The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Incremental Disparity Space Image computation for automotive applications

Mirko Felisa and Paolo Zani
VisLab – Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma, ITALY
http://www.vislab.it
{felisa,zani}@vislab.it

***Abstract*— Generating a depth map from a pair of stereo images is a challenging task, which is often further complicated by the additional restrictions imposed by the target application; in the automotive field, for example, real-time environment reconstruction is essential for safety and autonomous navigation systems, thus requiring reduced processing times, often at the expense of a somewhat limited degree of accuracy in the results. Nevertheless, a-priori knowledge on the intended use of the algorithm can also be exploited to improve its performance, both in terms of precision and computational burden.**

This paper presents three different approaches to incremental Disparity Space Image (DSI) computation, which leverage the properties of a stereo-vision system installed on a vehicle to produce accurate depth maps at sustained frame rates on commodity hardware.

I. Introduction

Soft real-time processing capabilities are mandatory in order to successfully interact with the highly dynamic environment typical of automotive applications; when producing depth maps by means of stereo-vision this constraint can be met using a local, correlation-based approach. While said category includes several algorithms, this study focuses on an incremental computation technique based on the SAD (Sum of Absolute Differences) metric: the underlying idea, and the related performance benefits, have been first investigated in [1] and [2] respectively; at the implementation level, further optimizations are possible, which exploit information about the hardware, like the inherent parallelism of modern multi-core, SIMD[1]-capable CPUs.

While driving in both urban and unstructured scenarios using a stereo imaging system like the one depicted in Fig. 1, the ground in front of the vehicle occupies a substantial portion of the images, and as such can be modeled, for example using a V-Disparity Image-based approach [3], [4]. In this situation, the objects that need to be detected are those standing out of the ground, or –if negative obstacles need to be handled too– delve into it to some limited extent. This observation, together with the fact that in automotive applications the vehicle roll can be neglected most of times, allows to limit the search range used for window matching to disparity values greater than those of the ground for any given image row; the actual search area can nevertheless be expanded to include some values that would result below the terrain surface, both to increase the robustness against the noise present in the ground estimation process, and to cope with the aforementioned negative obstacles, if it is the case.

[1]Single Instruction Multiple Data

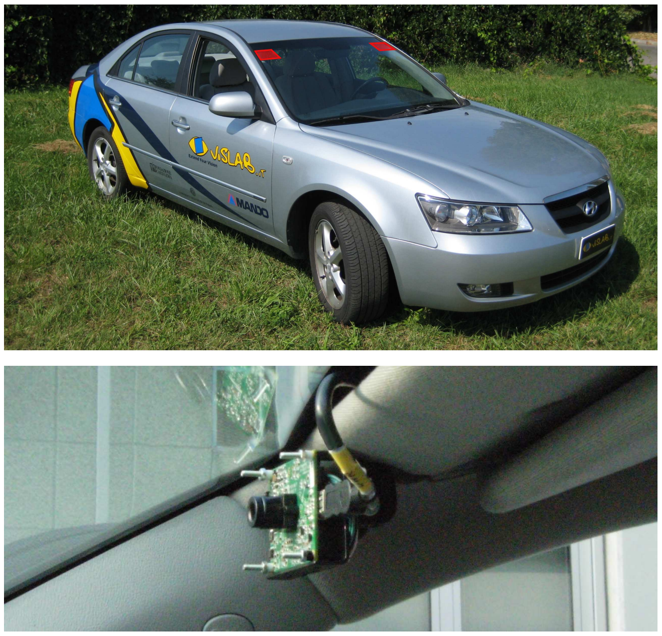


Fig. 1. A test vehicle: (a) highlighted in red, the stereo-vision system mounted on top of the windshield, and (b) a close-up of the right camera

The benefits of reducing the search space are a speedup of the DSI generation process, since fewer matches are involved, but also an increase in the quality of the resulting image, since spurious matches corresponding to unacceptable disparity values are automatically avoided. Fig. 2 compares a map obtained taking into account the ground position with one computed using a fixed range.

