EVALUATION OF VARIANTS OF THE SGM ALGORITHM AIMED AT IMPLEMENTATION ON EMBEDDED OR RECONFIGURABLE DEVICES

Matteo Poggi, Stefano Mattoccia

University of Bologna
Department of Computer Science and Engineering (DISI)
Viale del Risorgimento 2, Bologna, Italy
{matteo.poggi8,stefano.mattoccia}@unibo.it

ABSTRACT

Inferring dense depth from stereo is crucial for several computer vision applications and stereo cameras based on embedded systems and/or reconfigurable devices such as FPGA became quite popular in the past years. In this field Semi Global Matching (SGM) is, in most cases, the preferred algorithm due to its good trade-off between accuracy and computation requirements. Nevertheless, a careful design of the processing pipeline enables significant improvements in terms of disparity map accuracy, hardware resources and frame rate. In particular factors like the amount of matching costs and parameters, such as the number/selection of scanlines, and so on have a great impact on the overall resource requirements. In this paper we evaluate different variants of the SGM algorithm suited for implementation on embedded or reconfigurable devices looking for the best compromise in terms of resource requirements, accuracy of the disparity estimation and running time. To assess quantitatively the effectiveness of the considered variants we adopt the KITTI 2015 training dataset, a challenging and standard benchmark with ground truth containing several realistic scenes.

Index Terms— 3D, Stereo vision, Real-time, FPGA, Reconfigurable device, Embedded vision.

1. INTRODUCTION

Nowadays, dense depth data are required in several computer vision applications. For this reason, many sensors capable to provide dense depth maps according to different technologies have been proposed. In particular, 3D sensing devices can be broadly classified in *active* and *passive* sensors. Devices belonging to the first category infer depth by perturbing the scene with structured light patterns or other means. Notable examples of such technologies are structured light (e.g., the original Microsoft Kinect), Time of Flight (ToF) and LIDAR which measure the time between the emission of a signal and its detection in the in the receiver. Popular examples of ToF and LIDAR sensors are, respectively, Microsoft Kinect 2 and Velodyne LIDAR. Active techniques are usually quite accurate and well-suited for indoor environments. However, excluding LIDAR, they became unreliable in outdoor environment when the sensed scene is flooded by sunlight. Nevertheless, sensors based on LIDAR technology are very effective in outdoor environment but they often are quite expensive and include moving parts and cumbersome for specific application domains (e.g., wearable devices). Finally, we observe that active sensors may interfere with each other.

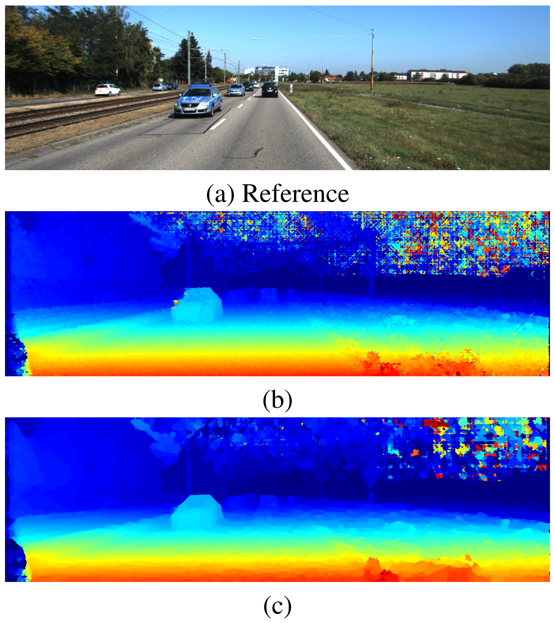
Passive devices, based on conventional imaging sensors, are not affected by most of the previously outlined limitations although with particular scene contents (e.g., in presence of low textured regions) the accuracy of passive sensors may become a major issue. In this context, depth from stereo is certainly the most popular passive technique to infer the 3D structure of the sensed scene and several algorithms have been proposed to address the *correspondence* problem. That is, starting from two or more images of the same scene acquired by different point of views, stereo algorithms aim at detecting the projection of the same point of the sensed scene in the images in order to infer depth from triangulation according to estimated parameters of the stereo setup [21]. The binocular setup, made of two imaging sensors, is the most widely adopted method to infer depth from stereo. In this field, one of the most popular stereo vision methods is the Semi Global Matching (SGM) algorithm proposed by Hirschmuller [11], which represents a good compromise between the accuracy of the inferred disparity map and the overall computational requirements. Nevertheless, due to its demanding requirements for real-time depth sensing in application domains characterized by low-power and small size constraints, many different computing architectures such as embedded computers or FPGAs have been adopted to design compact, lightweight and power efficient stereo cameras based on the SGM approach. In particular, although the proposed evaluation has a broader scope, our target architectures are mainly embedded devices containing a CPU (typically multi-core ARM processors) coupled with an FPGA (Field Programmable Gate Array) such as the Zynq 7000 device [29]. Application domains of 3D sensors based on such architectures are autonomous vehicles (e.g., rovers, drones, cars), wearable devices (e.g., assistive technology), video surveillance (e.g., people tracking and counting) and robotic/industrial applications.

