A PC-based Real-Time Stereo Vision System

L. Di Stefano, M. Marchionni, S. Mattoccia

Department of Electronics Computer Science and Systems (DEIS)
Viale Risorgimento 2, 40136 Bologna, Italy
University of Bologna

Advanced Research Center on Electronic Systems "Ercole De Castro" (ARCES)
Via Toffano 2/2, 40135 Bologna, Italy
University of Bologna

email: ldistefano@deis.unibo.it, mmarchionni@litio.it, smattoccia@deis.unibo.it

Abstract. This paper describes a stereo vision system that enables real-time dense depth measurements on a personal computer. The system relies on a very efficient stereo matching engine that, unlike many other approaches which use two distinct matching phases in order to detect unreliable matches, uses a single matching phase. Our matching engine allows for rejecting most unreliable matches by exploiting violations of the uniqueness constraint as well as analysing behaviour of correlation scores. Real-time capability has been achieved by deploying very efficient incremental calculation schemes aimed at avoiding redundant calculations and parallelising the computationally expensive portion of the code with Single Instruction Multiple Data (SIMD) parallel instructions, available nowadays on almost any state-of-the-art general purpose microprocessors. Experimental results on real stereo sequences and preliminary results concerning a 3D people tracking/counting application show the effectiveness of the proposed PC-based stereo vision system for real-time applications.

Key words: stereo, real-time, dense depth map, 3D, SIMD.

1. Introduction

Several applications, such as intelligent surveillance, people/objects tracking, autonomous vehicles, safety, haptic devices, robot control, can take advantage of real-time depth measurements. However, in the past decade real-time dense depth measurements could be achieved only by expensive, power consuming and large dedicated machines that were unsuited for most applications. Advances in general purpose microprocessor technology, such as higher clock frequencies, pipelined architectures, high speed buses and the availability of *Single Instruction Multiple Data* (SIMD) parallel capabilities, as well as the advent of cheap, high quality imaging sensors, allow nowadays for obtaining real-time depth measurements on standard general purpose machines.

Real-time dense depth measurements are currently feasible only with *local* algorithms (e.g. [2, 5, 11, 30]) which assign disparity values at each pixel on the basis of the photometric properties of the neighbouring pixels. These algorithms use a *local support* area,

referred to as *correlation window*, and exhibit regular computational structures that allow for deploying efficient incremental calculation schemes as well as SIMD parallel capabilities available in state-of-the-art general purpose microprocessors, such a those by Intel [8,24,25], AMD [18], Sun [9], Hewlett-Packard [7], Motorola [20] and those based on the ARM architecture [39].

Bidirectional matching (BM) [1] is a widely adopted method [2, 11, 30] for detecting unreliable matches in local algorithms. Althought it has proven to be effective in discarding several wrong matches necessarily yielded by local algorithms in presence of occlusions [11, 21, 30], bidirectional matching is characterised by a significant computational cost due to the presence of two distinct matching phases.

It is worth observing that *global* algorithms (e.g. [28]), which assign disparity values minimising a global cost function, provide more accurate dense depth measurements compared to local algorithms. Nevertheless, global algorithms have much higher computational costs that render this class of algorithms unsuited to real-time applications.

This paper describes a stereo vision system that enables real-time stereo applications on a standard personal computer (PC) by exploiting efficient calculation schemes and deploying the SIMD parallel capabilities available in general purpose microprocessors. The system relies on a local algorithm that detects unreliable matches using a single matching phase (SMP).

The paper is organized as follows. In Section 2 we review real-time stereo vision systems based on dedicated hardware as well as systems running on standard PCs. In Section 3 we describe the main blocks of the overall stereo system, while in Section 4 we describe the single matching phase stereo algorithm. The computational optimisation aimed at avoiding redundant calculations are reviewed in Section 5, the detailed description of the mapping of the optimised algorithm on a general purpose processor with SIMD extension is described in Section 6, and details concerning the implementation of the match reliability tests are described in Section 7.

Finally, Section 8 provides experimental results obtained by the SMP algorithm on real stereo sequences acquired in our laboratory. We also provide preliminary results of a 3D people counting application based on the proposed stereo vision system.

2. Previous work

This section provides an overview of the most relevant real-time stereo vision systems based on local algorithms proposed in the last decade. We analyse systems based on dedicated hardware as well as systems based on standard PCs.

The DSP based stereo system developed at INRIA by Faugeras et al. [2] uses a trinocular imaging system and the normalized cross correlation (NCC) as matching measure. Match validation is carried out by bidirectional matching and analyzing the peaks of the correlation function. Kanade et al. [5] developed a DSP based polynolcular system (up

to 6 cameras) capable of providing video-rate depth maps with 200×200 pixels images. The similarity measure was the Sum of Squared Differences (SSD), and a global value (SSSD) was computed adding the contributions given by the different matches between the various views. Interval Research [13] developed a FPGA-based board which fits into a standard PCI board. The system executes 42 disparity measurements per second, using as input a stereo pair from two cameras at 320×240 pixels of resolution. The matching core is based on the census transform [13]. Recently, Tyzx [45] proposed a compact system based on a similar algorithm and composed of a digital stereo camera and a FPGA-based board which fits into a standard PCI board. Tyzx stereo system delivers a maximum frame rate of 200 fps with 512×512 stereo pairs. Another FPGA-based stereo system which relies on the census transform was developed by Corke et al. [16]. It delivers 30 fps with 256×256 stereo pairs. Sarnoff Corporation developed a video processing chip [26] capable of performing video rate stereo matching with two images of 320×240 pixels. The stereo module uses the Sum of Absolute Differences (SAD) as similarity measure and can deploy the left-right consistency check at the cost of doubling the computational load. The SAZAN [17] machine consists of two PCI processing boards and a polynocular imaging system composed of nine cameras with a resolution of 320×240 pixels. The SAZAN stereo vision system combines multiple SAD measures (SSAD), provides sub pixel accuracy delivering 20 frames per second with image size of 200×200 and a disparity range of 25 pixels. Darabiha et al. [37] developed an FPGA-based real-time stereo vision system capable of delivering 30 fps with 256×360 stereo pairs using a multi-resolution, multi-orientation phase based algorithm [4]. The MSVM-II vision machine developed by Jia et al. [36] is a very compact (10×10 cm), stand alone multi baseline (up to 8 cameras in line configuration) stereo system. The system pre-processes images, correcting lens distortion and performing LoG filtering, finds corresponding points using the SSAD and provides sub-pixel depth measurements. It is based on a 80 MHz FPGA and delivers up to 30 fps when processing 640×480 images and up to 50 fps processing 320×240 images with a disparity range of 64 pixels. Subsequently, the same authors proposed MSVM-III [38] — a trinocular stereo vision system in a triangular configuration. MSVM-III is based on a 60 MHz FPGA, and with a disparity search range of 64 pixels delivers 30 fps and up to 120 fps, respectively, with 640×480 and 320×240 stereo pairs. The system developed at SRI [11] provides real-time stereo matching on standard personal computers as well as on a DSP board. The SRI system uses two images, which are preprocessed via the LoG filter and matched using the SAD measure. Point Grey Research [43] developed a polynocular (up to three cameras) stereo algorithm based on the SAD measure. Both systems rely on two matching phases (i.e. bidirectional matching) and run on a personal computer. The omnidirectional imaging system developed by Tanahashi et al. [27] uses 60 cameras and the Point Grey Research algorithm for obtaining real-time, ominidirectional depth measurements. The system proposed by Gluckman et. al [15] uses two cameras and two parabolic mir-

rors for generating real time ominidirectional depth measurements. The stereo matching algorithm relies on the SAD measure, and with 600×60 stereo pairs and a disparity range of 32 pixels delivers 7 fps on a 300 MHz Pentium II processor exploiting SIMD capabilities. Other recently proposed PC-based real-time stereo vision systems based on the SAD measure, bidirectional matching and SIMD capabilities are [30, 32, 33]. In [33] an efficient method aimed at reducing the *border-localization* problem is also proposed. Finally, it is worth pointing out that the OpenCV library [46] includes a real-time implementation of the Birchfield and Tomasi [19] algorithm based on dynamic programming. Interested readers may find in [48] a comparative study of some PC-based real-time dense stereo vision algorithms.