Although the combination of an incremental approach with limited search ranges is appealing, it comes at the cost of a processing flow encumbered by non-trivial border cases, which need to be addressed in order to obtain correct results; moreover, accurate camera calibration needs to be performed, and intervening miscalibrations (for example due to vibrations or hits) need to be detected and compensated: while this is certainly an issue to consider, recent imaging systems are becoming increasingly compact and lightweight, thus resulting less prone to mechanical stress.

II. Algorithms

The most direct approach to the application of limited search ranges to an incremental algorithm, and the first one to be developed, is presented in Sec. II-A, and derives from [1] and [2]; its main constraint is its inefficiency when handling sparse inputs (e.g. Sobel-filtered and thresholded images), since every output value depends on its neighbors: this requirement forces the computation of all the DSI points, even those for which it is known in advance that they are going to be discarded. To overcome this limitation the algorithm described in Sec. II-B has been devised: inter-pixel dependency is relaxed so that each one no longer requires the computation of the preceding points on the same row; this allows to skip unnecessary DSI values, at the cost of no longer having a processing time independent of correlation window width; moreover, it becomes possible to use variable search ranges within a given row. Aside being a trade-off which might or might not be beneficial, depending on the intended application, this approach results interesting in that it can be exploited to produce an alternate formulation of the fully incremental problem, which uses just one pixel comparison per tested disparity during the matching phase, regardless of window size, as detailed in Sec. II-C.

A. Incremental approach

Given two input stereo images (left and right, $\mathbf{L}$ and $\mathbf{R}$ respectively in the following) of size $width \times height$, a DSI can be computed by iterating on each row and each column of the base one ($\mathbf{R}$), matching a small region cropped around each point against the set of candidate windows on the related epipolar line of the inspection image ($\mathbf{L}$), and outputting the disparity value corresponding to

$$\underset{d \in [d_{min}, d_{max}]}{\arg\min} \; SAD(x, y, d) \tag{1}$$

The use of rectification [5], [6] allows to obtain horizontal epipolar lines, while to limit the search ranges a $height \times 2$ elements matrix called $\mathbf{K}$ is used, with each row containing the minimum and maximum disparity values to test during the matching phase, under the constraint that

$$\mathbf{K}[y, 0] \geq \mathbf{K}[y-1, 0] \, \forall \, y \in [1, height[\tag{2}$$

which is consistent with the intent of exploiting the information on row-wise ground disparity, and has the advantage of reducing the number of border cases to handle. The choice of having a SAD similarity metric over rectangular windows of size $(2m+1) \times (2n+1)$ leads to a matching score for a generic point at coordinates (x, y) of

$$SAD(x, y, d) \overset{\text{def}}{=} \sum_{i=-m}^{m} \sum_{j=-n}^{n} |\mathbf{R}[x+j, y+i] - \mathbf{L}[x+j+d, y+i]| \tag{3}$$

It has been shown in [1], [2], [7] that Eq. 3 can be expressed in an incremental fashion as

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

with

$$\begin{aligned} \Delta_y(x, y, d) \overset{\text{def}}{=} & \\ & \sum_{j=-n}^{n} |\mathbf{R}[x+j, y+m] - \mathbf{L}[x+j+d, y+m]| - \\ & \sum_{j=-n}^{n} |\mathbf{R}[x+j, y-m-1] - \\ & \quad \mathbf{L}[x+j+d, y-m-1]| \end{aligned} \tag{5}$$

which can be further reduced to

$$\Delta_y(x, y, d) = \Delta_y(x-1, y, d) - A(x, y, d) + B(x, y, d) \tag{6}$$

having

$$\begin{aligned} A(x, y, d) \overset{\text{def}}{=} & \\ & |\mathbf{R}[x-n-1, y+m] - \mathbf{L}[x-n-1+d, y+m]| - \\ & |\mathbf{R}[x-n-1, y-m-1] - \\ & \mathbf{L}[x-n-1+d, y-m-1]| \end{aligned} \tag{7}$$

and

$$\begin{aligned} B(x, y, d) \overset{\text{def}}{=} & \\ & |\mathbf{R}[x+n, y+m] - \mathbf{L}[x+n+d, y+m]| - \\ & |\mathbf{R}[x+n, y-m-1] - \mathbf{L}[x+n+d, y-m-1]| \end{aligned} \tag{8}$$