In the remainder of this paper we evaluate strategies for the deployment of a state-of-the-art stereo vision pipeline based on variants of the SGM algorithm on embedded or reconfigurable devices, by taking into account factors like accuracy of the final output, resource requirements and execution time.

2. RELATED WORK

Stereo is a well-known technique to estimate dense depth maps from two or more images and several approaches have been proposed to deal with this problem. Despite this, accuracy in challenging conditions still remains an open research issue. This fact has been clearly emphasized with recent datasets [8, 18, 22] made of scenes acquired in realistic and difficult environments. While some pitfalls are intrinsically related to the stereo setup, such as occlusions [5] and distinctiveness [16], others, such as low signal-to-noise ratio and untextured regions, typically occur in challenging environmental conditions characterized by poor illumination, reflective surfaces and so on. According to[21], stereo algorithms can be classified into two categories, *local* and *global* methods, depending on the sequence of the following four steps: a) matching cost computation, b) cost aggregation, c) disparity optimization and d) final disparity refinement. Local approaches [15, 3], usually perform steps a) and b) while global methods [27] mostly focus on a) and c). Local algorithms are usually faster but more prone to errors. This behavior is clearly emphasized on challenging datasets such as KITTI [8, 9, 18] and Middlebury 2014 [22]. On the other hand, by minimizing an energy term on the whole image exploiting appropriate approximated strategies [27], global approaches provide in general more accurate results at the expense of much higher computation efforts. In recent years the SGM algorithm [11] has become very popular due to its good trade-off between accuracy and computation requirements. It relies on multiple and independent disparity optimization steps performed along different paths (referred to as *scanlines*), typically 8 or 16. Disparity optimization is performed, by means of the Scanline Optimization (SO) [21] strategy, minimizing an energy function. Although SO is very fast, disparity optimization on a 1D domain may lead to *streaking* artifacts. SGM partially softens this problem by aggregating energy computed along different paths and by selecting, by means of a winner-takes-all (WTA) strategy, the disparity label with the minimum cost.

For the reasons outlined so far, SGM has been implemented, according to different strategies and simplifications, on almost any computing architecture, such as CPUs, GPUs, FPGAs and so on. In particular, low power and massively parallel devices such as FPGAs are often the optimal choice in order to design sensors achieving optimal performance/Watt.


(a) Reference

(b)

(c)

Fig. 1. Qualitative comparison of disparity maps computed by two different configuration of the SGM algorithm processing stereo pair #40 from KITTI 2015 training set (a). In (b) a 3×3 and in (c) a 13×13 local aggregation step have been applied prior to SGM. The two disparity maps (b) and (c) lead to different results in terms of accuracy and computation requirements.

These devices (e.g., Xilinx Zynq 7000 [29]) can be configured by means of hardware description languages (HDLs), such as VHDL or Verilog, or High Level Synthesis (HLS) tools using C or C++. Examples of stereo pipeline based on SGM mapped into FPGAs devices are [2, 7, 25, 23, 14, 28, 17]. On such architectures, even more than in software, a smart design is crucial in order to achieve accurate results in real-time, without violating the limited resource constraints of the adopted architecture. Other interesting architectures suited for implementation of the SGM algorithm for real-time dense depth sensing are embedded devices based on multicore CPUs [1]. In most cases, such CPUs also enable instruction level parallelism providing Single Instruction Multiple Data (SIMD) instructions widely adopted for mapping stereo vision algorithms (e.g, [11, 4]) on CPU-based systems.