3. The overall system

In this section we describe the main blocks, summarized in Figure 1, of the overall real-time stereo vision system proposed in this paper. The system acquires a stereo pair using two synchronized cameras; we use a monochrome digital *Mega-D Stereo Head* manufactured by Videre Design [44], which allows for acquiring, via the IEEE 1394 interface (also referred to as *firewire* or *i-link*), up to 1280×960 stereo pairs. Nevertheless, an equivalent stereo imaging system may be build with two relatively inexpensive digital cameras provided with an external trigger capability and connected to the PC through a high speed digital interface, such as IEEE 1394 or USB 2.0. The stereo matching algorithm assumes stereo pairs in *standard form*, that is, corresponding to epipolar lines lying on corresponding image scanlines, in order to reduce the search domain of homologous points (i.e., pixels in the two images belonging to the same point of the scene). Since the stereo pair acquired by the stereo cameras suffers from radial distorsion as well as from imperfect spatial alignment of the two cameras, a transformation known as *rectification* (see [14, 23] for details) must be applied in order to obtain a pair of images in standard form from the original ones. This transformation can be carried out in real-time if the *intrinsic* and *extrinsic* parameters [14] of the stereo imaging system are known. A procedure known as *calibration* [14], using some stereo pairs of known patterns, allows for estimating the parameters of the stereo imaging system. In this work calibration was carried out offline using the Matlab Camera Calibration Toolbox [40]. The data set used for calibrating the Mega-D Stereo Head as well as the intrinsic and extrinsic parameters estimated by the Matlab Camera Calibration Toolbox are available at [41].

The stereo pairs in standard form resulting from the *rectification* step are processed by the matching algorithm, which consists of the four major steps depicted in Figure 1 within the dashed box. In the *pre-processing* step the rectified images are normalized by subtraction of the mean values of the intensities computed in a small window centered at each pixel. This step allows for compensating for different settings of the cameras and-or different photometric conditions. Moreover, since matching turns out to be unreliable in

poorly textured areas, the variance of the intensities is calculated at each pixel based on a window of the same size as that used to obtain the mean values. This cue is used to detect regions with lack of texture [2]. The normalized images are matched according to the matching algorithm described in the next section, using the SAD error function. The reliability of the matches provided by the basic matching engine is improved by means of two tests, referred to as *distinctiveness* and *sharpness*, described in the remainder. In addition, the *match reliability tests* step uses the variance map computed in the pre-processing step to reject the matches found in poorly textured areas. The final *sub-pixel measurement* step performs sub-pixel refinement of disparities (up to 1/16) using a second degree interpolation function between the two scores closest to the the minimum. This step is carried out using integer operations only.

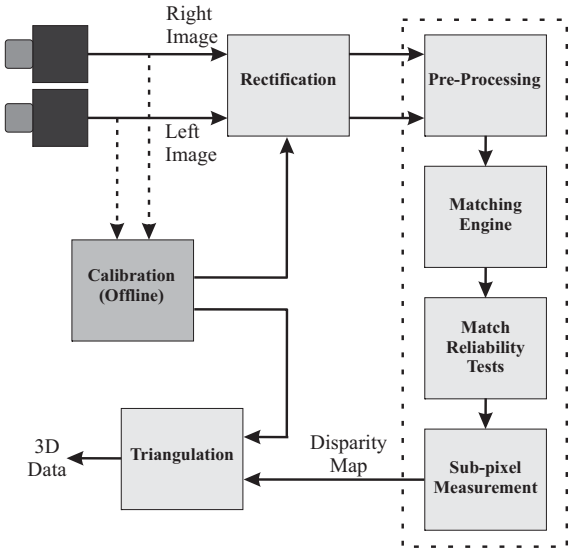

Fig. 1. Description of the overall stereo vision system.

Finally, after the sub-pixel measurement step, the disparity map generated by the stereo matching algorithm and the intrinsic and extrinsic parameters obtained with the offline calibration procedure enable for obtaining the 3D positions of homologous points by means of a procedure known as *triangulation* (see [14] for details). Triangulation is carried out by a routine written in C that assigns the (X,Y,Z) coordinates with respect to a defined reference coordinate system to each point (x,y) of the disparity map marked as valid.

4. The SMP matching engine

In *local* algorithms, given a point in the reference image, the homologous point is selected by searching along the corresponding scanline, of size W, in the other image, and within a certain disparity range, for the point that minimizes (maximizes) an error (similarity) function encoding the degree of dissimilarity (similarity) between two small regions of the stereo pair. In our algorithm, referred to as SMP, we use the SAD error function, which has proven to be a trade-off between reliability and computational cost. Nevertheless, other error functions (e.g., SSD), similarity functions (e.g., NCC) or non parametric functions (e.g. Census) could be used as well. Unlike algorithms based on bidirectional matching, which rely on a direct (i.e., left-to-right) and a reverse (i.e. right-to-left) matching phase, our algorithm uses only a direct matching phase. Our approach relies on the *uniqueness constraint*, which states that a 3D point can be projected at most to one point of each image of the stereo pair, as well as on the ability of modifying disparity measurements dynamically as long as the matching process proceeds.

Assume the left image as a reference one, and assume further that the disparity, d, belongs to the interval $[0..d_{max}]$ and that the left image is scanned from top to bottom and from left to right during the matching process. Starting from one point of left image, say $L(x-d_{max}, y)$, SMP searches within the interval $[R(x-d_{max}, y)..R(x, y)]$ for the best match candidate, evaluating the score of the SAD function for $x \in [d_{max}..(W - d_{max})]$. Then, for the successive point of reference image $L(x + 1 - d_{max}, y)$, the procedure is repeated to search for the best match within $[R(x+1-d_{max}, y)..R(x+1, y)]$. This process is iterated for the successive points along the scanline. Suppose now that the best match found for $L(x+\beta-d_{max}, y)$ is $R(x, y)$, with the score $SAD(x+\beta-d_{max}, x, y)$. The match established between $L(x+\beta-d_{max}, y)$ and $R(x, y)$ is referred to as $L(x+\beta-d_{max}, y) \rightsquigarrow R(x, y)$. It is well known that the photometric properties encoded by the SAD function as the main cue for establishing matches may be ambiguous (i.e. due to photometric distortions, occlusions and noise). Neverthelees, wrong matches expose inconsistencies within the set of already established matches that can be deployed to detect and discard them. Thus, suppose that another point of the left image, say $L(x + \alpha - d_{max}, y)$, with $\alpha \leq \beta$, has been previously matched to $R(x, y)$ with score $SAD(x + \alpha - d_{max}, x, y)$. This situation violates the uniqueness constraint, and with SMP allows for detecting wrong matches. In fact, based on the uniqueness constraint we assume that at least one of the two matches, i.e., either $L(x + \beta - d_{max}, y) \rightsquigarrow R(x, y)$ or $L(x + \alpha - d_{max}, y) \rightsquigarrow R(x, y)$, is wrong, and retain the match having the better score. Thus, if the currently analyzed point $L(x + \beta - d_{max}, y)$ has a better score than $L(x + \alpha - d_{max}, y)$ (i.e., $SAD(x+\beta-d_{max}, x, y) \leq SAD(x+\alpha-d_{max}, x, y)$), SMP will reject the previous match and accept the new one. This implies that the SMP algorithm allows for recovering from possible previous matching errors in a single matching phase.

The reliability of the disparity measurements provided by the basic matching core can

be improved by incorporating additional constraints. Since the SMP algorithm is aimed at real-time applications, we introduce new constraints by analysing the behaviour of the SAD function. This allows us to assess the reliability of the matches at a very small computational cost by exploiting the data already computed by the matching core, as well as the SIMD-processing capabilities provided by general purpose processors. In the SMP algorithm, the global minimum of the SAD error function is located very quickly using a parallel technique, described in Section 6, that with a few SIMD instructions yields the score (SAD_{min}) and the position within the disparity range (d_{min}) of the global minimum, as well as the scores (SAD_1, SAD_2, SAD_3) and positions (d_1, d_2, d_3) of three candidate minima, referred to as *pseudo-minima*. These exhibit small error scores but are not guaranteed to correspond to local minima. To discard ambiguous matches, we analyse the behaviour of the SAD function by means of two tests, called *distinctiveness test* and *sharpness test*, which are carried out using only the global minimum and three *pseudo-minima*. When the three *pseudo-minima* fall far from the position of the global minimum, the match is potentially ambiguous (for example, this occurs in presence of repetitive patterns), unless the error score of the global minimum is much smaller than those of the *pseudo-minima*. On the other hand, when the *pseudo-minima* are close to the position of the global minimum, the match can be considered reliable even though the score of the global minimum turns out to be not much smaller than those of the *pseudo-minima*.