It is possible to avoid the explicit computation of both the $A(x, y, d)$ and $B(x, y, d)$ terms since it is redundant, as $A(x, y, d) \equiv B(x-2n-1, y, d)$: let $\mathbf{A_x}$ be an auxiliary matrix of size $2n+1 \times max\_k$, with

$$max\_k \overset{\text{def}}{=} \max_{y \in [0, height[} (\mathbf{K}[y, 1] - \mathbf{K}[y, 0] + 1) \tag{9}$$

used to hold at position $(\bar{x}, d)$, with $\bar{x} = x \mod (2n+1)$, the values of $B(x, y, d)$ as they get computed while the base window moves along a given row of R; Δ_y can then be expressed as

$$\Delta_y(x, y, d) = \mathbf{\Delta_y}[d] - \mathbf{A_x}[\bar{x}, d] + B(x, y, d) \tag{10}$$

with $\mathbf{\Delta_y}$ being a vector of size max_k containing at position d the value $\Delta_y(x-1, y, d)$.

By combinig Eq. 4 and Eq. 10 is finally possible to express $SAD(x, y, d)$ as

$$\begin{aligned} SAD(x, y, d) = & \\ & SAD(x, y-1, d) + \\ & \mathbf{\Delta_y}[d] - \mathbf{A_x}[\bar{x}, d] + B(x, y, d) \end{aligned} \tag{11}$$

Using the formulas presented so far, in order to build a complete DSI it is necessary to handle four different categories of points:

- the first point of the first row, computed using Eq. 3;

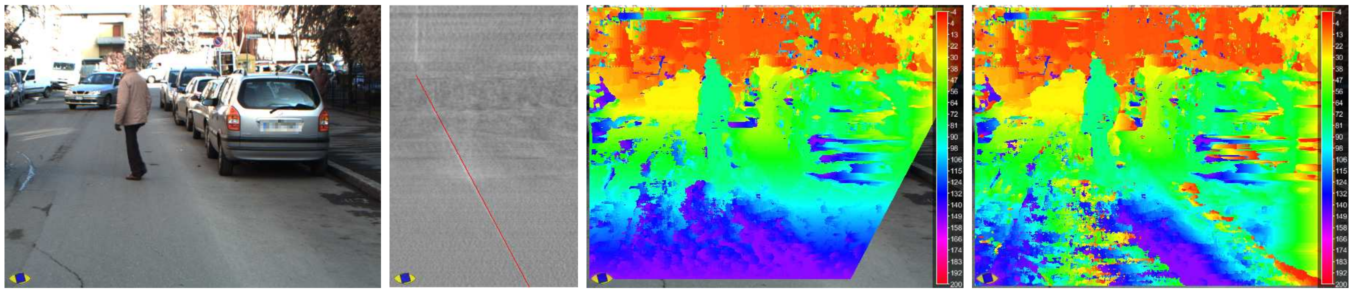

(a) (b) (c) (d)

Fig. 2. Disparity map generation: (a) right frame (512 × 384 pixels), (b) V-Disparity image with estimated ground disparities highlighted in red, (c) disparity map with range $[d_{gnd}(i), 150]\,\forall\, i \in [0, 384[$ using 8×8 pixels support windows and a 1.12 m baseline; (d) disparity map using the range $[0, 150]\,\forall\, i \in [0, 384[$. The DSI part missing in (c) corresponds to ground points in the right image falling outside the right border of the left image.

