Journal of Real-Time Image Processing
https://doi.org/10.1007/s11554-021-01175-y

ORIGINAL RESEARCH PAPER

Real-time stereo semi-global matching for video processing using previous incremental information

Jonay Toledo[1] · **Martin Lauer**[2] · **Christoph Stiller**[2]

Received: 25 April 2021 / Accepted: 29 September 2021

Abstract
This paper presents an incremental stereo algorithm designed to calculate a real-time disparity image. The algorithm is designed for stereo video sequences and uses previous information to reduce computation time and improve disparity image quality. It is based on the semi-global matching stereo algorithm but modified to reuse previous calculation information. Storing and reusing this information not only reduces computation time but improves accuracy in a cost filtering scheme. Some tests are presented to compare the computation time and results of the algorithm, which show that it can achieve better results in terms of quality and time than standard algorithms for some scenarios.

Keywords Stereo vision · Semi-global matching · Video processing · 3D reconstruction · Real time

1 Introduction

Robots and autonomous vehicles use depth maps as one of their main information sources for navigation, planning, etc [1, 2]. One of the most affordable and powerful sensors is a stereo camera. It is a low-cost device that can work indoors and outdoors, achieving medium to high accuracy 3D reconstructions. The main idea behind stereo is to compute the disparity as the difference between two images captured from different perspectives. It is used to detect obstacles in real time, but due to the high computation costs, usually is hardware dependent like in ref. [3] where a GPU is used to accelerate pedestrian detection. In ref. [4] a SoC FPGA-based embedded systems is used to achieve real-time stereo.

A stereo camera usually consists of two monocular and synchronized cameras with parallel optical axes and aligned image planes, which captures the same scene at the same time from different points of view. The camera rig geometry is known and the images are rectified to compensate for possible misalignments. Stereo cameras can also capture video, which must be processed to yield the disparity video map. Obtaining depth and distance from a disparity map is straightforward if the geometry of the stereo camera is known. The algorithm presented in this paper is designed to process video from moving stereo cameras in medium velocity robots and autonomous cars and it is intended for embedded devices with medium power CPUs and limited hardware resources. The objective of the algorithm is to reduce computation time and improve disparity quality by recycling previous information.

There are many algorithms designed to estimate disparity. Lazaros et al. [5] and Scharstein and Szeliski [6] provide a classification and a performance study of different algorithms. Stereo matching has a high computational demand, so good disparity needs considerable computation time or a very powerful computation system. Stereo algorithms can be divided into local matching, which only takes into account an area surrounding each point, and global matching, where the entire image is considered in a minimization function. Global matching techniques are usually more computationally expensive than local matching, but their results are generally better than local methods.

An intermediate point is the Semi-global matching (SGM) algorithm presented by Hirschmüller [7], where a semi-global search for the minimum cost of a matching

✉ Jonay Toledo
jttoledo@ull.es

Martin Lauer
martin.lauer@kit.edu

Christoph Stiller
stiller@kit.edu

[1] University of La Laguna, Canary Island, San Cristóbal de La Laguna, Spain

[2] Karlsruhe Institute of Technology, KIT, Karlsruhe, Germany

Published online: 21 October 2021

function is used to yield good minimization while reducing computational costs in comparison to global matching techniques. The authors present Mutual Information as a cost-matching pixel function, but some other functions can be used [8]. SGM has been shown to be one of the most applicable algorithms in stereo vision due to its accuracy and relatively low computation time. SGM is able to work in static tests, and in real scenarios like the KITTI dataset [9]. In these tests where different algorithms are compared, SGM variants are at or near the top, with reasonable computation times. Only machine learning-based algorithms outperforms SGM, but this kind of algorithm can only be used with similar images to the training ones. Thus, many current applications of stereo cameras use the SGM algorithm. The bibliography shows some modifications of the standard SGM, like in ref. [10], where the semi-global optimization process of the SGM is changed by a scanline optimization, providing better results but increasing computation time or in ref. [11] the stereo information is combined with cues from urban scene to improve the results and solve disparities conflicts.

SGM is not the slowest algorithm, but is far from real time in a standard implementation. As a result, many researchers have implemented variations of the algorithm in an effort to reduce its computation time. Cambuim et al. [12] presents an FPGA-based stereo vision system based on SGM. This system calculates disparity maps by streaming, which are scalable to several resolutions and disparity ranges. This paper also proposes a novel stream-ing architecture to detect noisy and occluded regions. In ref. [13], SGM is implemented in a FPGA, yielding 30 fps in 640x480 images with a 128-pixel disparity range, but using a FPGA with memory limitations implies the use of a modified algorithm to reduce memory usage, as in ref. [14]. However, a CPU implementation is more convenient for most applications; to this end, the algorithm has been modified to reduce computation time, usually by taking advantage of parallelism on a multi-core CPU or by using a Single Instruction Multiple Data (SIMD) instruction set, as in ref. [15, 16] and, where the authors present methods to improve the efficiency of SGM on general purpose PCs by relying on fine-grained parallelization and multiple cores. The improvements in SGM presented in the literature rely on its implementation in specific hardware in an effort to reduce computation time based on hardware improvements like FPGA, SIMD instructions, parallel processing, etc. To the authors' knowledge, no algorithm-based modification exists designed for video processing that reuses video image information to improve computation time independently of the hardware structure, as proposed in this paper.

2 Method description

Standard stereo methods are designed to work with a pair of static images, not with a dynamic video. The disparity computation of each pair of images starts from zero, with all previous information being discarded. The fact that the new pair of images is very similar to the last one is not used. However, this previous information can be reused to improve disparity results and reduce computation time. This scheme is represented in Fig. 1 and it offers some advantages over classical static image algorithms in terms of speed and quality.

- The difference between two consecutive images is very small if the frame rate is high enough, so part of the computational cost is already paid. The algorithm will reuse previously computed data, saving time in a dynamic programming scheme.
- A static disparity image exhibits noise due to image inconsistency, different illumination, etc. This can be filtered by reusing information from the previous frame to yield a more robust disparity image. Temporal filters made a posteriori over the final disparity image are less useful, so the disparity image information is already assigned to different discrete values and the uncertainty between these values is lost.

The main drawback of this scheme is the amount of memory necessary to save the previous computation, but for an algorithm designed to execute on a standard CPU, where a few gigabytes of memory are usually available, it is an acceptable cost. The amount of memory necessary to execute the algorithm, for a 1-byte gray-scale image, is $(2 * ImageWidth * ImageHeight * Disparity) * 2bytes$. For a 640x480 stereo image, the algorithm needs 340 MB of memory, including the extra space needed to save the current images and temporal data. In a stereo video sequence, the difference between two consecutive images depends on

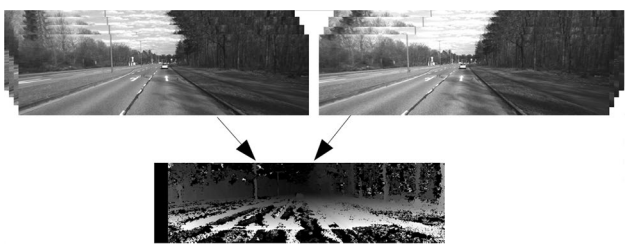


Fig. 1 Disparity computation based on previous information for a moving camera in an autonomous vehicle

frame rate and ego motion, but it is usually quite small with a medium frame rate (our tests show that only 20% of the pixels change between two consecutive frames), so the disparity map for 80% of the image is the same as in the previous frame. The main idea of this algorithm is to reuse part of this data.