The following relationship, referred to as *sharpness test*, evaluates the degree of aggregation of the *pseudo-minima* in proximity of the global minimum:

$$\Delta d = \sum_{i=1}^{3} |d_i - d_{min}| \tag{1}$$

A low Δd value (the lowest value is 4) indicates that the *pseudo-minima* are localized in proximity of the global minimum, and thus the match is accepted as reliable. Conversely, a high Δd value means that the *pseudo-minima* are spread within the disparity range, and hence the match is potentially ambiguous. In order to evaluate the reliability of the matches that do not satisfy the *sharpness test*, we perform an additional test, referred to as the *distinctiveness test*, aimed at evaluating whether the score of the global minimum is much smaller than those of the *pseudo-minima*. The following relation

$$\Delta SAD = \sum_{i=1}^{3} (SAD_i - SAD_{min}) \tag{2}$$

embodies information about the distinctiveness of the global minimum with respect to the three *pseudo-minima*. Actually, we consider the ratio $\frac{\Delta SAD}{SAD_{min}}$ to evaluate the distinctiveness of the global minimum, with high ratios indicating distinctive global minima.

Other papers [2,6,33] rely on a similar analysis of the matching function to detect and discard unreliable matches. Yet, the novelty of our proposal consists in the deployment of the pseudo-minima, i.e., in carrying out a parallel, SIMD-based analysis.

5. Efficient incremental computation techniques

The calculation of SAD scores is the most expensive task carried out by the stereo algorithm and several incremental calculation schemes aimed at avoiding redundant calculations have been proposed. In this section, in order to describe the mapping of the stereo algorithm on a SIMD architecture, we review a very efficient technique described in [29] which relies on a triple incremental calculation scheme.

Let $SAD(x, y, d)$ be the SAD score between a window of size $(2n + 1) \times (2n + 1)$ centered at coordinates (x, y) in the left image and the corresponding window centered at $(x + d, y)$ in the right image:

$$SAD(x, y, d) = \sum_{i,j=-n}^{n} \big|L(x + j, y + i) - R(x + d + j, y + i)\big| \tag{3}$$

One can notice that $SAD(x, y + 1, d)$ can be obtained from $SAD(x, y, d)$ as follows

$$SAD(x, y + 1, d) = SAD(x, y, d) + \Omega(x, y + 1, d) \tag{4}$$

with

$$\begin{aligned}\Omega(x, y + 1, d) = &\sum_{j=-n}^{n} \big|L(x + j, y + n + 1) - R(x + d + j, y + n + 1)\big| \\ &- \sum_{j=-n}^{n} \big|L(x + j, y - n) - R(x + d + j, y - n)\big|\end{aligned} \tag{5}$$

representing the difference between the SADs associated with the lowermost and uppermost rows of the matching window. Moreover, $\Omega(x, y + 1, d)$ can be obtained from $\Omega(x - 1, y + 1, d)$ by simply considering

$$\begin{aligned}\Omega(x, y + 1, d) = &\Omega(x - 1, y + 1, d) \\ &+ \big|L(x + n, y + n + 1) - R(x + d + n, y + n + 1)\big| \\ &- \big|L(x + n, y - n) - R(x + d + n, y - n)\big| \\ &- \big|L(x - n - 1, y + n + 1) - R(x + d - n - 1, y + n + 1)\big| \\ &+ \big|L(x - n - 1, y - n) - R(x + d - n - 1, y - n)\big|\end{aligned} \tag{6}$$

This scheme allows for rendering the number of operations independent of the size of the matching window, since only four elementary operation are needed to obtain the SAD score at each new point.
The pre-processing step requires computation of the mean and variance of the two images. If we consider the left image, and putting $N = 2n + 1$, the mean is given by

$$\mu(x,y) = \frac{1}{N^2} \sum_{i,j=-n}^{n} L(x+j, y+i) = \frac{1}{N^2} S_1(x,y) \tag{7}$$

while the variance can be expressed [10] as

$$\sigma^2(x,y) = \frac{1}{N^2} \sum_{i,j=-n}^{n} L^2(x+j, y+i) - \mu^2(x,y) = \frac{1}{N^2} S_2(x,y) - \mu^2(x,y) \tag{8}$$

It can be easily verified that the computation of mean and variance can be carried out using the following schemes:

$$S_1(x, y+1) = S_1(x,y) + \Omega_{S_1}(x, y+1) \tag{9}$$

$$\Omega_{S_1}(x, y+1) = \sum_{j=-n}^{n} \big(L(x+j, y+n+1) - L(x+j, y-n) \big) \tag{10}$$

$$\begin{aligned} \Omega_{S_1}(x, y+1) = & \Omega_{S_1}(x-1, y+1) + L(x+n, y+n+1) - L(x+n, y-n) \\ & - L(x-n-1, y+n+1) + L(x-n-1, y-n) \end{aligned} \tag{11}$$

$$S_2(x, y+1) = S_2(x,y) + \Omega_{S_2}(x, y+1) \tag{12}$$

$$\Omega_{S_2}(x, y+1) = \sum_{j=-n}^{n} \big(L^2(x+j, y+n+1) - L^2(x+j, y-n) \big) \tag{13}$$

$$\begin{aligned} \Omega_{S_2}(x, y+1) = & \Omega_{S_2}(x-1, y+1) + L^2(x+n, y+n+1) - L^2(x+n, y-n) \\ & - L^2(x-n-1, y+n+1) + L^2(x-n-1, y-n) \end{aligned} \tag{14}$$

In both the matching and pre-processing step it is possible to introduce a third level of incremental computation aimed at achieving additional speed-up. In fact, formulas (6), (11) and (14) show that the pixels at the corners of the correlation window contribute to two terms, say A and B, where A includes the two pixels at the left corners and B those at the right corners.

Because term B plays the role of term A when the correlation window is shifted horizontally by $2n+1$ units, with a very small memory cost we can store the most recent $2n+1$ B terms so that they can be re-used $2n+1$ units later in place of the A terms. If we denote the array of the B terms by T, each element can be referenced with the index $\tilde{x} = x \ mod \ (2n+1)$, and thus all elements are visited each time the correlation window is shifted horizontally by $2n+1$ units. When shifting the window by one unit, a new B term is calculated while the needed A term is fetched from $T(\tilde{x})$. After both terms have been used, $T(\tilde{x})$ is updated with the newly calculated B term.

To introduce this third level of incremental computation into the pre-processing step, formula (11) is rewritten as follows:

$$\Omega_{S_1}(x, y+1) = \Omega_{S_1}(x-1, y+1) + \big(L(x+n, y+n+1) - L(x+n, y-n)\big) - T_1(\tilde{x}) \tag{15}$$

where

$$T_1(\tilde{x}) = L(x-n-1, y+n+1) - L(x-n-1, y-n) \tag{16}$$

with $\tilde{x} = x \ mod \ (2n+1)$.

Similarly, formula (14) becomes:

$$\Omega_{S_2}(x, y+1) = \Omega_{S_2}(x-1, y+1) + \big(L^2(x+n, y+n+1) - L^2(x+n, y-n)\big) - T_2(\tilde{x}) \tag{17}$$

where

$$T_2(\tilde{x}) = \big(L^2(x-n-1, y+n+1) - L^2(x-n-1, y-n)\big) \tag{18}$$

with $\tilde{x} = x \ mod \ (2n+1)$.

Finally, in the matching step the third level of incremental computation is applied for each disparity value $d \in [0, d_{max}]$; thus, the array T grows by one dimension and formula (6) is rewritten as follows:

$$\begin{aligned}\Omega(x, y+1, d) = &\Omega(x-1, y+1, d) - T(\tilde{x}, d) + \big|L(x+n, y+n+1) - R(x+d+n, y+n+1)\big| \\ &- \big|L(x+n, y-n) - R(x+d+n, y-n)\big|\end{aligned} \tag{19}$$

where

$$T(\tilde{x},d) = \big|L(x-n-1,y+n+1) - R(x+d-n-1,y+n+1)\big| - \big|L(x-n-1,y-n) - R(x+d-n-1,y-n)\big| \quad (20)$$

with $\tilde{x} = x \ mod \ (2n+1)$ and $d \in [o, d_{max}]$.