- the following points of the first row, for which the following holds true

$$\begin{aligned} SAD(x,y,d) = &\\ & SAD(x-1,y,d) + \\ & \sum_{i=-m}^{m} |\mathbf{R}[x+n,y+i] - \mathbf{L}[x+n+d,y+i]| - \\ & \sum_{i=-m}^{m} |\mathbf{R}[x-n-1,y+i] - \\ & \qquad \mathbf{L}[x-n-1+d,y+i]| \end{aligned} \tag{12}$$

  (using the same principle exploited to obtain Eq. 10, it is not necessary to compute both the incremental terms);
- the first point of a generic row, computed using Eq. 4 and 5;
- a generic point within the image, computed using Eq. 11.

Note that the $\mathbf{\Delta_y}$ and $\mathbf{A_x}$ terms are being initialized when analyizing the first row and the first pixel of a generic row respectively, and then iteratively updated as processing goes by.

It must be observed that search ranges shrink when the base window is close to the image borders:

$$\begin{cases} d_{min}(x,y) = \max(n-x, \mathbf{K}[y,0]) \\ d_{max}(x,y) = \min((width-1-n-x, \mathbf{K}[y,1]) \end{cases} \tag{13}$$

Furthermore, having set no constraint on the values of $\mathbf{K}[y,1]$ it can happen that

$$\mathbf{K}[y,1] < \mathbf{K}[y+1,1] \tag{14}$$

in which case it is no longer possible to use Eq. 11 to obtain the SAD value unless $\Delta_y(x, y+1, d)$, $d \in [\,d_{max}(x,y),\, d_{max}(x,y+1)\,]$ is computed at row y even if not used while generating $SAD(x,y,d)$, $d \in [\,d_{max}(x,y),\, d_{max}(x,y+1)\,]$. Note that since there is no guarantee on the existence of $\Delta_y(x, y-1, d)$ in the relevant interval, the most straightforward way to proceed is to use Eq. 12; while there are cases where this might not be the most efficient solution –namely, whenever $\mathbf{K}[y-1,1] \geq \mathbf{K}[y+1,1] > \mathbf{K}[y,1]$– those are a small minority, especially given the fact that if $\mathbf{K}[y,1] = f(\mathbf{G}[y])$, with $\mathbf{G}[y]$ being the ground disparity at row y, this value will tend to be increasing from top to bottom of the image.

To improve the robustness against noise in the ground disparity estimation process, and to detect some negative obstacles, it is advisable to include some disparities lower than that of the ground on a given row during the matching phase; allowing this requires to handle the fact that when $\mathbf{K}[y,0] = \bar{d} < 0$ the first $|\bar{d}|$ points of the row cannot be computed as per Eq. 11, since $\mathbf{A_x}[\bar{x}, d_{min}(x,y)]\,\forall\, x \in [\,1,\, |\bar{d}|\,[$ is not available. This happens because Eq. 13 implies that $\max(n-x, \bar{d}) = n-x$, as $n-x > \bar{d}\;\forall\, x \in [\,1,\, |\bar{d}|\,[$ with $n > 0, \bar{d} < 0$, which means that $\max(n-x, \bar{d}) > \max(n-x-1, \bar{d})\,\forall\, x \in [\,1,\, |\bar{d}|\,[$, or $d_{min}(x,y) > d_{min}(x-1,y)$, and more exactly $d_{min}(x,y) = d_{min}(x-1,y)+1$. The solution is again to abandon fully recursive computation, using Eq. 5 instead.

B. Semi-Incremental approach

If there is a special interest in reducing the processing time when generating sparse depth maps, like the one depicted in Fig. 3, a possible approach is to use only Eq. 4 and Eq. 5 to compute each score value.

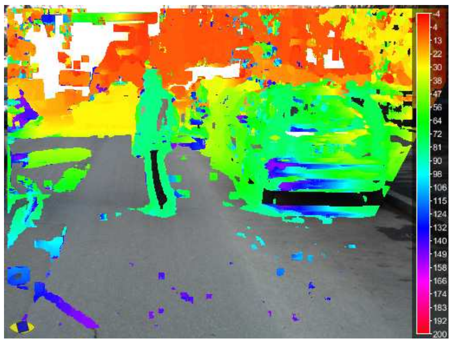


Fig. 3. Sparse disparity map: only windows having 8 bit luminance values with $\sigma^2 > 30$ are considered.