2.1 Matching cost

The matching cost is the metric used to measure the similarity between two areas of the left and right images. The matching cost can be calculated using standard techniques presented in previous papers; for example, in ref. [8] the authors present a study of the application of different matching cost functions to SGM. In ref. [17] an adaptive stereo similarity measure using a weighted combination of different measures, in which the weights depend on the local image structure is presented. Disparity map accuracy depends on the matching cost, but the difference is not significant taking into account computation time. In this paper, the dissimilarity presented by Birchfield and Tomasi [18] is used as the matching cost. This method has demonstrated good results with low computation time, but any other method can be used. In the case of Birchfield and Tomasi, dissimilarity measures the difference between pixels in the epipolar line between the left and right images. It is quite tolerant to noise and illumination changes and it is calculated according to Eq. (1).

$$\begin{aligned} I^{-}(p) &= \frac{1}{2}(I(p) + I(p-1)) \\ I^{+}(p) &= \frac{1}{2}(I(p) + I(p+1)) \\ I_{min}(p) &= min(I^{-}(p), I^{+}(p), I(p)) \\ I_{max}(p) &= max(I^{-}(p), I^{+}(p), I(p)) \\ \bar{d}(p_R, p_L) &= max(0, I(p_L) - I_{max}(p_R), I_{min}(p_R) - I(p_L)) \end{aligned} \tag{1}$$

Where $p(x, y)$ is the pixel position in the epipolar line in a rectified image, or the equivalent epipolar line in the distortion image function in a non-rectified image and it can be p_L or p_R for the left or right image, and $I(p)$ is the pixel p intensity in the image. This dissimilarity $\bar{d}$ is applied for all possible disparities N_D between the left and right images.

As Eq. (1) shows, the matching cost of pixel p_L only depends on its neighborhood, so for stereo video processing, the dissimilarity is applied to every pixel in the image for the whole disparity range in each frame. In Eq. (2) C_k, with size $(W * H * D)$ (Width, Height, Disparity), represents all the matching costs for each pixel for each disparity for a frame k in a stereo image pair. The dissimilarity value for each pixel p_i and each disparity d depends only on p_{Li} and $p_{R(i+d)}$ pixels on left and right images.

$$C_k(p_i, d) = \bar{d}(p_{Li}, p_{R(i+d)}) \quad \forall d \in N_D \quad \forall i \tag{2}$$

Using the previous frame matrix C_{k-1}, the current C_k can be generated by processing only the changes between image $k-1$ and k. In order to find the changes, the frames are filtered using a bilateral filter [19], which removes noise from the image but preserves the edges. Thus, the difference between k and $k-1$ filtered frames is obtained using a temporal filter. The result is a robust inter-frame pixel change detector. Equation (3) shows the spatial filter in each image, where Ω is the size of the window filter, which is set to 5 in the tests, f_r is the range kernel and g_s is the spatial kernel, both based on a Gaussian function which is adjusted by the σ parameter, set to 35 in the tests.

$$\begin{aligned} F(p) &= \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(||I(x_i) - I(p)||) g_s(||x_i - p||) \\ W_p &= \sum_{x_i \in \Omega} f_r(||I(x_i) - I(p)||) g_s(||x_i - p||) \\ f_r(d) &= g_s(d) = \mathcal{N}(d) = e^{-\frac{(d_x)^2 + (d_y)^2}{2\sigma^2}} \end{aligned} \tag{3}$$

$$\begin{aligned} Th(p_L) &:= T(|I(p_L)_k - I(p_L)_{k-1}| > THRE) \\ Th(p_L + d(p_R)) &:= T(|I(p_R)_k - I(p_R)_{k-1}| > THRE) \\ Th(p_L) &:= ([d(p_L)_{k-1} = INVALID) \\ T(x) &:= \begin{cases} x \neq 0 \rightarrow T(x) = 1 \\ x = 0 \rightarrow T(x) = 0 \end{cases} \end{aligned} \tag{4}$$

The reference frame is the left image. $Th()$ (Eq. 4) is the function for the changed left image, where the changed pixels are set to 1 for processing. The changes are obtained directly from the pixel differences between the current and previous left images after filtering (Eq. 4 first row). The changes between two consecutive right images are represented in $Th()$ by its disparity $d(p_R)$. The previous disparity is available, so it can be used to map right-image pixels p_R in the left image. Thus, the changed pixels in the right image will be projected in the left and C_K will be calculated for those points (Eq. 4 second row). This does not change anything in theory, so if one-pixel changes in the left image, it should change in the right as well, but it does fix some glitches in the final disparity image. The points with no disparity assigned are recalculated also (Eq. 4 third row).

Equation 4 shows the threshold for obtaining a binary function which indicates if that pixel has changed between the current and previous frames. The operator $T()$ is 1 if its argument is true, and 0 otherwise. In the implementation, it is important to save the current state of each pixel image, including the value for unchanged pixels. In this way, small incremental changes in the image, smaller than the threshold, will be processed in subsequent frames if

they change enough from the original one. For example, if one pixel changes its value by 1 unit over several frames, it will be considered as a changed pixel when the difference between its current value and the value saved the last time C_k was calculated exceeds the threshold. Then, its cost C_k will be recalculated and the pixel's current value will be saved.

By applying this technique, the matrix C_k can be constructed recursively according to Eq. (5). If the neighborhood of the pixel has not changed, the previous value of C_{k-1} is used, and if the pixel has changed more than the threshold, this position is filled according to Eq. (2).

$$C_k(p,d) := Th(p)(C_k(p,d)) + (1 - Th(p))(C_{k-1}(p,d)) \quad (5)$$

These steps allow generating an equivalent C_k from C_{k-1}, but with less computation time than calculating it from zero. The changed pixels between two consecutive frames are obtained as an intermediate step of the algorithm.

2.2 Cost aggregation

The function presented in Eq. (6) is the key step in the SGM algorithm. It is an energy function $E()$ over disparity map D. By minimizing this energy function, the disparity between the left and right images is estimated. A global minimization of this 2D energy function is NP-complete for many discontinuity-preserving energies [20], so a heuristic method is proposed according to Eq. (6). The first term is the sum of all the pixel matching costs for the disparities of D. The second term is a penalty P_1 if the disparity change between two consecutive pixels is 1, and a larger penalty P_2 if this disparity change is greater than 1 in the neighborhood N_p of pixel $p(q)$. The operator $T()$ is 1 if its argument is true, and 0 otherwise as defined in Eq. (4).

$$E(D) := \sum_p C(p,D) + \sum_{q \in N_x} P_1 T(|D| = 1) + \sum_{q \in N_p} P_2 T(|D| > 1) \quad (6)$$

The key step in the semi-global matching algorithm is the search space reduction of Eq. (6). The algorithm does not search for matches in the whole search space; rather, it reduces the search to 8 or 16 directions where a 1D search is performed The results of all the searches are then combined to obtain the minimum in every direction. That reduces the computational cost from an NP-complete problem, to O(WHD) (Width, Height, Disparity), yielding a similar result, in most cases, to global matching. Usually, 8 directions are enough to provide good results with a reasonable computation time, although the initial algorithm proposes 16.