6. Mapping to a processor with parallel SIMD instructions

Our stereo algorithm is designed to run on Intel Pentium processors providing at least the MMX [8] multimedia instruction set. Moreover, since the current generation of Pentium processors (Pentium III and Pentium 4) provide a better support for SIMD (Single Instruction Multiple Data) programming with the SSE (Streaming SIMD Extensions) [25] instruction set, we have also developed an optimised SSE mapping. Nevertheless, since the different SIMD architectures are provided in most cases with similar capabilities, the extension to other SIMD architectures is straightforward.

6.1. Pre-processing step

The *preprocessing step* takes as input the left and right frame buffers, calculates the means and obtain the normalized frames used in the matching step. In addition, the left frame variances are also evaluated. The pre-processing steps relies on the incremental calculation scheme described in Section 5 to dramatically decrease the number of operations executed at each pixel. Figure 2 shows which steps are taken to preprocess an input frame. The pixel (n, n) is processed first to compute its mean by running two nested loops visiting all the pixels in the window centered at (n, n). With this step, $2n + 1$ elements are stored in a `DeltaMean` vector, each of them containing the sum of the pixels within the light gray vertical stripes. $2n + 1$ elements are also stored in a `DeltaVariance` vector for the sum of squared values of the same pixels.

The next step consists in moving to the remaining pixels of the first row and applying horizontal recursion. For a given pixel $(x, n), x > n$, the computation of the mean needs the sum of values of all pixels in the window centered at (x, n). Those values can be obtained from the same sum computed for the previous pixel $(x-1, n)$, with a correction given by the two vertical strips shown in Figure 2 on the left and right sides of the window centered at (x, n). The sum of values corresponding to the strip on the left is found in the `DeltaMean` vector in position $j = (x-n)mod(2n+1)$. For each pixel (x, n) one loop only is then required, to compute the sum of values of the pixels in the right strip. An identical process making use of the `DeltaVariance` vector is carried out to compute the variance (sum of squared values) for the pixels $(x, n), x > n$ in the left image.

A generic row in a frame is processed by making a strong use of SIMD instructions. This is possible by first initializing some data structures while processing the first pixel

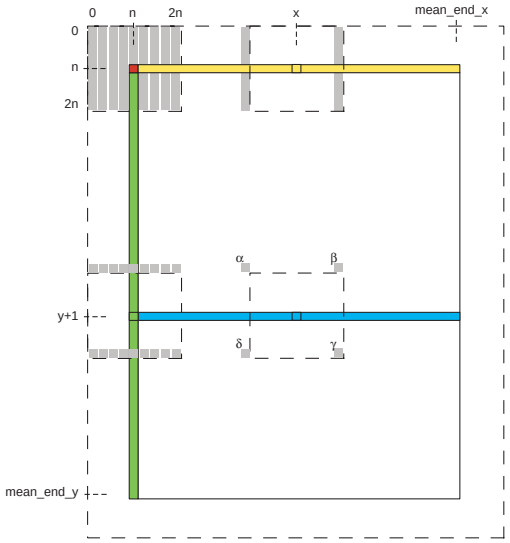

Fig. 2. Pixels involved in the preprocessing step.

$(n, y+1)$ of the current row. The mean for the pixel $(n, y+1)$ is given by the sum of values referred to the pixels in the window centered on (n, y) and the corrections represented by the horizontal strips on the top and bottom of the window centered at $(n, y+1)$. Variance calculation differs from the above simply because it is based on the squared values of the same pixels.

When visiting both horizontal strips, the `DeltaMean` and `DeltaVariance` arrays are initialized so that each element contains, respectively, the difference of values and the difference of squared values between a pixel in the bottom strip and the corresponding pixel in the top strip. This is done to support the third incremental computation described in (15) and (17). SIMD programming brings optimal results when applied to the final stage of the preprocessing step because it allows for parallel mean and variance computations.

The first (*vertical*) incremental step consists in retrieving the sums of values and sums of squared values corresponding to the pixels $(x+i, y)$, $i \in [0..7]$ located in the previous row and correcting those quantities with the contributions $\Omega_{s1}(x+i, y+1)$ and $\Omega_{s2}(x+i, y+1)$, $i \in [0..7]$ in formulas (9) and (12).

The second (*horizontal*) incremental step computes the terms $\Omega_{s1}(x+i, y+1)$ and $\Omega_{s2}(x+i, y+1)$, $i \in [0..7]$, as a progressive correction of the same terms corresponding to the previous pixel. Formulas (11) and (14) show that the corrections are given by the pixels located at the corners of the window centered on the current pixel. With

reference to one pixel out of the eight processed in parallel, Figure 2 shows that Ω_{s1} is given by $(\gamma - \beta) - (\delta - \alpha)$, while Ω_{s2} – by the term $(\gamma^2 - \beta^2) - (\delta^2 - \alpha^2)$. The third and final incremental step is aimed at computing and storing terms $(\gamma - \beta)$ and $(\gamma^2 - \beta^2)$ so that they can be re-used as the terms $(\delta - \alpha)$ and $(\delta^2 - \alpha^2)$ when processing the pixels $(x + (2n + 1) + i, y + 1)$, $i \in [0..7]$.

The described process is slightly complicated by the need to store the terms $(\gamma - \beta)$ in words and the terms $(\gamma^2 - \beta^2)$ in double words. Using words and double words rather than bytes reduces the number of parallel operations from 8 to 4 and from 8 and 2, respectively. The following MMX code shows how we compute the terms $(\gamma - \beta)$. The eight γ values are read from the bottom-right corner of the correlation window and cast to words with the MMX instructions `punpcklbw` and `punpckhbw`, and the same is done for the eight β values from the top-right corner. A pair of `psubsw` (packet subtract words with saturation) MMX instructions are finally used to compute in parallel the four plus four $(\gamma - \beta)$ differences.

```
pxor      mm7, mm7          ; mm7=0
                            ; DnR: accessing 8 gamma values
movq      mm6, [esi+ecx]    ; mm6 = pixel 0-7 from corner DnR (BYTES)
movq      mm5, mm6          ; mm5 = mm6 copy
punpcklbw mm6, mm7          ; mm6 = pixel 0-3 from corner DnR (WORDS)
punpckhbw mm5, mm7          ; mm5 = pixel 4-7 from corner DnR (WORDS)
                            ; UpR: accessing 8 beta values
movq      mm4, [esi]        ; mm4 = pixel 0-7 from corner UpR (BYTES)
movq      mm3, mm4          ; mm3 = mm4
punpcklbw mm4, mm7          ; mm4 = pixel 0-3 from corner UpR (WORDS)
punpckhbw mm3, mm7          ; mm3 = pixel 4-7 from corner UpR (WORDS)
                            ; (gamma - beta) terms
psubsw    mm6, mm4          ; mm6 = (0-3 DnR) - (0-3 UpR)
psubsw    mm5, mm3          ; mm5 = (4-7 DnR) - (4-7 UpR)
```

6.2. Matching step

The *matching step* accepts as input the normalised frames computed in the preprocessing step and generates the disparity map for the pair of original frames. The complexity of the matching step is d_r (with $d_r = d_{max} + 1$) times higher than the complexity of the preprocessing step. Thus, the code optimisation carried out within the matching step have a strong impact on the overall execution time. The matching step is implemented according to the incremental computation scheme described in Section 5. Figure 3 shows which steps are taken to process the normalised left and right frames in order to establish matches between pixels in the left frame and pixels in the right frame.