This strategy surely renders the computational burden dependent on window width, but still for small correlation windows and non-dense DSI this algorithm can outperform the one presented in Sec. II-A, especially on modern CPUs (supporting the Intel®SSE 4.1 instruction set) which allow a very efficient SIMD implementation. The strategy adopted in deriving Eq. 10 can be employed again, using a support matrix $\mathbf{C_y}$ of size $width \times 2m+1 \times max\_k$ to store intermediate results so that

$$SAD(x,y,d) = SAD(x,y-1,d) - \mathbf{C_y}[x,\bar{y},d] + D(x,y,d) \quad (15)$$

with $\bar{y} = y \mod (2m+1)$ and

$$D(x,y,d) \overset{\text{def}}{=} \sum_{j=-n}^{n} |\mathbf{R}[x+j,y+m] - \mathbf{L}[x+j+d,y+m]| \quad (16)$$

The DSI construction process in this case involves only two different categories of points:

- the points of the first row, computed using Eq. 3;
- a generic point within the image, computed using Eq. 15 and 16.

This algorithm still requires to satisfy both Eq. 13 and the constraints deriving form Eq. 14; furthermore, it must be taken into account that it is possible to completely avoid the SAD value computation for any given point only if there is no other one to compute along the same column in the m rows below and above it. Given a binary input mask of pixels to process, it is possible to (recursively) assign each to one of three categories:

- points that need both SAD values computation and score function minimization in order to obtain a disparity value;
- points that only need SAD values computation to sustain the incremental scheme;
- points that can be skipped altogether.

After this classification step, the processing takes place as explained above.

C. Full-Incremental approach

The semi-incremental algorithm presented in Sec. II-B can be made fully incremental again: it is straightforward to prove that

$$D(x,y,d) = \mathbf{D_x}[d] - E(x,y,d) + F(x,y,d) \quad (17)$$

with $\mathbf{D_x}$ being a vector of size max_k holding at position d the value $D(x-1,y,d)$, while

$$E(x,y,d) \overset{\text{def}}{=} |\mathbf{R}[x-n-1,y+m] - \mathbf{L}[x-n-1+d,y+m]| \quad (18)$$

and

$$F(x,y,d) \overset{\text{def}}{=} |\mathbf{R}[x+n,y+m] - \mathbf{L}[x+n+d,y+m]| \quad (19)$$

This means that

$$D(x,y,d) = \mathbf{D_x}[d] - \mathbf{E_x}[\bar{x},d] + F(x,y,d) \quad (20)$$

with $\bar{x} = x \mod (2n+1)$, and $\mathbf{E_x}$ being updated with values from $F(x,y,d)$ as they get computed while moving along a given row. Combining Eq. 20 with Eq. 15 finally leads to

$$SAD(x,y,d) = SAD(x,y-1,d) - \mathbf{C_y}[x,\bar{y},d] + \mathbf{D_x}[d] - \mathbf{E_x}[\bar{x},d] + F(x,y,d) \quad (21)$$

which means that each SAD value can be obtained with just one comparison, corresponding to the term $F(x,y,d)$, regardless of window size; note that $\mathbf{C_y}$ gets iteratively updated with the result of $\mathbf{D_x}[d] - \mathbf{E_x}[\bar{x},d] + F(x,y,d)$.