The weight in Eq. (6) includes no penalty if the disparity of the next pixel is equal to the previous one. There is a small penalty P_1 if the disparity changes by only 1 pixel, allowing for smooth changes, perspective, and curved surfaces, and a higher cost P_2 if a disparity change greater than 1 is found. The algorithm looks for small disparity increases but allows for large disparity changes if its cost plus P_2 is lower than the integration of multiple small disparity costs. This happens when a new object or the end of an object is found. These discontinuities are usually associated with changes in images.

The cost from a direction r is specified in Eq. (7), where r is one of the directions [(1,0), (− 1,0), (0,1), (0,− 1), (1,1), (1,− 1), (− 1,1), (− 1,− 1)] and L_r is the cost evaluated recursively from the previous cost and from the matching cost function in the pixel.

$$\begin{aligned} L_r(p,d) := C(p,d) + \min_i(L_r(p-r,d), \\ Lr(p-r,d-1) + P1, Lr(p-r,d+1) + P1, \\ \min_i(L_r(p-r,i) + P_2) - \min_k L_r(p-r,k) \end{aligned} \quad (7)$$

The search is applied to the whole cost function, C_k, but some part of this search can be recycled such that only changed points between two consecutive images are defined in Eq. (4). In this case, if the cost matrix C_k changes, the search starts from the changed pixel and continues until the exploration line reaches the end of the image or the minimization reaches the same cost function value, as per Eq. (6), for the same pixel in the previous frame. For instance, if a uniform textureless area is moving in the image, the only change in the cost function C_k would be the first and last pixel of this area. The search will start with the first changed pixel, and it will finish where the obstacle ends, so the disparity image will be updated properly.

After this step, the minimum cost disparities for each pixel are calculated, so the next step is to build the disparity map based on the accumulated cost $S(p, d)$ in each direction, according to Eq. (8).

$$S(p,d) := \sum_r L_r(p,d) \quad (8)$$

2.3 Disparity computation

The disparity image D_{lr} (disparity image from left to right) is calculated by looking for the disparity with the minimum cost for each pixel in Eq. (9). D_{rl} (disparity from right to left) can also be calculated in the same step by filling the appropriate cell in matrix D_{rl}. $D_{rl}(p)$ can be filled multiple times due to noise or errors in the selected disparity, occlusions, etc., so the minimum one is selected. This procedure can reduce disparity errors such that $D_{rl}(p)$ should be similar to D_{lr}. If this does not hold, it indicates a possible error has been detected and p should be marked as an invalid disparity, Eq. (10). This is the source of unassigned pixels in the disparity image, as shown in the results section.

$$\begin{aligned} D_{lr}(p) &:= min_d S(p,d) \\ D_{rl}(p) &:= min(D_{rl}(p), min_q S(p-d,q)) \end{aligned} \tag{9}$$

A sub-pixel estimation can be made by matching quadratic curves between disparities in the same image row, yielding better accuracy in the final disparity image.

$$D_{lr} := \begin{cases} if |D_{rl(p)} - D_{lr}(p)| < THRES & D_{lr} \\ else & error \end{cases} \tag{10}$$

As in the previous section, the disparity is only computed where the value of $S(p, d)$ is changed, so if the image is similar to the previous one, the calculation time would be significantly reduced.

2.4 Cost function filtering

One of the advantages of incremental disparity calculations is that the cost matrix, used to obtain the previous disparity image, is available. This allows the use of recursive filters to integrate the previous image costs into the current cost matrix to reduce noise. In a stereo disparity image obtained only from the two current images, there are many small glitches, noise, and empty zones, usually in low-texture image areas, due to the fact that the cost matrix in that part of the image is not representative. By integrating the cost function of previous frames, these noise points can be reduced; previous information can help to discriminate the current disparity for each pixel where this disparity cost is not clear in the current frame. The result is a more stable and accurate disparity map over time and the number of pixels marked as invalid is reduced.

The application of a temporal filter between two consecutive final static disparity maps can generate poor results because very different disparity values can be applied for the same pixel in two consecutive frames. This effect happens in areas where the disparity is not well defined (areas with poor texture). The filter based on the algorithm cost function C_{k-1}, C_k produces better and smoother results and the change between disparities in each frame will be smaller without increasing the computational cost. Eq. (11) includes the application of the temporal filter, where C_{kf} represents the filtered cost function, and k_1, k_2 the filtering coefficients. The weights of the filter can be selected based on the image frame rate, so if the frame rate is high, changes between images should be smaller, and the weight of previous C_{k-1} should be higher. This Cost Function filtering is applied only to the changed pixels to reduce computation time. The information from the cost function of previous frames is included in C_{k-1}, so a historic recursive filtering is maintained.

$$C_{k_f} := k_1 \cdot C_k + k_2 \cdot C_{k-1}; k_1 + k_2 := 1 \tag{11}$$

3 Results

The code is programmed in C++ using the GCC 7.5.0 compiler and OpenCV 3.1.0 as the base library [21]. It was tested on an Intel Core i7-4510U in a computer with 8GB of RAM. There is no reference dataset of execution time or performance for stereo algorithms in video sequences, so the computation time comparison is made between a standard SGM (OpenCV optimized SGM using SIMD CPU specific instructions) and the video incremental SGM presented in this paper. The computation time for the Incremental SGM depends on the frame rate. If the frame rate is low, incremental SGM tends to the same time, or a little longer than standard SGM, but maintains other advantages like robustness due to temporal cost filtering. If the frame rate is high, meaning changes between images are small, incremental video SGM reduces the time significantly compared to a standard SGM method.

The tests compare the result of the standard SGM algorithm to the incremental version presented in this paper. The incremental version applies a positional filter and reuses the previous information for the unchanged pixels. It uses a threshold to detect the change between two consecutive images. In the case of threshold 0, only the pixel with exactly the same value in the two images is reused, so the result for an SGM algorithm is not the same as in an inc algorithm with threshold 0. The incremental algorithm will work better with two similar images, and the computation time will be significantly reduced. It also works well with noisy images, where the cost filtering can improve the results. The error in the disparity image will increase with the threshold, so a convenient threshold should be selected based on the image sequence.

3.1 Synthetic test images

The first test of the algorithm is to compare the results between ground truth and disparity images. For this purpose, the synthetic video sets presented by Christian Richardt from the University of Cambridge in ref. [22] are used. The sets are composed of synthetic stereo images in five different scenarios with about 100 images in each scenario.