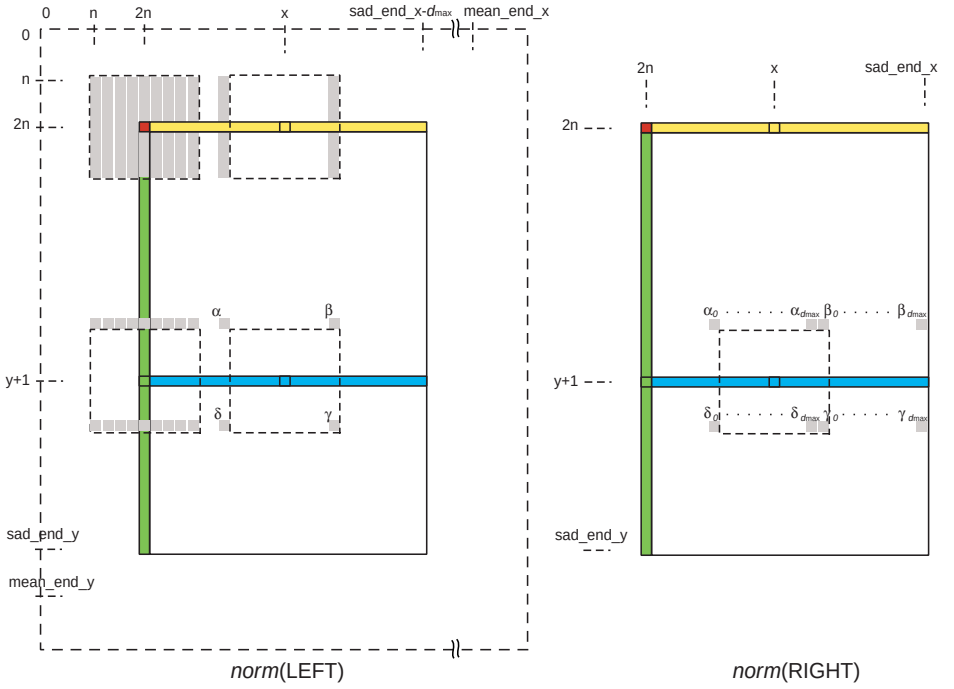

Fig. 3. Pixels involved in the matching step.

The matching process consists in visiting each pixel (x, y) in the normalised left frame to evaluate the d_r terms $SAD(x, y, d)$, $d \in [0..d_{max}]$ indicating the error scores between (x, y) and the d_r pixels $(x+d, y)$, $d \in [0..d_{max}]$ located within the normalised right frame. The best match found for the current pixel (x, y) is identified by the smallest score:

$\texttt{SAD\_MIN} = SAD(x, y, \texttt{disparity})$

The first processed pixel is located in the left normalized frame at coordinates $(2n, 2n)$. For each disparity value, two nested loops visit all pixels within the correlation window centered at $(2n, 2n)$, computing one term: $SAD(2n, 2n, d)$, as indicated by formula (3). Each term is compared with the smaller of the previously computed so that, after considering all d_r disparity values, the best match for the pixel $(2n, 2n)$ is found. Similarly to the pre-processing step, the partial sums of absolute values associated with $2n + 1$ vertical strips shown in Figure 3 must be stored to sustain the incremental computation required for the remaining pixels of the first row. The main difference between the two steps is that the matching step requires storing $(2n + 1) \cdot d_r$ terms instead of just $2n + 1$.

For a generic pixel $(x, 2n)$ in the first row of the left normalised frame, the d_r terms $SAD(x, 2n, d)$ are computed by adding to the previously stored terms $SAD(x-1, 2n, d)$, the corrections given by the difference between the sum of absolute values associated with the vertical strip on the right of the correlation window and its analogue associated with the vertical left strip. The most interesting portion of code in our stereo algorithm is the matching process for a generic row, since it is heavily based on SIMD techniques. Following the same scheme adopted in the pre-processing step, the generic row is processed after initializing some data structures when processing the first pixel of the generic row. The d_r terms $SAD(2n, y+1, d)$ with $d \in [0..d_{max}]$ are computed as corrections of the terms $SAD(2n, y, d)$ with $d \in [0..d_{max}]$ stored when processing the previous row, based on the following relation:

$$SAD(2n, y+1, d) = SAD(2n, y, d) + \sum_{j=2n-n}^{2n+n} T(j, d) \qquad d \in [0..d_{max}] \tag{21}$$

where

$$T(j, d) = \big|L(j, y+n+1) - R(j+d, y+n+1)\big| - \big|L(j, y-n) - R(j+d, y-n)\big| \tag{22}$$

Figure 4 shows the case of $d = 6$. The correction is given by summing up the contribution of the $2n+1$ terms $T(j, 6)$, each of them being the difference of two absolute values. The first absolute value refers to the pixel j in the bottom horizontal strip in the left image and the pixel $j+6$ in the corresponding strip in the right image. The second absolute value refers to pixels in same positions but located in the top horizontal strips.

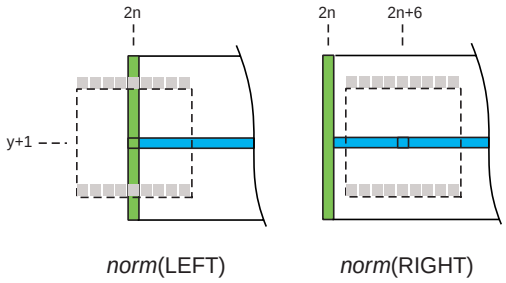

Fig. 4. Matching for the first pixel of a generic row.

A generic row at coordinate $y+1$ is processed in the matching step with a main loop iterating, one by one, each pixel of the row. In order to compute the first absolute value appearing in formula (19), we load a multimedia register with eight copies of the same left pixel $L(x+n, y+n+1)$ and, in parallel, read the eight pixels $R(x+d+n, y+n+1)$,

with $d \in [0..7]$, at the bottom of the correlation window. The following code relies on *arithmetic with saturation*, and shows how the eight absolute values are computed.

```
                        ; mm1 contains 8 copies of the
                        ; DnR pixel in Left frame
movq    mm2, [esi+ebx]  ; Read 8 DnR pixels from Right frame
movq    mm3, mm1        ; in mm3 8 copies of DnR pixel from Left
movq    mm4, mm1        ; mm4 = 8 copies of DnR pixel from Left
                        ; DnR absolute difference
psubusb mm3, mm2        ; Left(DnR)-Right(DnR), negative terms to 0
psubusb mm2, mm4        ; Right(DnR)-Left(DnR), negative terms to 0
por     mm3, mm2        ; merge non zero terms (abs values) in mm3
```

The first `psubusb` (*subtract unsigned packed bytes with saturation*) instruction stores in `mm3` the positive differences between the eight copies of the pixel in the left image and the eight pixels read from the right image, while the negative differences are converted to zero due to *arithmetic with saturation*. The second `psubusb` instruction replaces the factors of the packed difference by fact, saturating the previously positive terms and storing as positive those previously saturated. The final `por` (packed or) instruction merges in `mm3` the positive results of the two computed differences, providing eight absolute values. Formula (19) is computed for the current pixel a total of d_r times, which involves evaluating d_r differences of absolute values and accessing d_r terms $T(\tilde{x}, d)$. Each time the mentioned formula is computed, the difference of absolute values is also used to replace the corresponding term $T(\tilde{x}, d)$ by fact, completing the inner level of incremental computation.

6.3. SIMD search for the best match

The d_r terms $SAD(x, y+1, d)$ computed for the current pixel $(x, y+1)$ represent d_r error scores between the current pixel and $(x+d, y+1)$, with $d \in [0..d_{max}]$ pixels in the right image. The following MMX/SSE code shows how the algorithm searches for the minimum SAD term within the d_r available ones.

```
movq mm0, [edx+ecx*8]       ; mm0: current 4 SAD min (WORDS)
movq mm1, HiDisparities     ; mm1: current 4 indexes
                            ; dmax,dmax-1,dmax-2,dmax-3

movq mm6, Mask04040404      ; mm6: mask to decrease indexes by 4
movq [edi], mm1             ; [edi] holds indexes 4 SAD min

loopMin:
movq     mm2, [edx+ecx*8-8] ; mm2: next 4 SAD values (WORDS)
```

```
psubw    mm1, mm6          ; mm1: next 4 indexes
pminsw   mm0, mm2          ; mm0: updates current 4 SAD min
pcmpeqw  mm2, mm0          ; mask: FFFF=min changed, 0=no change
maskmovq mm1, mm2          ; update [edi] indexes where mm2=FFFF

loop loopMin               ; jumps decreasing ecx by 1
movq mm1, [edi]            ; mm1: disparities 4 SAD (pseudo)min
```

The search begins by initializing the `mm0` register with the four terms $SAD(x, y+1, d)$, $d \in [d_{max}-3..d_{max}]$ computed last and the `mm1` register with the disparity values $d_{max}-3$, $d_{max}-2$, $d_{max}-1$ and d_{max} of the SAD terms in `mm0`. The memory locations pointed by the `edi` register are used to store the disparities of the smaller SAD terms currently found. The loop begins loading `mm2` register with the four SAD terms immediately preceding the previously accessed SADs and `mm1` register with the respective disparities.