Independently of the target platform, since memory footprint and memory bandwidth represent two major issues of the SGM algorithm, in [12] was proposed a variant of the standard SGM approach, referred to as eSGM, capable to drastically reduce the memory footprint and bandwidth at the cost, however, of an increased running time and hence of the overall latency. On the software side, we mention that some recent works successfully improved the accuracy of SGM, by increasing the amount of information processed by the single SOs [6] or by deploying confidence measures and machine learning [19, 20, 24].

In the next section, we'll evaluate the performance and the resources required by some variants of the SGM algorithm in order to determine the best trade-off between resource requirements and accuracy. Although our target architectures are mainly reconfigurable devices, such as FPGAs of FPGAs with multi-core ARM processors, the outcome of this work can be of interest for other computing architectures and in particular with CPU-based embedded systems with constrained resources.

3. STEREO PROCESSING PIPELINE

In this section, we outline the structure of the processing pipeline deployed for disparity map computation. According to the taxonomy proposed by Scharstein and Szeliski [21], we can define three main steps: matching cost computation, in Section 3.1, cost aggregation, in Section 3.2, and disparity optimization, outlined in Section 3.3.

3.1. Matching cost computation

Assuming a rectified stereo pair, matching costs can be obtained according to different similarity or dissimilarity functions and by taking into account different image cues. Some methods rely on a pixel-wise strategy (e.g., absolute difference of pixel intensity) while other methods, such as Normalized Cross Correlation (NCC) or the Zero mean NCC (ZNCC), compute matching cost on patches. Among these latter approaches, the census transform [30] proved to be very effective [13] for stereo matching. It is a non-parametric local image transform based on the relative ordering of nearby pixels intensity with respect to the one under examination and it is robust to strong radiometric distortions that typically occur in practical application scenarios [13]. Each pixel intensity value is replaced by a string of $(m \times n - 1)$ bits, being $m \times n$ the window centered on the pixel itself. Then, the matching cost between two pixels is obtained computing the Hamming distance between the two corresponding census transform. Although this strategy turns out to be very effective, matching costs become very noisy in low textured areas. In [26], an improved version of the census transform was proposed using a ternary encoding which to deal with difference of intensities below a certain threshold. Compared to the original census transform, this strategy doubles the number of bits required becoming for ternary census transform $(m \times n - 1) \times 2$.

Increasing the support region deployed for the census transform leads in general to more accurate results. Unfortunately, this also increases the logic required to compute and store the outcome of the census transform. This fact is particularly relevant when dealing with FPGAs. Thus, in order to cover larger areas, enabling reduced resource requirements without renouncing to effectiveness of the census transform, we deploy a chessboard pattern, as shown in Figure 2, taking into

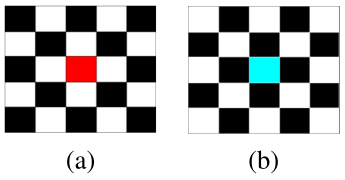


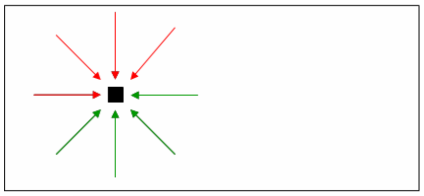
Fig. 2. Simplified patterns deployed to compute the census transform: *even* (a) and *odd* (b).

account only half of the total pixels contained into the window. Considering the same area, this strategy halves the number of bits required for census transform $((m \times n - 1)/2\ )$ and for the ternary counterpart $((m \times n - 1)\ )$. Depending on the initial considered pixel at the top-left of the window, we define two patterns, referred to as *odd* and *even* pattern and shown in Figure 2.

