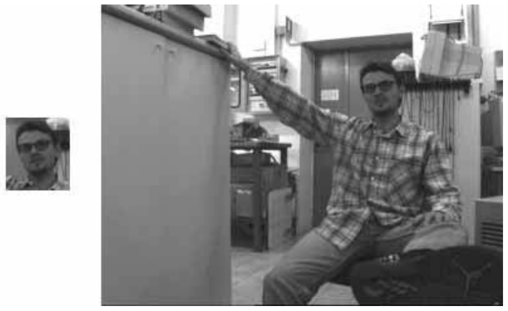
Fig. 2. *albert*: $W \times H = 320 \times 240$, $M \times N = 51 \times 58$, $\eta_{max} = NCC(198,43) \simeq 0.995$, $\eta_{Z_{max}} = ZNCC(198,43) \simeq 0.9592$

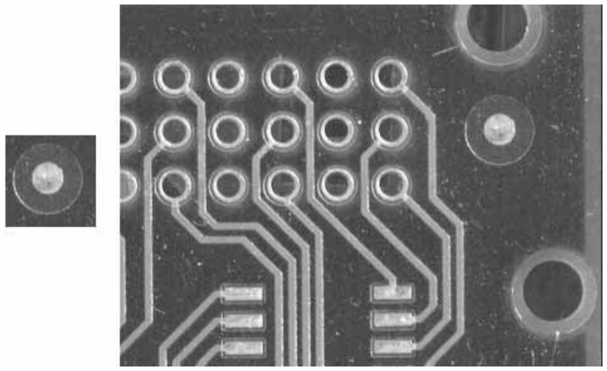
Fig. 3. *pcb3*: $W \times H = 384 \times 288$, $M \times N = 72 \times 73$, $\eta_{max} = NCC(268,65) \simeq 0.997$, $\eta_{Z_{max}} = ZNCC(268,65) \simeq 0.9724$

Acknowledgment

We wish to thank Autonomous Systems Center (VASC) at Carnegie Mellon University for the use of image "plants" .

References

[1] Changming Sun, "Fast Optical Flow Using 3D Shortest Path Techniques" *Image and Vision Computing*, Vol. 20 (13/14), 981-991, 2002, 981-991
[2] Changming Sun, "Fast Stereo Matching Using Rectangular Subregioning and 3D Maximum-Surface Techniques" *Int. Journal of Computer Vision*, Vol. 47(1/2/3), 2002, 99-117
[3] Changming Sun, Shmuel Peleg, "Fast panoramic stereo matching using cylindrical maximum surfaces", *IEEE Trans. on Systems, Man and Cybernetics Part B*, Vol. 34(1), Feb. 2004, 760-765
[4] O. Faugeras, B. Hotz, H. Mathieu, T. Viille, Z. Zhang, P. Fua, E. Theron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, C. Proy, "Real time correlation-based stereo: Algorithm, implementations and applications", *INRIA, Tech. Rep. RR-2013*, 1993.
[5] Du-Ming Tsai, Chien-Ta Lin, "Fast normalized cross correlation for defect detection", *Pattern Recognition Letters*, Vol. 24, 2003, 2625-2631

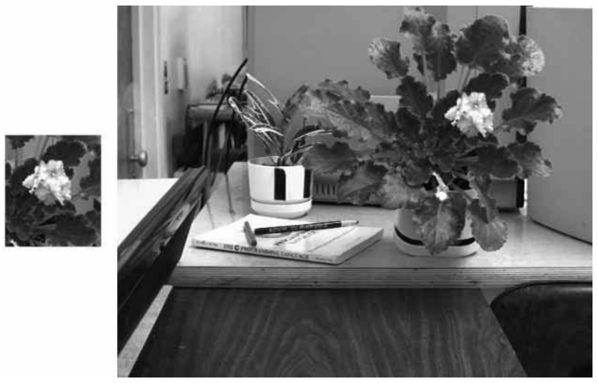

Fig. 4. *plants*: $W \times H = 512 \times 400$, $M \times N = 104 \times 121$, $\eta_{max} = NCC\,(333, 66) \simeq 0.986$, $\eta_{Z_{max}} = ZNCC\,(333, 66) \simeq 0.9576$

[6] Du-Ming Tsai, Chien-Ta Lin, Jeng-Fung Chen, "The evaluation of normalized cross correlations for defect detection", *Pattern Recognition Letters*, Vol. 24, 2003, 2525-2535

[7] A. Rosenfeld, G.J. Vanderburg, "Coarse-Fine template matching", *IEEE Trans. on Sys., Man and Cyb.*, Vol. 7, 1977, 104-197

[8] A. Rosenfeld, G.J. Vanderburg, "Two-stage template matching", *IEEE Trans. on Image Processing*, Vol. 26, 1977, 384-393

[9] W. Krattenthaler, K.J. Mayer, M. Zeiler, "Point correlation: a reduced-cost template matching technique" *1st IEEE Int. Conf. on Image Processing (ICIP 1994)*, Vol. I, September, 1994, Austin, Texas, USA, 208-212

[10] L. Di Stefano, S. Mattoccia, "Fast Template Matching using Bounded Partial Correlation", *Machine Vision and Applications*, Vol. 13, 2003, 213-221

[11] L. Di Stefano, S. Mattoccia, M. Mola "An Efficient Algorithm for Exhaustive Template Matching based on Normalized Cross Correlation", *IAPR Int. Conf. on Image Analysis and Processing (ICIP 2003)*, September 17-19, 2003, Mantova, Italy, pp 322-327

[12] L. Di Stefano, S. Mattoccia, "A sufficient condition based on the Cauchy-Schwarz inequality for efficient Template Matching", *IEEE Int. Conf. on Image Processing (ICIP 2003)*, September 14-17, 2003, Barcelona, Spain

[13] M. J. Mc Donnell, "Box-Filtering Techniques", *Computer Graphics and Image Processing*, Vol. 17, 1981, 65-70

[14] P. Viola, M. Jones, "Rapid Object Detection using Boosted Cascade of Simple Features", *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR 2001)*, Vol. I, 2001, 511-518

[15] O. Veksler, "Fast Variable window for Stereo Correspondence using Integral Images", *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR 2003)*, Vol. I, 2003, 556-561

[16] Ruby B. Lee, "Multimedia Extensions for General-Purpose Processors", *IEEE Workshop on Signal Processing Systems Design and Implementation*, 1997, Leicester, United Kingdom, 9–23

[17] Lee R. B., "Subword Parallelism with MAX-2", *IEEE Micro*, 16(4), 1996, 51–59

[18] Peleg A. and Weiser U., "MMX Technology Extension to the Intel Architecture", *IEEE Micro*, 16(4), 1996, 42–50

[19] Tremblay M., O'Connor M., Narayanan V., He L., "VIS Speeds New Media Processing", *IEEE Micro*, 16(4), 1996, 10–20

[20] Oberman S., Favor G., Weber F., "AMD 3DNow ! Technology: Architecture and Implementations", *IEEE Micro*, 19(2), 1999, 37–48

[21] Diefendorff K., Dubey P.K., Hochsprung R., Scale H., "VIS Speeds New Media Processing", *IEEE Micro*, 20(2), 2000, 85–95

[22] Sharangpani H., Arora K., "Itanium Processor Microarchitecture", *IEEE Micro*, 20(5), 2000, 24–43

[23] Shreekant T. and Huff T., "Implementing streaming SIMD extensions on the Pentium III processor", *IEEE Micro*, 20(4), 2000, 47–57

[24] Advanced RISC Machines, www.arm.com