The searching process is based on the SSE instruction `pminsw` (minimum of packed signed words), which updates the current four minimum SAD terms in `mm0` by considering the values in `mm2`. The MMX instruction `pcmpeqw` (compare packed words for equality) is then used to build a bit mask identifying which term in `mm0` is changed. The mask is used with the SSE instruction `maskmovq` (store selected bytes of quadword) to selectively move in `[edi]` the disparities of the SADs terms recently moved in `mm0` by the `pminsw` instruction. At the end of the loop, `mm0` contains four SAD terms, one of which is the smallest of the d_r available, while the other three the so-called *pseudo minima*. The name *pseudo minima* comes from the consideration that the parallel search method does not ensure that the three terms are the smallest SADs greater than the minimum. They only have good chances for this because of the continuity of the error function. The portion of code following the search method is used to isolate the term `SAD_MIN` as the smallest SAD out of the four terms in `mm0` and retrieve the corresponding `disparity`. At the same time, we carry out the *sharpness* (Eq. (1)) and *distinctiveness* (Eq. (2)) tests, using the smallest SAD and the three pseudo minima.

7. Constraints check

From the previous computation we have collected all the information needed to evaluate the reliability of the best match identified by `SAD_MIN` and `disparity`. The first reliability test checks if the reference pixel in the left image is located in a low-textured region. If the variance computed for that pixel at the preprocessing step is less than a given threshold, the region is considered poorly textured and the match is discarded. In the opposite case, we consider two possibilities: a) the match involves a pixel in the right image never matched before, and b) the match involves a pixel in the right image already matched in a previous iteration. Case a) the sharpness test, which requires evaluation of Eq. (1), with d_{min} = `disparity` and checking if Δd is smaller than a given threshold

(the lowest possible value for Δd is 4 while our threshold is commonly set to 7). If the test is passed, all the pseudo-minima fall relatively close to each other and the match is accepted because it is considered reliable. If the test fails, the distinctiveness test follows, which requires calculation of Eq. (2) with SAD_{min} = `SAD_MIN` and checking if the ratio $\frac{\Delta SAD}{SAD_{min}}$ is greater than a given threshold (our threshold is commonly between 1/8 for a strong reliability and 1/32 for a weaker reliability). If the test succeeds, the 3 pseudo-minima have SADs considerably higher than `SAD_MIN` and the match is accepted while, if the test fails, the match is discarded. Case b) ensures satisfaction of the uniqueness constraint, testing if the `SAD_MIN` value computed for the current match is greater than the `SAD_MIN` obtained in the previous match. If the test succeeds, the current match is discarded, while if it fails, the previous match is discarded and the current match is tested as described in case a).

8. Experimental results and applications

In this section we discuss the experimental results obtained by our stereo vision system on a real stereo sequence, referred to as "*Count*", which was acquired in our laboratory with a monochrome Mega-D Stereo Head from Videre Design [44]. The stereo camera had a fixed baseline of about 9 cm and was equipped with a pair of 4.8 mm lenses.

As shown by the three rectified frames 0001, 0116 and 0189 of the "*Count*" sequence (leftmost images in Figures 5, 6, 7), the video captures three people moving around a camera, placed at about 2.5 meters over the floor. The sequence presents several challenging situations, such as occlusions occurring between people and/or objects, uniform regions (e.g., the desk and the furniture in the background, the wall in the foreground), specular regions (e.g., portions of the floor in the middle of the scene) and repetitive patterns generated by the tiles on the floor.

The rightmost images in Figures 5, 6, 7 show the disparity maps generated by the SMP algorithm processing the three rectified stereo pairs. The parameters of the algorithm were: disparity range of 32 pixels, threshold on variance intensity $\sigma = 2$ and a correlation window of 15×15 pixels. Moreover, the disparity maps were interpolated using the subpixel factor of 1/16. The disparity maps are encoded with grayscale values: brighter levels represent points closer to the camera while unmatched points are represented in white.

The disparity maps in Figures 5, 6, 7 show that the rough 3D structure of the scene has been correctly recovered (e.g., the people appearing in the scene and most of the background). Nevertheless, the disparity maps contain some wrong measurements (e.g., a portion of the wall behind the desk, some points in the floor). However, the constraints embodied in the SMP algorithm and the variance intensity threshold $\sigma = 2$ used for detecting untextured regions allow for discarding several ambiguous regions, such as portions of the wall, of the furniture and of the floor.

Moreover, it is worth observing that the disparity maps shown in the figures are affected by the *border-localisation* problem that causes inaccurate fitting of the object's border. Althought some authors (e.g [33,34]) proposed techniques aimed at reducing the *border-localisation* problem, it is worth pointing out that the results provided by these algorithms are still less accurate than those generated with slower *global* algorithms (e.g. [28]).

The SMP algorithm turns out to be quite fast. In fact, on a Pentium III processor at 800 MHz with a disparity range of 64 pixels it delivers disparity maps at 25.94 fps with stereo pairs of 320 × 240 pixels, and 1.86 fps with stereo pairs of 1024 × 768 pixels.

The PC based stereo vision system is currently used in a *3D people counting* application that is under development in our laboratory. The application is aimed at tracking and counting people/objects in the field of view of the stereo camera in real-time using a standard PC. The current approach uses a virtual *plan-view* [42] of the scene and relies on the information provided by the SMP algorithm combined with the information provided by a change detection algorithm [35].

Figures 8, 9 and 10 show the previous three frames of the "*Count*" sequence and the statistics concerned with people moving from region A to region B (red color) and people moving from region B to region A (green color). In the first shot of the sequence, shown in Figure 8, the green line on the floor shows the 3D position of the *virtual gate* between the two regions.

It is worth observing that the system can handle difficult situations, like the severe occlusion occurring between the two people at the center of the scene shown in Figure 10. The entire sequence, as well as experimental results concerning other stereo sequences acquired in our laboratory are available at [41].

Recently a comparative study of fast dense stereo vision algorithms has been proposed [48]. The authors implemented algorithms based on BM and on the SMP approach: for both algorithms, they also implemented the multiple window approach proposed in [33]. Other algorithms examined in the paper are the OpenCV [46] implementation of the Birchfield and Tomasi algorithm [19], and three algorithms available at [47] and described in [31]: *winner takes all* based on SSD (referred to SSD), *dynamic programming* (DP) and *scanline optimization* (SO). The eight examined algorithms have been tested (see [48] for details and the metric used) on synthetic data generated using a 3D-visualisation module as well as on the same data set corrupted with *white gaussian noise*. Althought it seems that the authors implemented only the basic SMP matching engine described in section 4 (i.e., without the match reliability tests), the mean error of SMP is in the worst case about 5% greater than those of BM. Moreover, on the noisy data set the mean error of the basic SMP algorithm is smaller than those of DP, SO and SSD. Finally, it is worth pointing out that the SMP approach has always turned out to be the fastest algorithm among those examined, and that the speed-up increased with larger image size and disparity range.

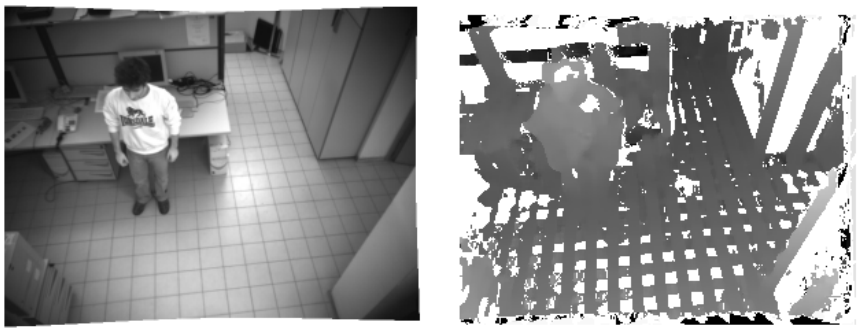

Fig. 5. First frame of the "Count" stereo sequence. (Left) Rectified left image. (Right) Disparity map obtained by the SMP algorithm.