For the experiment, the incremental stereo algorithm presented in this paper was compared to a standard semi-global matching, yielding the results presented in Fig. 2. In Fig. 2a, the error is calculated as the sum of the difference between ground truth and the algorithm result for every pixel in the image divided by the number of pixels. In this measurement, pixels without a valid disparity assigned are removed. This experiment was conducted with different

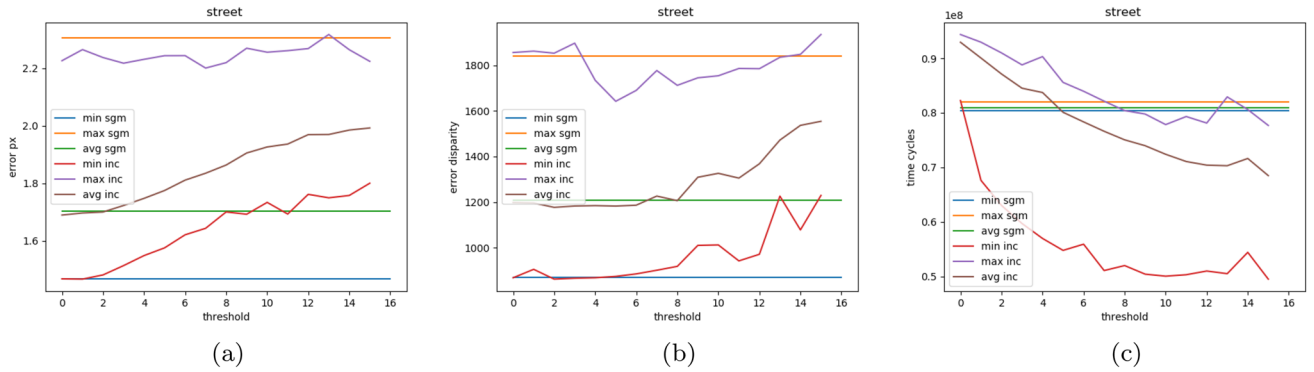


Fig. 2 Standard and incremental SGM comparison in street sequence. **a** Error for SGM and inc algorithm. **b** Number of pixels unassigned. **c** Computation time

thresholds in the incremental algorithm. As expected, the best results were obtained when the threshold is small, and the error grows with the threshold. In Fig. 2a, the error in the incremental algorithm is similar to the standard SGM algorithm for a threshold of 0. The difference between the two algorithms when the threshold is set to 0 is due to the positional filter and to the reutilization of previous data. The error increases with the threshold; for example, for a threshold of 5, the error increases by 0.1 units. The results include min and max errors for all the frames in the sequence and the average error of all the frames.

Figure 2b show the minimum, maximum and average number of pixels where the algorithm cannot find a valid disparity. The number of unassigned pixels is approximately the same as in the standard SGM algorithm with a threshold of 0; however, in Fig. 2b, the spatial filter and change detection applied to the stereo image resolve some unassigned pixels, yielding better results than the standard SGM algorithm. If the threshold increases, so does the number of unassigned pixels, as expected. Figure 2c shows the execution time in cycles of both algorithms. These times fall as the threshold increases, as expected. The street sequence have very large changes between images, so the computation time of the incremental algorithm is worse than a standard SGM; however, when the threshold increases, this time difference is reduced, and from an adequate threshold of about 5 pixels, the incremental algorithm takes less time.

Table 1 shows the tank sequence for the same data set. It includes the average processing time, error between ground truth and algorithm output, and number of unassigned pixels. The behavior of the system is similar to the street sequence when the processing time in cycles is higher, but as the threshold increases, this time is reduced, and with a threshold of 5, it is less than the standard SGM. The error and the number of unassigned pixels start from a similar value, and increase with the threshold as expected.

As this first test shows, the algorithm can yield fast results depending on the image characteristics, and should be applied to stereo videos with small differences between frames. The number of unassigned pixels and the disparity error can also be reduced. These results can be improved using temporal cost function filtering, as shown in the next sections.

3.2 Real images with ground truth

To test the algorithm with real images, we used the sequence of images presented in the Stereo Benchmark Overview of the ETH Zürich [23]. It consists of a variety of indoor and outdoor scenes using a high-precision laser scanner and

Table 1 Tank sequence results

	Time		Error		Unassigned	
Thres	SGM	Inc	SGM	Inc	SGM	Inc
0	88e6	94e6	2.39	2.39	1733.74	1733.52
2	88e6	90e6	2.39	2.51	1733.74	1755.98
4	88e6	89e6	2.39	2.82	1733.74	1883.52
6	88e6	83e6	2.39	3.20	1733.74	2124.16
8	88e6	81e6	2.39	3.67	1733.74	2493.52
10	88e6	76e6	2.39	4.08	1733.74	2785.47
12	88e6	73e6	2.39	4.36	1733.74	3174.29

captured in both high-resolution DSLR imagery and synchronized low-resolution stereo videos. The disparity ground truth images are included in the low-res many-view dataset as laser conversion to disparity images. The change between two consecutive images is greater than that designed for the current algorithm, but the actual images and the ground truth are a good test for the incremental algorithm. It is the only dataset that the authors were able to find with dense disparity images as ground truth for stereo image sequences.

Figure 3 shows the results in the terrains and electro sequences. In both, the pixel error is lower when the threshold is small, and it increases with the threshold. In Fig. 3a, the incremental algorithm yields better results than the standard algorithm even with a threshold of 8 pixels. This difference is less in Fig. 3b, but even so, with a small threshold the incremental algorithm provides a better result than the standard SGM algorithm. The number of unassigned pixels is less in the incremental algorithm for both sequences, as shown in Fig. 3c and d. This shows that in real images, where actual cameras include some noise, the filter to detect changed pixels can reduce the number of unassigned pixels. With a threshold of 0, the results are better for both cases, so the positional filter, and reusing the previous information, can improve the results. The computation time follows the expected behavior, as shown in Fig. 3e and f, that as the threshold grows, the time is reduced. The execution time for small thresholds is higher than the standard algorithm, but it decreases as the threshold grows. The algorithm can yield better results for shorter computation times than the standard algorithm.

Table 2 shows the playground sequence. In this sequence, the behavior is similar to the previous one, where the processing time is longer for the incremental algorithm with a small threshold. With a threshold of 5 or higher, the time in the incremental algorithm is reduced. The differences between ground truth and the output and the number of unassigned pixels are quite similar for both algorithms, and remain quite stable when the threshold is changed.

This test shows that the incremental algorithm works better for actual images than for synthetic ones. This happens because real images have noise and artifacts that the spatial filter for detecting changed pixels corrects. The results in terms of error and unassigned pixels are also very good.