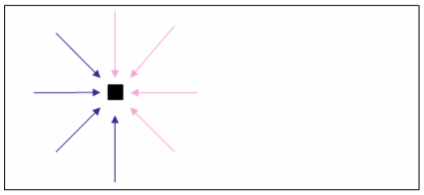
3.2. Cost aggregation

Once computed the matching cost for each pixel and for each disparity (this data structure is typically referred to as Disparity Space Image (DSI)), a local aggregation step is usually deployed in order to reduce matching ambiguity. The Block Matching (BM) algorithm is a fast local aggregation method, which sums equally the contribution of the pixels inside a window of size $N \times N$. Its accuracy is related to the choice of the window. A small patch enables good performance near depth discontinuities but provides poor results when dealing with low-textured areas. On the other hand, a larger patch is more accurate in this latter circumstance but does not preserve depth discontinuities. Differently from more sophisticated and accurate approaches such as those based on the guided [10] filter approach [15, 3] that take into account cues from the reference image, the BM strategy enables a fast DSI refinement suited for real-time performance.

However, summing the costs on a window of size $N \times N$ leads to an increase of the maximum obtainable value. Although this fact does not represent a problem for a standard software implementation, targeting the FPGA, the additional resources required to encode the aggregated costs may represent a major issue due to more complex design and memory footprint. Specifically, compared to a point-wise matching cost, on a $N \times N)$ window it requires $log_2(N \times N)$ more bits. In order to maintain as low as possible the hardware resource requirements, a solution consists in averaging rather than simply summing the values within the window. This strategy enables to encode the aggregated cost with the same number of bit of the point-wise cost. On the other hand, this strategy reduces the range of the matching costs and might lead to a lower overall accuracy. However, as we'll show in the experimental result section, averaging turns out to be a very good trade-off.



(a)



(b)

Fig. 3. The eight scanline involved into SGM_8 algorithm. On top, red paths are deployed by SGM_8 during the first image scan (from top-left to bottom-right), green paths during the second scan (from bottom-right to top-left). The paths deployed by SGM_a variant are depicted in red, by SGM_b variant in green. On bottom, the scanlines deployed by SGM_c variant are depicted in blue and by SGM_d variant in pink).

3.3. Semi Global Matching

As already pointed out, the SGM algorithm [11] has been widely adopted for hardware related implementations due to its good trade-off between accuracy and computation requirements. SGM relies on the SO algorithm that enables to minimize, along a certain linear path referred to as scanline, the following energy term for each pixel p:

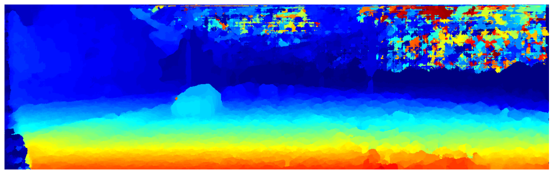
$$E(p, d) = E_{data}(p, d) + E_{smooth}(p, d) \quad (1)$$

The data term E_{data} encodes the (aggregated or pointwise) matching cost for pixel p while the smoothing term E_{smooth} enforces the local smoothness of the disparity map. In particular, SGM adopts two smoothing penalties, $P1$ and $P2$. Their role is to encourage small disparity changes between nearby pixels penalizing higher discontinuities on the disparity maps. To achieve this, $P1$ has to be strictly minor than $P2$. Then, for a pixel p its E_{smooth} term at disparity hypothesis d is computed according to equation 2,

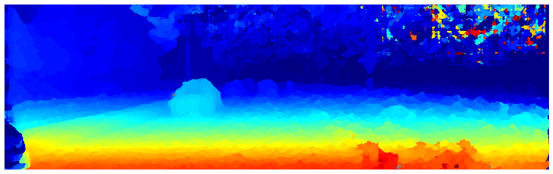
$$E_{smooth}(p, d) = \min_{q>1} \big(C(p-1, d), C(p-1, d \pm 1) + P1, \\ C(p-1, d \pm q) + P2 \big) - min_k \big(C(p-1, k) \big) \quad (2)$$