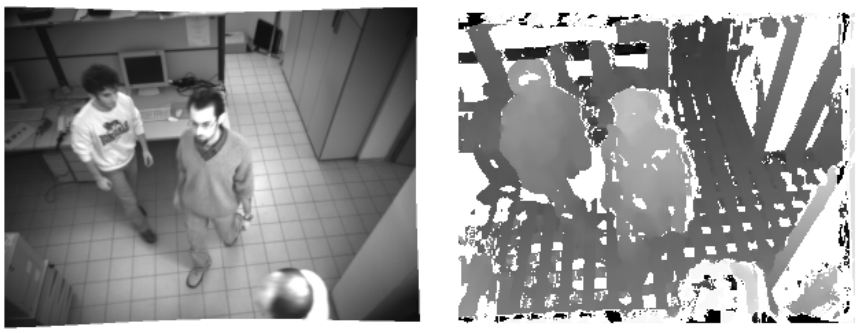

Fig. 6. Frame 0116 of the "Count" stereo sequence. (Left) Rectified left image. (Right) Disparity map obtained by the SMP algorithm.

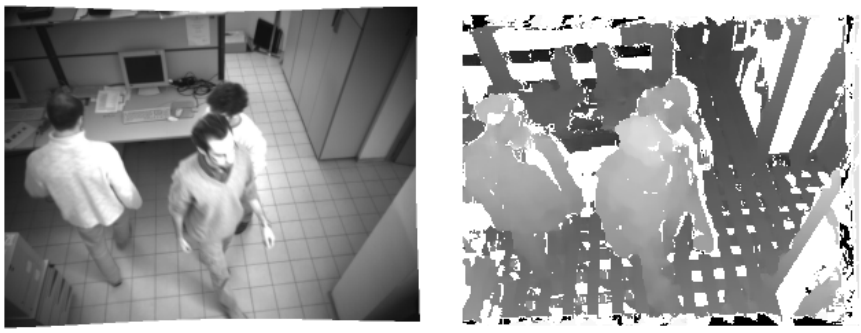

Fig. 7. Frame 0189 of the "Count" stereo sequence. (Left) Rectified left image. (Right) Disparity map obtained by the SMP algorithm.

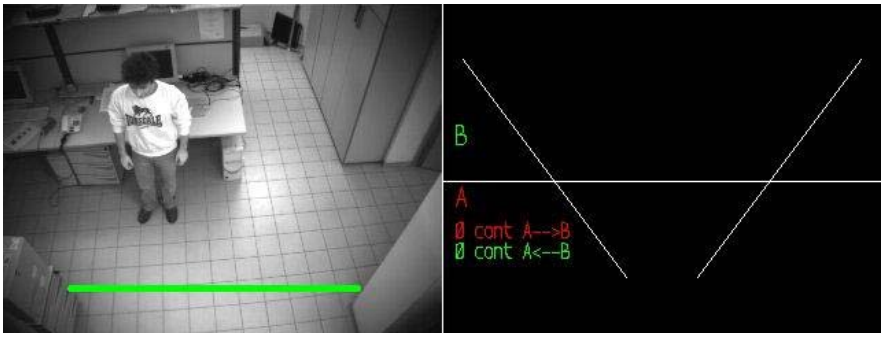

Fig. 8. First frame of the "Count" stereo sequence. (Left) Original left image: the green line on the floor shows the 3D position of the *virtual gate* between regions A and B. (Right) plan-view map and statistics about the crossings in the two directions.

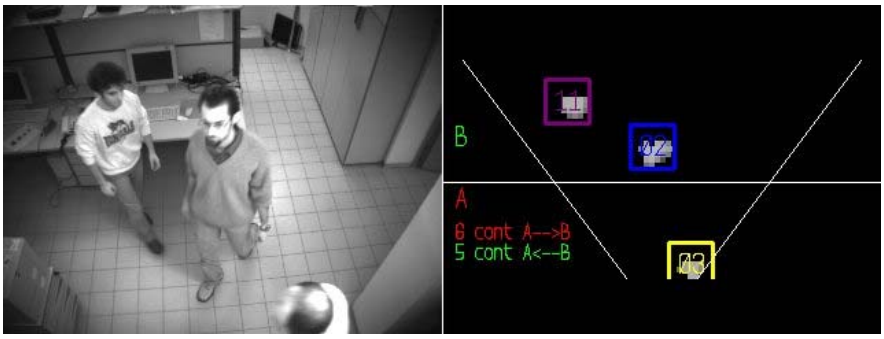

Fig. 9. Frame 0116 of the "Count" stereo sequence. (Left) Original left image. (Right) plan-view map with the tracked people and statistics about the crossings in the two directions.

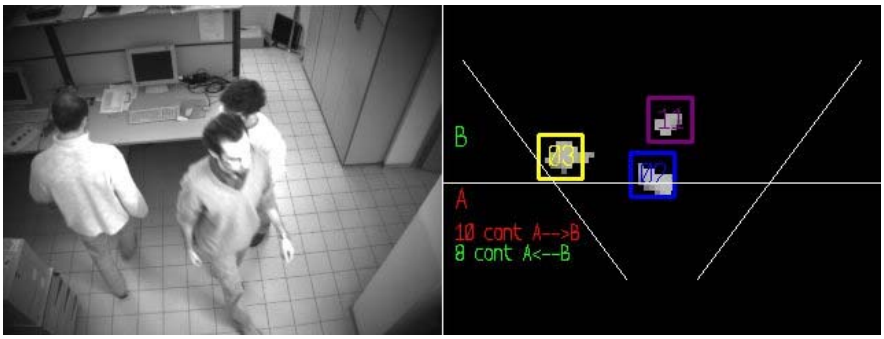

Fig. 10. Frame 0189 of the "Count" stereo sequence. (Left) Original left image. (Right) plan-view map with the tracked people and statistics about the crossings in the two directions.

9. Conclusion

We have described a PC based stereo vision system suitable for real-time applications. The system relies on an efficient local stereo matching algorithm based on a single matching phase (SMP). The matching algorithm discards unreliable matches by detecting violations of the uniqueness constraint. Further reliability improvement is achieved, at a low computational cost thanks to the use of parallel SIMD programming, by constraining the behavior of the error scores. The SMP algorithm exploits efficient recursive calculation schemes, which allows for avoiding redundant computations. A further speed up was achieved by parallelising the most computationally expensive portion of the algorithm on a parallel architecture. Thus, we have provided a detailed description of the parallel mapping of SMP onto a mainstream general purpose processors with SIMD capabilities. Experimental results on real stereo sequences as well as those obtained by the outlined *3D people counting* application, show the effectiveness of the proposed real-time PC based stereo vision system.

References

1991
[1] Fua P. : Combining stereo and monocular information to compute dense depth maps that preserve depth discontinuities. 12th. Int. Joint Conf. on Artificial Intelligence, Sydney, Australia, 1292–1298.
1993
[2] Faugeras O., Hotz B., Mathieu H., Viville T., Zhang Z., Fua P., Thron E., Moll L., Berry G. Vuillemin J., Bertin P.and Proy C. : Real-time correlation-based stereo: Algorithm, Implementation and Applications. INRIA Technical Report n. 2013.
1994
[3] Zabih R., Woodfill J. : Non-parametric local transforms for computing visual correspondence. European Conf. on Computer Vision, Stockholm, Sweden, 151–158.
[4] Fleet D.J. : Disparity from local weighted phase-correlation. Int. Conf. on Systems, Man, and Cybernetics, San Antonio, USA, Vol. 1, 48–54.
1995
[5] Kanade T., Kato H., Kimura S., Yoshida A., Oda K. : Development of a video-rate stereo machine. Int. Robotics and Systems Conf., (3), 95–100
[6] RObert L., Buffa M., Hebert M. : Weakly-calibrated stereo perception for rover navigation. Fifth Int. Conf. on Computer Vision.
1996
[7] Lee R. B. : Subword parallelism with MAX-2. IEEE Micro, 16(4), 51–59.
[8] Peleg A., Weiser U. : MMX technology extension to the intel architecture. IEEE Micro, 16(4), 42–50.
[9] Tremblay M., O'Connor M., Narayanan V., He L. : VIS speeds new media processing. IEEE Micro, 16(4), 10–20.
1997
[10] Sun C. : A fast stereo matching method. Digital Image Computing: Techniques and Applications, Auckland, New Zealand, 95–97.
[11] Konolige K. : Small vision systems: hardware and implementation. 8th Int. Symp. on Robotics Research, Hayama, Japan, 111–116.
[12] Ruby B. Lee : Multimedia extensions for general-purpose processors. IEEE Workshop on Signal Processing Systems Design and Implementation, Leicester, United Kingdom, 9–23.
[13] Woodfill J., Von Herzen B. : Real-Time Stereo Vision on the PARTS Reconfigurable Computer. IEEE Symp. on FPGAs for Custom Computing Machines , Napa Valley, USA, 201–209.