As it happens for the algorithm presented in Sec. II-A, there are four categories of points to consider:

- the first point of the first row, computed as

$$SAD(x,y,d) = \sum_{i=-m}^{m} \mathbf{C_y}[x,i+m,d] \quad (22)$$

  with

$$\mathbf{C_y}[x,i+m,d] = \sum_{j=-n}^{n} |\mathbf{R}[x+j,y+i] - \mathbf{L}[x+j+d,y+i]| \quad (23)$$

- the following points of the first row, obtained again using Eq. 22, but with $\mathbf{C_y}$ assuming the values

$$\mathbf{C_y}[x,i+m,d] = \mathbf{C_y}[x-1,i+m,d] - \mathbf{E'_{xy}}[\bar{x},i+m,d] + |\mathbf{R}[x+n,y+i] - \mathbf{L}[x+n+d,y+i]| \quad (24)$$

  with $\mathbf{E'_{xy}}$ being a matrix of size $2n+1 \times 2m+1 \times max\_k$ holding the incremental terms computed $2n+1$ columns before;
- the first point of a generic row, computed as

$$SAD(x,y,d) = SAD(x,y-1,d) - \mathbf{C_y}[x,\bar{y},d] + \sum_{j=-n}^{n} \mathbf{E_x}[j+n,d] \quad (25)$$

  with

$$\mathbf{E_x}[j+n,d] = |\mathbf{R}[x+j,y+m] - \mathbf{L}[x+j+d,y+m]| \quad (26)$$

- a generic point within the image, computed using Eq. 21.

The use of Eq. 22, Eq. 24 and Eq. 25 allows both to obtain the desired SAD values and to correctly initialize the auxiliary buffers. Being a fully incremental algorithm, all the border cases presented in Sec. II-A apply, but are handled differently. The effects of Eq. 14 are compensated by extending the SAD values computation using Eq. 21 beyond $\mathbf{K}[y,1]$ up to $\max_{i\in[0,2m+1]}(\mathbf{K}[y+i,1])$, in order to fill the $\mathbf{C_y}$ buffer with values needed by the subsequent rows; note however that disparity values belonging to this range are not considered during the minimization step. Negative disparities at the beginning of a row are compensated by applying Eq. 15, which does not require incremental terms from position $(x-1, y, d)$, which are not available, as previously explained.

III. Benchmarks

The test system is equipped with 8 GB DDR2 800 MHz RAM and an Intel® Core™2 Duo E8400 processor running at 3 GHz, supporting the SSE 4.1 instruction set, and featuring 128 KB L1 cache, 6 MB L2 cache and a 1333 MHz FSB. The compiler used is GNU GCC version 4.2.4, with optimization flags `-O3 -march=nocona -fforce-addr -ftracer -s`.

The algorithms presented so far have been implemented both in plain C++ and using SIMD instructions, while the multiple cores of the underlying hardware have been exploited by splitting the processing into different independent threads operating on disjoint horizontal stripes of the image.

Three tests have been carried out, involving the generation by all the described algorithms of different DSI types from a pair of 512×384 stereo images, captured using a 1.12 m baseline; various window sizes have been evaluated, and Tab. I contains, for all the proposed benchmarks, the number of window comparisons performed, defined as the sum of the active reference pixels, each multiplied by the corresponding search range size; the use of even windows is motivated by the possibility of slightly better SIMD implementations.

TABLE I
WINDOW COMPARISONS PERFORMED IN EACH BENCHMARK

	Comparisons		
Win Size [px]	Fixed Range	Ground	Sparse
4x4	23166207	14434843	8153744
8x8	22812867	14166947	9860957
12x12	22461943	13901467	10534092
16x16	22113435	13638403	10914608

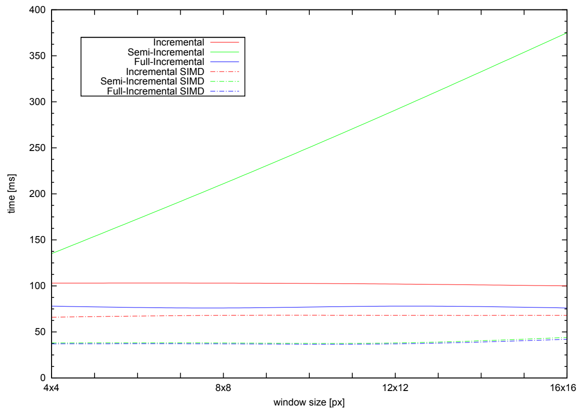
A. Fixed ranges

The first benchmark consists in the generation of the dense DSI depicted in Fig. 2-d, which is characterized by constant search ranges: $d \in [0, 150]\, \forall y \in [0, 384[$.