which takes the minimum cost of the pixel in position $p-1$ along the path summed to its penalty term (computed according to the difference between the disparity hypothesis d and the one assigned to the previous pixel $p-1$ along the scanline). Moreover, the minimum costs computed at $p-1$ is subtracted from the smoothness term. This enables to obtain an upper bound for the final cost given by the maximum value of the data term E_{data} plus $P2$. According to the SO algorithm defined so far, SGM performs such procedure independently along multiple paths (e.g., 8 or 16), summing up the cost computed along each scanlines. In its final stage, SGM determines the best disparity according to a winner-takes-all (WTA) strategy on the resulting sum. While the single SO usually produces gross artifacts on the resulting disparity map, the strategy adopted by SGM leads to much more accurate results. Concerning efficiency, the key factor is that multiple SO instances can be independently computed, thus enabling parallel implementation on suitable architectures. However, the original SGM algorithm [11] requires to scan the stereo pairs two times and has a very large memory footprint to store the whole DSI. A typical implementation of the SGM algorithm on 8 scanlines, referred to as SGM_8, consists of a two-phase approach, which first scans the image from top-left to down-right and performs four out of the eight total scanlines, then scans from down-right to top-left to compute the remaining four paths, as depicted in Figure 3 (a). Unfortunately this strategy requires to store an entire DSI, in order to sum the contribution of the different paths in the two scans. Another solution, is to perform a single scan only, deploying only four out of eight scanlines. This allows to compute disparity values *on-the-fly*, by storing only a single line of the image during the scan and thus greatly reducing the memory requirements. On the other hand, using only these four scanlines usually reduces also the overall accuracy compared to the original SGM strategy. In the remainder we we'll refer to an implementation scanning the image from top-left to bottom-right as SGM_a and to the other solution (scanning from bottom-right to top-left) as SGM_b. The memory footprint of SGM_a and SGM_b is much lower with respect to SGM_8 being related only by the width of the stereo pair and the disparity range (while the memory footprint of SGM_8 is related to width and height of the stereo pair and the disparity range). Moreover, being in a typical stereo pair the image width larger than the image height, we also take into account two more methods, referred to as SGM_c and SGM_d. These two further approaches scan the image, respectively for SGM_c and SGM_d, from bottom-left to top-right and from top-right to bottom-left. Figure 3 depicts the eight scanlines, highlighting those processed by SGM_a (red), SGM_b (green), SGM_c (blue) and SGM_d (pink). It is worth to note that variants based on 4 paths halve the latency of the original SGM_8 algorithm.

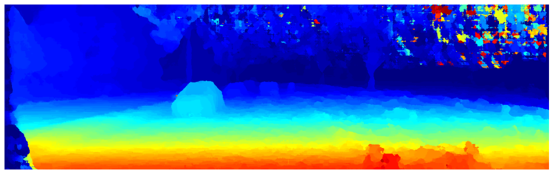
Another strategy to reduce the memory requirements of SGM_8 was proposed in [12]. This variant, referred to as eSGM, aims at obtaining results comparable to SGM_8 with a much lower memory footprint. Starting from the assumption that, in most cases, the disparity chosen by the WTA on the aggregated SOs is one of those obtained by applying the WTA on the outcome of the single SOs, eSGM only stores a $w \times h \times 4$ DSI achieving an accuracy almost equivalent to SGM_8. Unfortunately, this method requires a third scan,



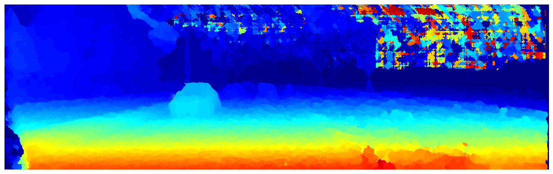
(a)



(b)



(c)



(d)

Reference

Fig. 4. Qualitative comparison between SGM_a (a), SGM_b (b), SGM_c (c) and SGM_d (d) on the stereo pair nr. 40 (reference image at the bottom) from KITTI 2015 training dataset, highlighting different artifacts on the disparity maps depending on the adopted strategy.

increasing the running time/latency of SGM_8 by 50%.

4. EXPERIMENTAL RESULTS

In this section, we evaluate several configuration of the pipeline outlined on the KITTI 2015 training dataset [18]. In order to reduce the amount of parameters in our evaluation we fix the size of the census transform to 5×5 that, according to our experiments, gives the best trade-off between accuracy and resource requirements (setting the upper bound of the matching cost to 24, enabling a 5 bit encoding). For ternary transform, we set a threshold value of 1. We also tuned $P1$ and $P2$ terms to, respectively, 2 and 30. These penalties are multiplied by the size of the window deployed for cost aggregation only if the results are not averaged, in order to keep the same relationship between matching costs and penalties.