3.3 Cost filtering

In this section, a weight-level cost filter is included in the incremental SGM algorithm. The costs obtained from the previous frame are stored and used in the next one, so information from the previous frame is included in the next one by using Eq. (11). In this case, $k_1 = k_2 = 0.5$ are used as weights. This helps the algorithm to achieve better disparity

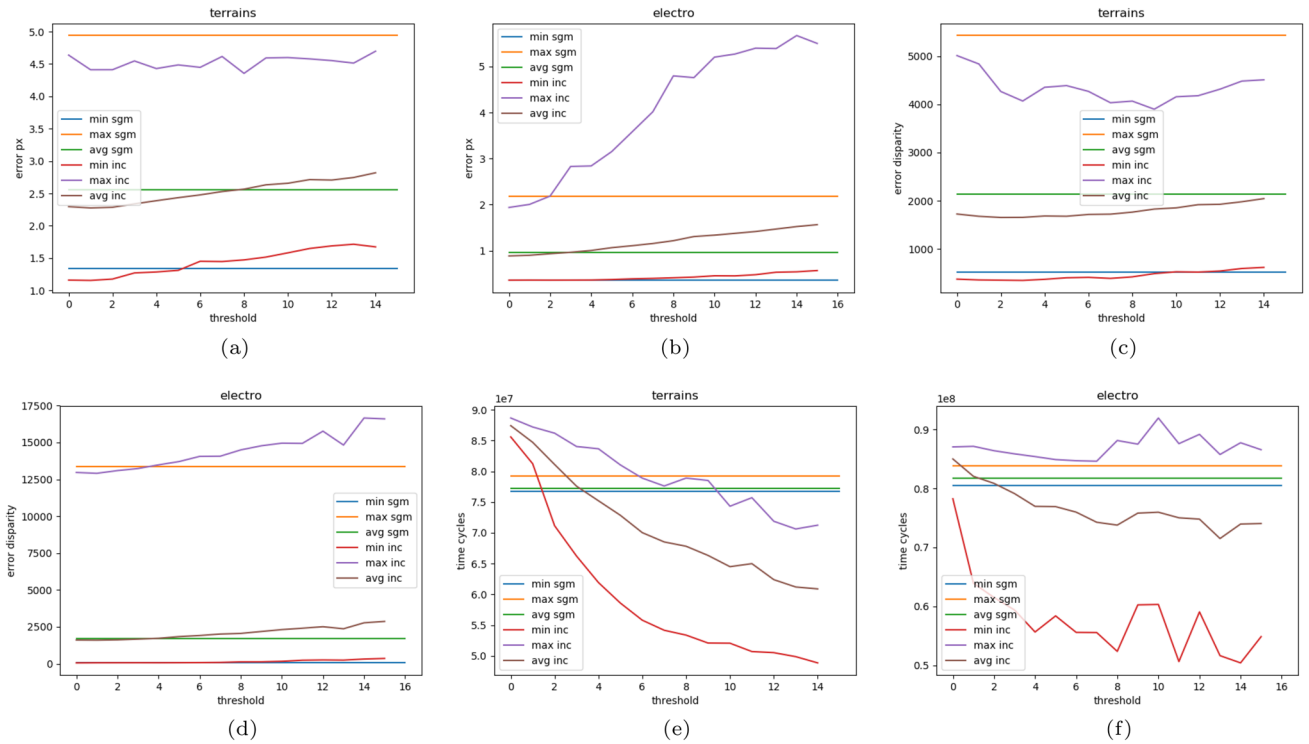


Fig. 3 Standard and incremental SGM comparison in actual images with ground truth. Error, terrains sequence (**a**) and electro sequence (**b**). Number of unassigned pixels, electro (**c**) and terrains (**d**). Computation time electro (**e**) and terrains (**f**)

Table 2 Playground sequence results

Thres	Time		Error		Unassigned	
	SGM	Inc	SGM	Inc	SGM	Inc
0	83e6	87e6	2.86	2.72	10964.72	10865.20
2	83e6	85e6	2.86	2.75	10964.72	10740.62
4	83e6	83e6	2.86	2.80	10964.72	10714.18
6	83e6	81e6	2.86	2.86	10964.72	10701.87
8	83e6	80e6	2.86	2.97	10964.72	10776.70
10	83e6	79e6	2.86	3.06	10964.72	10886.11
12	83e6	78e6	2.86	3.13	10964.72	10973.11

results when it is difficult to discriminate the correct disparity for a specific pixel. With this technique, some glitches in low-texture areas are removed, the final disparity image is smoothed, and it exhibits better consistency.

Figure 4 shows the behavior of the algorithm, including cost filtering, for the same sequence as Fig. 3. The execution time for the same sequence is approximately the same as when no cost filtering is applied (Fig. 3e, f and 4e, f); however, the disparity error between standard SGM and incremental SGM changes significantly (Fig. 3a, b and 4a, b). The temporal cost filtering function selects better disparities by using the information from the previous frame when this is not clear in the same frame. In the terrains sequence, the average disparity error decreases by 0.75 unit, or approximately a 25% reduction. In the electro sequence, the error reduction depends on the threshold, but with a threshold of 4 pixels, it is also 0.75 units, about a 30% reduction. The same thing happens with the number of unassigned pixels (Figs. 3c, d and 4c, d). The temporal cost function filter used in the incremental algorithm can assign a disparity of about 900 pixels more in the terrains and electro sequences depending on the threshold.

Table 3 shows the cost filtering playground sequence. The processing time is longer; however, the results in unassigned pixels are considerably lower. The difference between two consecutive frames reduces the algorithm's performance, but the reduction in unassigned pixels compensates for this extra time.