TABLE II
PROCESSING TIMES – FIXED RANGES

Win Size [px]	Processing Time [ms]					
	Incremental		Semi-Incremental		Full-Incremental	
	C++	SIMD	C++	SIMD	C++	SIMD
4x4	103	66	135	38	78	37
8x8	103	68	211	38	76	37
12x12	102	68	291	38	78	37
16x16	100	68	375	44	76	42

Results are as expected: in the plain C++ implementation the Full-Incremental approach outperforms the others, being the one with the lowest complexity, while the Semi-Incremental algorithm processing time is (linearly) dependent on the correlation window width. On the other hand, the SIMD implementation shows that the Semi-Incremental



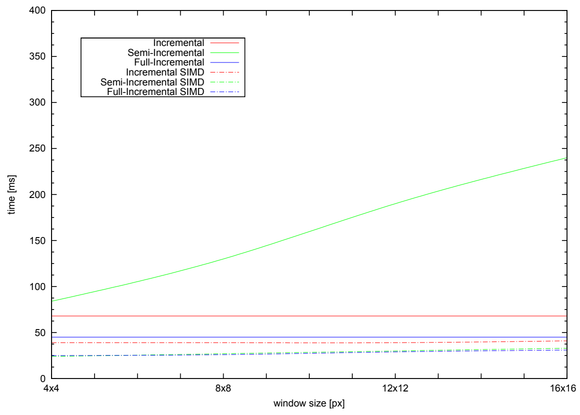
algorithm is better suited for vectorization, since it has a more direct mapping to hardware instructions, and while the Full-Incremental approach is still the fastest, the final performance difference is quite small.

B. Ground-based ranges

The second benchmark involves the generation of the dense DSI depicted in Fig. 2-c, with each search range starting some pixels below the estimated ground disparity: $d \in [\mathbf{G}[y] - 3, 150]\, \forall y \in [0, 384[$.

TABLE III
PROCESSING TIMES – GROUND-BASED RANGES

Win Size [px]	Processing Time [ms]					
	Incremental		Semi-Incremental		Full-Incremental	
	C++	SIMD	C++	SIMD	C++	SIMD
4x4	68	39	84	24	45	25
8x8	68	39	130	27	45	26
12x12	68	39	190	30	45	29
16x16	68	41	240	33	45	31



In this test figures are lower than in the preceding one, and while this is no surprise, since the number of performed comparisons has been noticeably reduced, the general trend remains unchanged.

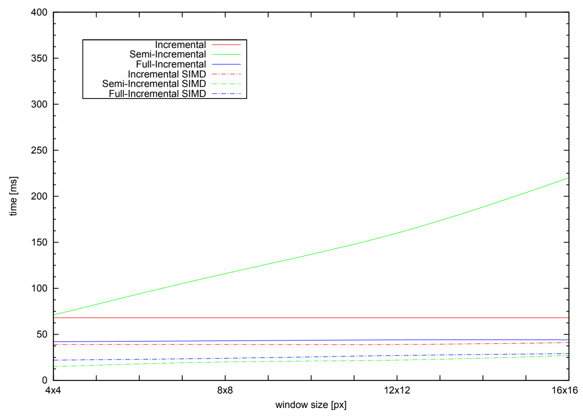
C. Sparse map

The last benchmark produces the sparse DSI depicted in Fig. 3-c, which uses the same ground-limited ranges of the previous test, with the additional constraint that only points whose reference window 8-bit luminance values have $\sigma^2 > 30$ need to be generated.