census grid	BM size	err. rate	bits per cost
binary, full	1×1	10.18%	6
ternary, full	1×1	10.16%	7
binary, full	3×3	8.75 (8.78%)	9 (6)
ternary, full	3×3	8.61 (8.63%)	10 (7)
binary, full	5×5	8.03 (8.04)%	11 (6)
ternary, full	5×5	7.77 (7.81)%	11 (7)
binary, full	7×7	7.58 (7.60)%	12 (6)
ternary, full	7×7	7.28 (7.32)%	12 (7)
binary, full	9×9	7.37 (7.39)%	13 (6)
ternary, full	9×9	7.01 (7.04)%	13 (7)
binary, full	11×11	7.33 (7.33)%	13 (6)
ternary, full	11×11	6.90 (6.92)%	14 (7)
binary, full	13×13	**7.33 (7.33)%**	14 (6)
ternary, full	13×13	**6.90 (6.92)%**	14 (7)

Table 1. Comparison between binary and ternary census transform, computed on full grid and increasing window size for local aggregation (BM) and SGM_8. Error rates on the KITTI 2015 training dataset are reported for aggregated and averaged matching costs. The rightmost column reports the number of bits required to encode aggregated and averaged matching costs.

Table 1 reports results concerned with SGM_8 by increasing the size of the window deployed by the BM strategy for cost aggregation. Each row shows results concerned with binary and ternary census transform, computed on the full grid with 5×5 patches. The first two rows are concerned with point-wise matching costs. In our experiments, for both transforms, the error rate decreases until the aggregation window is 13×13. The second error rate, reported within round brackets, is obtained by averaging the costs with respect to the window size. We can observe as this operation increases the error rate by an amount smaller than 0.04% in the worst cases. On the rightmost column, we report the number of bits for each cost value. Of course, without averaging, it increases with larger windows. On the other hand, if we average the costs over the window, the amount of bit required to encode these values is the same and equal to the case of missing aggregation. Compared to the point-wise matching costs, aggregating and averaging on larger windows enables to improve (respectively, about 2.85% and 3.24% reduction of the error rate on binary and ternary transforms) at no additional cost in terms of resource requirements. Therefore, to summarize, averaging matching cost turns out to be an optimal strategy when targeting reconfigurable devices.

Table 2 reports experiments related to SGM_8 with full and sparse census grids (odd and even) for binary and ternary census transform on the most accurate configuration (i.e., 13×13) reported in Table 1. While a sparse grid slightly increases

census	err. rate	bits per cost
binary, full	7.33 (7.33)%	14 (6)
binary, odd	7.45 (7.56)%	13 (5)
binary, even	7.36 (7.47)%	13 (5)
ternary, full	6.90 (6.92)%	14 (7)
ternary, odd	6.73 (6.80)%	14 (6)
ternary, even	**6.72 (6.76)%**	14 (6)

Table 2. Comparison between binary and ternary census transform, with local aggregation on 13×13 window and full, odd and even sparse grids. Ternary census transform on even sparse grids outperforms all the other configurations. In particular, in its average form (values between round brackets), the bits per cost are the same required by the binary census transform but ternary achieves a lower error rate.

census	err. rate
binary full,	7.39 (7.93)%
binary odd,	7.37 (7.51)%
binary even,	7.26 (7.40)%
ternary full,	6.94 (6.97)%
ternary odd,	6.77 (6.84)%
ternary even,	6.76 (6.81)%

Table 3. Error rate for eSGM algorithm with both binary and ternary transforms, with full, odd and even grids.

the error rate of the binary census transform, the ternary census transform on sparse grids allow to increase the overall accuracy. In particular, the even sparse grid with ternary census transform achieve the best result compared to all the other configurations As for previous case, averaging slightly reduces accuracy. However, for the reason previously outlined, the averaged even sparse grid with ternary census transform represents the optimal choice dealing with reconfigurable devices enabling for a notable reduction of bits required to store costs in Block-RAM and an overall reduction of LUTs (Look-Up-Tables) for computation.

In Table 3, we report results concerned with the eSGM algorithm. It performs 3 scans instead of the 2 required by SGM_8, but requires much less memory to store a portion of the DSI. By looking at binary and ternary census in their three variants, we can observe how the error rates are very close to those achieved by SGM_8 (less than 0.60% and 0.08% difference, respectively, for binary and ternary transforms on worst case). This means that this solution is highly valuable when there are no strict timing constraints (it introduces a third scan, increasing the running time/latency by 50%) but the memory requirements, in terms of footprint and/or bandwidth, are major constraints such in the case of FPGA devices.