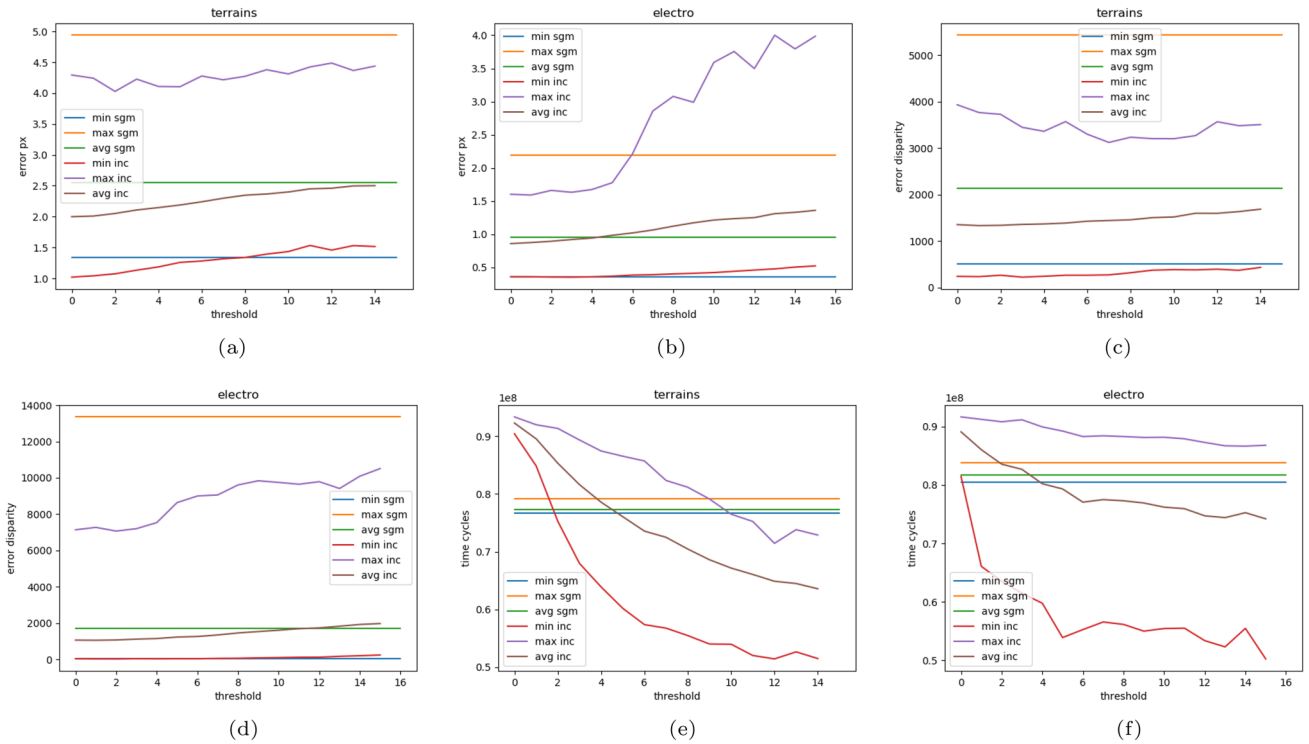


Fig. 4 Standard and incremental SGM comparison in actual images with ground truth. Error terrains sequence (**a**) and electro sequence (**b**). Number of unassigned pixels, terrains (**c**) and electro (**d**). Computation time, terrains (**e**) and electro (**f**)

Table 3 Cost filtering playground sequence results

Thres	Time SGM	Time Inc	Error SGM	Error Inc	Unassigned SGM	Unassigned Inc
0	83e6	91e6	2.86	2.80	10964.72	9113.44
2	83e6	89e6	2.86	2.82	10964.72	9046.97
4	83e6	87e6	2.86	2.89	10964.72	9057.40
6	83e6	85e6	2.86	2.99	10964.72	9057.35
8	83e6	84e6	2.86	3.05	10964.72	9209.68
10	83e6	82e6	2.86	3.11	10964.72	9238.07
12	83e6	81e6	2.86	3.22	10964.72	9404.78

The use of low-level cost function filtering between frames generates a more accurate disparity map and fewer unassigned pixels, with a negligible increase in computation time. In the examples shown in Fig. 4 with a threshold of 5 pixels, a more accurate algorithm in terms of disparity and number of unassigned filters is obtained in less time. The changes between images in these sequences are quite high, so with more suitable images, the algorithm should obtain an even lower computation time with better disparity image accuracy.

3.4 Real high speed images

The third test was carried out using the sequence of images presented by the Heidelberg Collaboratory for Image Processing HCI in ref. [24], where two Photonfocus MV1-D1312-160-CL, which deliver 100 frames per second and global shutter technology, are used to acquire the images. The sequence does not include ground truth, so error comparisons cannot be made. The images are from traffic scenes in challenging situations, such as very dark environments, snow, rain, reflections, etc.

Figure 5 shows the results of the comparison between incremental SGM and standard SGM, indicating that the average frame execution time decreases from 2.6e8 cycles to 1.8e8 cycles, a 30% time reduction (Fig. 5b), for a threshold of 5 pixels in the Avenue sequence. The number of non-matching pixels is reduced for small thresholds and it maintains approximately the same value (Fig. 5a) when the threshold increases. This experiment shows that time is reduced significantly and the disparity map quality is similar, better than many stereo block matching algorithms.

Figure 6 shows the number of unassigned pixels for the flying snow sequence of the HCI dataset. This sequence is quite challenging because it depicts a road while it is snowing, so the disparity image contains considerable noise and many unclassified pixels. The number of unassigned pixels is considerably smaller when using incremental SGM with cost filtering, Fig. 6b, than in the standard SGM Fig. 6a. The results show that weighting filters yield better results than the standard algorithm, with shorter computation times

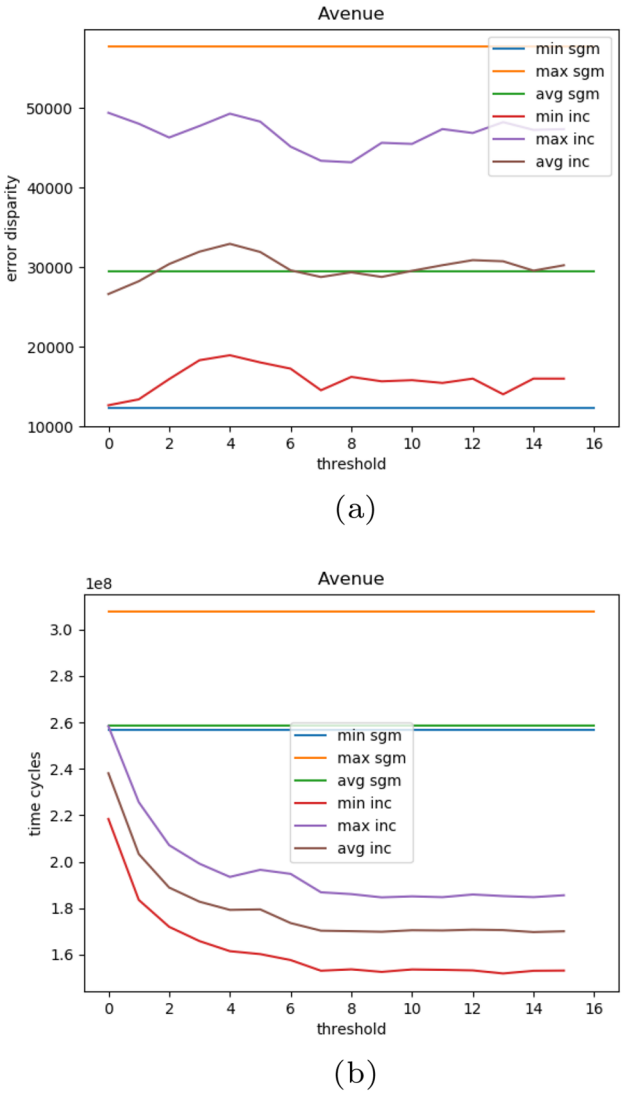


Fig. 5 Unassigned pixels (**a**) and execution time (**b**) avenue sequence

Table 4 shows the car truck sequence. The filter (filt) and not filter (nf) versions of the incremental algorithm are shown. The processing time is longer for the filter algorithm, but the number of unassigned pixels is reduced considerably.