1998
[14] Trucco E., Verri A. : Introductory Techniques for 3-D Computer Vision. Prentice Hall.
[15] Gluckman J., Nayar S.K., Thorek K.J. : Real-Time Omnidirectional and Panoramic Stereo. DARPA Image Understanding Workshop, Monterey, USA, 299–303

1999
[16] Corke P., Dunn P., Banks J. : Frame-rate stereopsis using non-parametric transforms and programmable logic, IEEE Int. Conf. on Robotics and Automation, Detroit, USA, 1999, 1928–1933
[17] Kimura S., Shinbo T., Yamaguchi H., Kawamura E., Naka K. A Convolver-Based Real-Time Stereo Machine (SAZAN). Conf. on Computer Vision and Pattern Recognition, 457–463.
[18] Oberman S., Favor G., Weber F. : AMD 3DNow ! Technology: Architecture and Implementations. IEEE Micro, **19(2)**, 1999, 37–48
[19] Birchfield S., Tomasi C. Depth Discontinuities by Pixel-to-Pixel Stereo, Int. Journal of Computer Vision, **35(3)**, 1999, 269–293

2000
[20] Diefendorff K., Dubey P.K., Hochsprung R., Scale H. VIS Speeds New Media Processing, IEEE Micro, **20(2)**, 2000, 85–95
[21] Egnal G. and Wildes R. Detecting binocular half-occlusions: empirical comparisons of four approaches, Int. Conf. on Computer Vision and Pattern Recognition, 2000, (2), 466–473
[22] Fusiello A., Roberto V. and Trucco E. Symmetric Stereo with Multiple Windowing, Int. Journal of Pattern Recognition and Artificial Intelligence, **14(8)**, 2000, 1053–1066
[23] Fusiello A., Trucco E. and Verri A. A compact algorithm for rectification of stereo pairs, Machine Vision and Applications, **12(1)**, 2000, 16–22
[24] Sharangpani H., Arora K. Itanium Processor Microarchitecture, IEEE Micro, **20(5)**, 2000, 24–43
[25] Shreekant T. and Huff T. Implementing streaming SIMD extensions on the Pentium III processor, IEEE Micro, **20(4)**, 2000, 47–57
[26] Van der Val G., Hansen M. and Piacentino M. The ACADIA Vision Processor, 5th Int. Workshop on Computer Architecture for Machine Perception, Padova, Italy, 2001, 31–40
[27] Tanahashi H., Yamamoto K., Caihua Wang, Niwa Y. Development of a stereo omnidirectional imaging system (SOS), IEEE Int. COnference on Industrial Electronics, Control and Instrumentation, Nagoya, Japan, 289–294

2001
[28] Kolmogorov V. and Zabih R. Computing Visual Correspondence with Occlusions using Graph Cuts, Int. Conf. on Computer Vision, 2001, 508–515

2002
[29] Di Stefano L., Marchionni M., Mattoccia S. and Neri G.A Fast Area-Based Stereo Matching Algorithm, 15th IAPR/CIPPRS Int. Conf. on Vision Interface, Calgary, Canada, 2002
[30] Di Stefano L. and Mattoccia S. Real-Time Stereo within the VIDET Project, Real Time Imaging, **8(5)**, 2002, 439–453
[31] Scharstein D. and Szeliski R. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms,, Int. Journal of Computer Vision, **47(1-3)**, 2002, 7–42
[32] Muhlmann K., Maier D., Hesser J. and Manner R. Calculating Dense Disparity Maps from Color Stereo Images, an Efficient Implementation, Int. Journal of Computer Vision, **47(1-3)**, 2002, 79–88
[33] Hirschmuller H., Innocent P. and Garibaldi J. Real-Time Correlation-Based Stereo Vision with Reduced Border Errors, Int. Journal of Computer Vision, **47(1-3)**, 2002, 229–246
[34] Okutomi M., Katayama Y. and Oka S. A Simple Stereo Algorithm to Recover Precise Object Boundaries and Smooth Surfaces, Int. Journal of Computer Vision, **47(1-3)**, 2002, 261–273

2003
[35] Di Stefano L., Mattoccia S. and Mola M. A Change Detection Algorithm based on Structure and Colour, IEEE Int. Conf. on Advanced Video and Signal Based Surveillance , Miami, USA, 252–259.
[36] Jia Y., Xu Y., Liu W., Yang C., Zhu Y., Zhang X., An L. : A Miniature Stereo Vision Machine for Real-Time Dense Depth Mapping. 3th Int. Conf. Computer Vision Systems, Graz, Austria, 268–277
[37] Darabiha A., Rose J., Maclean J.W. Video-rate stereo depth measurement on programmable hardware. Int. Conf. on Computer Vision and Pattern Recognition, Madison, Wisconsin, Vol. 1, 203–210.

2004
[38] Jia Y., Xu Y., Liu W., Yang C., Zhu Y., Zhang X., An L. : A Miniature Stereo Vision Machine (MSVM-III) for Dense Disparity Mapping, 17th Int. Conf. on Pattern Recognition, Cambridge, UK, Vol. 1, 728–731.
[39] Advanced RISC Machines, www.arm.com
[40] Bouguet J.Y.Camera Calibration Toolbox, www.vision.caltech.edu/bouguetj/calib_doc/
[41] Di Stefano L., Marchionni M., Mattoccia S. Experimental results, www.vision.deis.unibo.it/~smattoccia/stereo.htm
[42] Harville M.Stereo person tracking with adaptive plan-view templates of height and occupancy statistics, Image and Vision Computing, 22(2) , 127–142.
[43] Point Grey Research, www.ptgrey.com
[44] Videre Design, www.videredesign.com
[45] Tyzx, www.tyzx.com
[46] Open Source Computer Vision Library, http://www.intel.com/research/mrl/research/opencv/
[47] Middlebury Stereo Vision Page www.middlebury.edu/stereo
[48] Sunyoto H., van der Mark W., Gavrila D. M. A Comparative Study of Fast Dense Stereo Vision Algorithms, IEEE Intelligent Vehicles Symposium, Parma, Italy, 252–259

Luigi Di Stefano graduated from the University of Bologna as an Engineer of Electronics in 1989. In 1990 he joined the Department of Electronics, Computer Science and Systems (DEIS) of the University of Bologna, where he received a Ph.D in Electronic Engineering and Computer Science in 1994. In 1995, he was granted a post-doc fellowship at the Trinity College, Dublin. He is currently an Associate Professor at the Faculty of Engineering of the University of Bologna, where he teaches courses on Computer Architecture and Computer Vision. His research activity has been focused on image processing and computer vision algorithms as well as on parallel/dedicated computing architectures for the processing of image and video data. In 2001 he also joined ARCES (Advanced Research Centre on Electronic Systems "Ercole De Castro"). He is a member of the IEEE Computer Society and of the Italian Chapter of the IAPR (International Association for Pattern Recognition).

Massimiliano Marchionni graduated from the University of Bologna in 2001 as Engineer of Electronics. Skilled in code optimization and software design, he had worked for his thesis at the Department of Electronics, Computer Science and Systems (DEIS), developing Computer Vision algorithms for real-time applications. Being in contact with doctor S. Mattoccia and professor L. Di Stefano over past years he has collaborated with them in writing papers on Computer Vision. Currently he is working at Arteco spa as a researcher focused on motion detection, motion tracking and camera control.

Stefano Mattoccia received a Master Degree in Electronic Engineering and a PhD in Electronic Engineering and Computer Science from the University of Bologna in 1997 and 2002, respectively. At the University of Bologna he joined the Department of Electronics Computer Science and Systems (DEIS), Advanced Research Center on Electronic Systems "Ercole De Castro" (ARCES). Currently, he is a Research Associate at the Faculty of Engineering, where he teaches Logic Design. His main research interests are computer vision, real-time 3D vision and sensors. He is a member of the Italian Chapter of the International Association for Pattern Recognition (IAPR).