TABLE IV

PROCESSING TIMES – SPARSE MAP

Win Size [px]	Processing Time [ms]					
	Incremental		Semi-Incremental		Full-Incremental	
	C++	SIMD	C++	SIMD	C++	SIMD
4x4	68	39	71	15	42	22
8x8	68	39	116	20	43	24
12x12	68	39	160	22	44	27
16x16	68	41	220	27	44	29



The applied filtering is indeed trivial, but processing times are considerably smaller than in the two previous tests, while the information content of interest is mostly left untouched: the cars and the pedestrian appearing in the scene are still clearly visible, while the ground surface is almost completely removed.

It is interesting to note that in this case the SIMD implementation of the Semi-Incremental algorithm results the fastest one, given its ability to avoid the computation of unneeded points.

IV. CONCLUSIONS

This paper has presented three different approaches to incremental Disparity Space Image computation in automotive environments, which leverage the knowledge about the expected field of application (most notably, the presence of the ground covering a substantial portion of the image, and its lack of texture) to reduce the overall computational burden, and to improve the quality of the resulting depth map.

Performance tests have shown that on general-purpose hardware the proposed Full-Incremental algorithm can produce dense maps at near real-time rates, while keeping a control flow simple enough to allow SIMD vectorization, which almost halves the processing times; conversely, the much simpler Semi-Incremental algorithm has proven a viable solution only when adequate hardware is available: nevertheless, it results the best option if its ideal working conditions are met, which is happening most of the time in a typical urban scenario.

While a quantitative anlayisis of the quality of the generated depth maps is not easy to carry out, mainly because producing a reliable ground truth is not an easy task in outdoor environments, the extensive tests performed before and during the DARPA Grand Challenge 2005 [4] and the DARPA Urban Challenge [8] have shown that the proposed approaches provide an effective and reliable reconstruction of the vehicle surroundings.

V. ACKNOWLEDGEMENTS

The authors gratefully thank Alberto Broggi for his support and review of this project.

REFERENCES

[1] O. Faugeras, B. Hotz, H. Mathieu, T. Viéville, Z. Zhang, P. Fua, E. Théron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, and C. Proy, “Real-time correlation-based stereo : algorithm, implementations and applications,” INRIA, Tech. Rep. 2013, Aug. 1993.

[2] L. D. Stefano, M. Marchionni, S. Mattoccia, and G. Neri, “A fast areabased stereo matching algorithm,” *Image and Vision Computing*, vol. 22, pp. 983–1005, 2004.

[3] R. Labayrade, D. Aubert, and J.-P. Tarel, “Real Time Obstacle Detection in Stereo Vision on non Flat Road Geometry through “V-Disparity” Representation,” in *Procs. IEEE Intelligent Vehicles Symposium 2002*, Paris, France, June 2002.

[4] A. Broggi, C. Caraffi, P. P. Porta, and P. Zani, “The Single Frame Stereo Vision System for Reliable Obstacle Detection used during the 2005 Darpa Grand Challenge on TerraMax,” in *Procs. IEEE Intl. Conf. on Intelligent Transportation Systems 2006*, Toronto, Canada, Sept. 2006, pp. 745–752.

[5] O. Faugeras, *Three-dimensional computer vision: a geometric viewpoint*. Cambridge, MA, USA: MIT Press, 1993.

[6] A. Fusiello, E. Trucco, and A. Verri, “A compact algorithm for rectification of stereo pairs,” *Machine Vision and Applications*, vol. 12, no. 1, pp. 16–22, 2000.

[7] L. Di Stefano, M. Marchionni, and S. Mattoccia, “A pc-based real-time stereo vision system,” *MG&V*, vol. 13, no. 3, pp. 197–220, 2004.

[8] Y.-L. Chen, V. Sundareswaran, C. Anderson, A. Broggi, P. Grisleri, P. P. Porta, P. Zani, and J. Beck, “TerraMax: Team Oshkosh Urban Robot,” *Journal of Field Robotics*, vol. 25, no. 10, pp. 841–860, Oct. 2008.