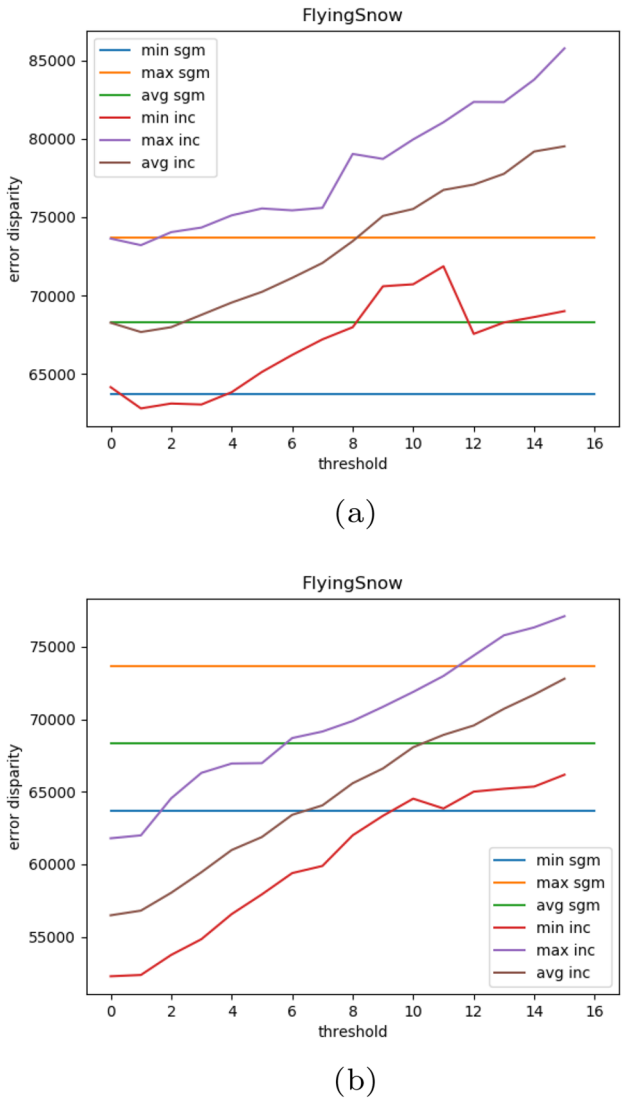


Fig. 6 Unassigned pixels without filter (**a**) and with filter (**b**) snow sequence

The execution time is reduced with the threshold, so with a threshold of 4, the execution time is shorter than the original SGM, and the number of unassigned pixels is reduced.

Figure 7 shows the same disparity image for a standard SGM algorithm Fig. 7a and the filtered version of the incremental SGM Fig. 7b in the Flying Snow Sequence. The filtered incremental SGM yields smooth surfaces, and many pixels in the road are correctly classified compared to standard SGM. The quality of the disparity maps using the filter is better and requires considerably less computation time than standard SGM.

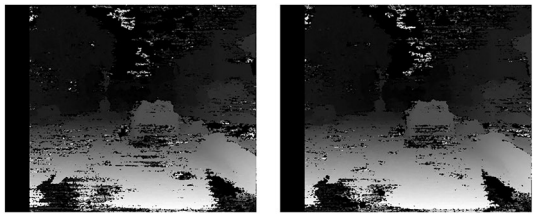
3.5 Dynamic threshold adjustment

In an actual implementation, a procedure is needed to dynamically adjust the threshold to the image characteristics. In this implementation, the frame processing time is used to tune the threshold. The expected behavior of time vs threshold is shown in Fig. 6b, where the time is reduced exponentially. In this figure, it is clear that the best threshold is about 6; a higher threshold does not improve the time significantly, but it can reduce disparity image quality.

The adjusting algorithm is shown in algorithm 1. It begins with a threshold T set to 1, and it tests different thresholds, continuously looking for the best time-quality ratio. Even frames are computed with a threshold of $T + 1$ and odd frames with $T - 1$. After processing the frame, the median processing time for the active thresholds is calculated in a time window of 30 seconds. If the processed frames with a higher threshold give a shorter computation time, the T is increased by 1 unit. The measurement is made based on the median of at least 3 elements, to avoid singular point errors. To keep the threshold from continuously increasing, the time must be decreased by a factor indicated by the user in the *PImprovement* parameter. The user can thus select the ratio between speed and quality. In this test, a 5% time reduction is needed to increase T. If the execution time of $T - 1$ is less than 5% greater than $T + 1$, the threshold is reduced. Figure 8 shows the result, and how the threshold and execution time are tuned. According to Figure 5b, the best parameter for this example is 6, which is obtained via dynamic tuning. The processing time is also reduced as expected. If the image characteristics change, the threshold will adapt by dynamically adjusting to a new optimal point.

Table 4 Car truck sequence results

	Time			Unasign		
	SGM	Inc	Inc	SGM	Inc	Inc
Th		nf	filt		nf	filt
0	25e6	26e6	28e6	15548.4	15478.7	14058.8
2	25e6	25e6	26e6	15548.4	15636.1	14065.9
4	25e6	23e6	24e6	15548.4	16296.9	14575.5
6	25e6	22e6	23e6	15548.4	17157.8	15315.7
8	25e6	22e6	22e6	15548.4	17875.9	15863.5
10	25e6	21e6	21e6	15548.4	18989.7	16553.3
12	25e6	21e6	21e6	15548.4	19619.0	17400.7



(a) (b)

Fig. 7 **a** Disparity image for standard SGM and **b** incremental SGM with actual images

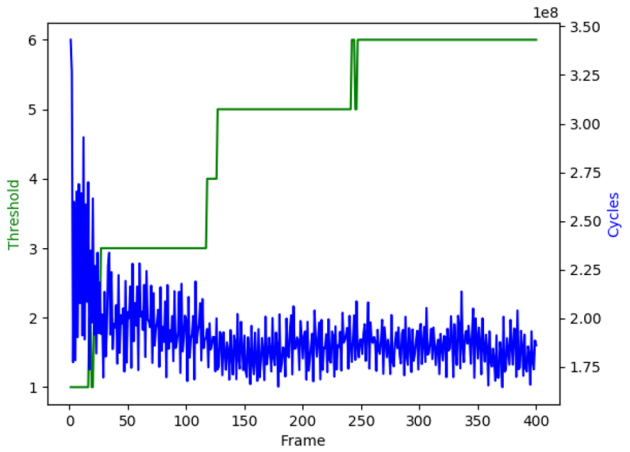


Fig. 8 Dynamic threshold adjustment for Avenue sequence

Algorithm 1 Dynamic threshold

Require: T, PImprovement
  $PI \leftarrow (1 - PImprovement)$
  if $isOdd(frame)$ **then**
    $time \leftarrow Compute(frame, T+1)$
    $t[T+1,k] \leftarrow time$
  else
    $time \leftarrow Compute(frame, T-1)$
    $t[T-1,k] \leftarrow time$
  end if
  if $median_k(t(T+1,k)) < median_k(t(T-1,k) * PI$ **then**
    $T \leftarrow T+1$
  else
    if $median_k(t(T+1,k)) > median_k(t(T-1,k) * PI$ **then**
      $T \leftarrow T-1$
    end if
  end if

4 Conclusion

This paper presents a stereo SGM that is modified to include historical information in the calculation of the disparity image. This allows for improved computation speed and improved accuracy due to temporal filtering by using the cost from previous disparity images. Previous cost filtering is applied to obtain smoother and more accurate disparity images. This algorithm is intended for CPU-based stereo with low computational resources and enough memory to save the previous computation state, such as embedded devices that process information at medium frame rates, or for high-power computation CPUs for high frame rates in autonomous vehicles.

Some tests are presented to validate the algorithm, including execution time, disparity quality and unmatched pixels. These tests show that when the change between two consecutive frames is small, the computation time for the disparity image using the incremental SGM can be considerably reduced. Cost filtering improves the disparity image quality when the image has low texture or noise without increasing computation time. The results show the advantages of the algorithm and its applicability to many scenarios. The algorithm can be used to improve execution time and disparity image quality in scenarios where the change between frames is small, yielding a real-time version of the SGM algorithm.

Acknowledgements The authors gratefully acknowledge the contribution from the Spanish Ministry of Science under “Jose Castillejo” mobility grants and University of La Laguna under “Movilidad de excelencia para el PDI de la ULL”.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

References

1. Morales, N., Toledo, J., Acosta, L., Sánchez-Medina, J.: A combined voxel and particle filter-based approach for fast obstacle detection and tracking in automotive applications. IEEE Trans. Intell. Trans. Sys. **18**(7), 1824–1834 (2017). https://doi.org/10.1109/TITS.2016.2616718
2. Morales, N., Morell, A., Toledo, J., Acosta, L.: Fast object motion estimation based on dynamic stixels. Sensors **16**(8) (2016). https://doi.org/10.3390/s16081182. https://www.mdpi.com/1424-8220/16/8/1182
3. Li, J., Wu, J., You, Y., Jeon, G.: Parallel binocular stereo-vision-based gpu accelerated pedestrian detection and distance computation. J. Real Time Image Process. **17**(3), 447–457 (2020)

4. Perri, S., Frustaci, F., Spagnolo, F., Corsonello, P.: Stereo vision architecture for heterogeneous systems-on-chip. J. Real Time Image Process. **17**(2), 393–415 (2020)
5. Lazaros, N., Sirakoulis, G.C., Gasteratos, A.: Review of stereo vision algorithms: from software to hardware (2008)
6. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. Comput. Vis. **47**(1–3), 7–42 (2002)
7. Hirschmuller, H.: Accurate and efficient stereo processing by semi-global matching and mutual information. Computer vision and pattern recognition, 2005. CVPR 2005. IEEE computer society conference on **2**, 807–814 (2005)
8. Hirschmuller, H., Scharstein, D.: Evaluation of cost functions for stereo matching. Comput. Vis. Pattern Recogn. (2007)
9. Geiger, A., Ziegler, J., Stiller, C.: StereoScan: dense 3d reconstruction in real-time. In: 2011 IEEE intelligent vehicles symposium (IV), IEEE, Baden-Baden, Germany, pp. 963–968 (2011)
10. Schonberger, J.L., Sinha, S.N., Pollefeys, M.: Learning to fuse proposals from multiple scanline optimizations in semi-global matching. In: The European conference on computer vision (ECCV) (2018)
11. Hadfield, S., Lebeda, K., Bowden, R.: Stereo reconstruction using top-down cues. Computer vision and image understanding, Large-Scale 3D Modeling of Urban Indoor or Outdoor Scenes from Images and Range Scans. **157**, 206 – 222 (2017)
12. Cambuim, L.F.S., Oliveira, L.A., Barros, E.N.S., Ferreira, A.P.A.: An fpga-based real-time occlusion robust stereo vision system using semi-global matching. J. Real Time Image Process. **17**(5), 1447–1468 (2020)
13. Banz, C., Hesselbarth, S., Flatt, H., Blume, H., Pirsch, P.: Real-time stereo vision system using semi-global matching disparity estimation: Architecture and FPGA-implementation. Proceedings—2010 International conference on embedded computer systems: architectures, modeling and simulation, IC-SAMOS 2010, pp. 93–101 (2010)
14. Hirschmüller, H., Buder, M., Ernst, I.: Memory efficient semi-global matching. ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci. **3**, 371–376 (2012)
15. Facciolo, G., Franchis, C.D., Meinhardt, E.: MGM: a significantly more global matching for stereovision. BMVC, Swansea (2015)
16. Spangenberg, R., Langner, T., Adfeldt, S., Rojas, R.: Large scale semi-global matching on the CPU. In: Intelligent vehicles symposium proceedings, IEEE, pp. 195–201 (2014)
17. Saygili, G., van der Maaten, L., Hendriks, E.A.: Adaptive stereo similarity fusion using confidence measures. Comput. Vis. Image Underst. **135**, 95–108 (2015)
18. Birchfield, S., Tomasi, C.: A pixel dissimilarity measure that is insensitive to image sampling. IEEE Trans. Pattern Anal. Mach. Intell. **20**(4), 401–406 (1998)
19. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Sixth international conference on computer vision (IEEE Cat. No.98CH36271), pp. 839–846 (1998)
20. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23**(11), 1222–1239 (2001)
21. Bradski, G.: The openCV library. Dr. Dobb's J. Softw. Tools (2000)
22. Richardt, C., Orr, D., Davies, I., Criminisi, A., Dodgson, N.A.: Real-Time Spatiotemporal Stereo Matching Using the Dual-Cross-Bilateral Grid BT (https://richardt.name/publications/dcbgrid/datasets/). Springer, Berlin, pp. 510–523 (2010)
23. Schöps, T., Schönberger, J.L., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M., Geiger, A.: A multi-view stereo benchmark with high-resolution images and multi-camera videos, (https://www.eth3d.net/overview). In: Conference on computer vision and pattern recognition (CVPR) (2017)
24. Meister, S., Jähne, B., Kondermann, D.: Outdoor stereo camera system for the generation of real-world benchmark data sets, (https://hci.iwr.uni-heidelberg.de/benchmarks/Challenging_Data_for_Stereo_and_Optical_Flow). Opt. Eng. **51**, 21101–21107 (2012)

Jonay Toledo is an Associate Professor at University of La Laguna (ULL), He received his Masterin Computer Science in 2001, Master in Electronics in 2002 and Ph. D. in Automatic Control in2008. His current research interests include mobile robots, autonomous vehicles, automatic controland embedded systems.

Martin Lauer received the Diploma degree in computer science from Karlsruhe University and thePh.D. degree in computer science from Osnabrück University in 2004. He was a Post-DoctoralResearcher with Osnabrück University in the areas of machine learning and autonomous robots.Since 2008, he has been leading a Research Group with the Karlsruhe Institute of Technology. Hismain research interests are in the areas of machine vision, autonomous vehicles, and machinelearning.

Christoph Stiller received Diploma degree in electrical engineering, in Aachen,Germany, and Trondheim, Norway, and the Ph.D. from RWTH Aachen University in 1994. In 1995,he joined the Corporate Research and Advanced Development, Robert Bosch GmbH, Hildesheim,Germany. In 2001, he became the Chaired Professor with Karlsruhe Institute of Technology,Germany. He served as the President of the IEEE Intelligent Transportation Systems Society from2012 to 2013 and the Vice-President since 2006. He served as the Editor-in-Chief of the IEEEIntelligent Transportation Systems Magazine from 2009 to 2011, and as an Associate Editor of theIEEE Transactions on Image Processing from 1999 to 2003.