Table 4 reports the results obtained by the single scan variants of SGM, respectively SGM_a exploiting only top-left to bottom-right scanlines and SGM_b using the remaining, SGM_c using bottom-left to top-right paths and SGM_d the remaining. We tested these approaches with binary and ternary transform on the full, odd and even grids. The results reported in Table 4 show how these variants lead to very different results in terms of accuracy, with a notable margin between the highest and lowest error rates. Therefore, it is crucial to choose the optimal scan strategy. On KITTI 2015, SGM_d achieve the best results in the ternary-even configuration and, moreover, it enables the lowest memory footprint (the same of SGM_c), being related to the height of the input image instead to its width as for SGM_a and SGM_b. Figure 4 shows, qualitatively, how the choice of the method (SGM_a, SGM_b,SGM_c and SGM_d) affects the resulting disparity maps on a frame of the KITTI 2015 training dataset.

Finally, we collect the most accurate configurations for the three variants of SGM (single-scan algorithms, original two-scans SGM_8 and $eSGM$) on Table 5 to highlight their strengths and weaknesses. We can observe how the best performing single-scan algorithm on KITTI 2015, which is SGM_d, also is the fastest method and requires the lowest amount of memory when averaging the costs during local aggregation. On the other hand, it yields a higher error rate with respect to the other two methods. The most accurate results are achieved by the original SGM_8 method, which also requires the highest amount of memory (more than 15 times the amount required by the other solutions), doubling the running time/latency with respect to the single-scan approach SGM_d. Finally, $eSGM$ enables an optimal trade-off in terms of accuracy and memory requirements, but increases the running time/latency by 50% with respect to the original algorithm SGM_8, maintaining almost the same error rate, and by 200% compared to the single-scan method, which obtain a nearly 2% increased error rate with respect to SGM_8 and $eSGM$.

5. CONCLUSIONS

In this paper, we have reported an extensive evaluation of different configurations of a stereo pipeline based on variants of the SGM algorithm suited for implementation on embedded or reconfigurable devices with limited resources. We evaluated the six variants of the SGM in 90 different configurations and validated the results on the challenging KITTI 2015 training dataset with ground truth. In our experiments we found out that the ternary census transform is slightly more effective than the binary version. Moreover, we observed that averaging the matching cost leads to results almost equivalent to the conventional matching cost summation strategy. This fact is very important when dealing with reconfigurable devices in order to reduce hardware resources. Finally, we highlighted the three top-performing variants of the SGM algorithm, with their own strengths and weaknesses, in order to find the best compromise in terms of accuracy, memory requirements and running time.

Acknowledgments The authors would like to acknowl-

census	SGM_a error	SGM_b error	SGM_c error	SGM_d error
binary full	10.37 (10.57)%	12.88 (12.55)%	8.72 (8.59)%	8.88 (9.01)%
binary odd	13.96 (14.78)%	18.31 (18.10)%	9.42 (9.31)%	9.90 (10.35)%
binary even	14.53 (15.32)%	18.82 (18.52)%	9.59 (9.48)%	10.14 (10.59)%
ternary full	9.37 (9.57)%	12.27 (11.96)%	8.70 (8.55)%	8.27 (8.41)%
ternary odd	10.28 (10.60)%	12.53 (12.26)%	8.47 (8.34)%	8.41 (8.64)%
ternary even	10.62 (10.97)%	12.97 (12.69)%	8.61 (8.48)%	8.54 (8.81)%

Table 4. Evaluation of the four single-scan variants of SGM: SGM_a, SGM_b, SGM_c and SGM_d, as depicted in Figure 3. Full, odd and even grid for both binary and ternary census transforms are reported, with local aggregation on a 13×13 window, results with average of the costs are reported between round brackets.

Algorithm	Error rate	Memory	Number of scans
ternary full, 13×13, SGM_d	8.27 (8.41)%	**0.47 (0.20)** MB	**1**
ternary odd, 13×13, SGM_8	**6.72 (6.76)%**	205.50 (88.07) MB	2
ternary odd, 13×13, $eSGM$	6.77 (6.84)%	16.11 (6.90) MB	3

Table 5. Evaluation of the most accurate single-scan algorithm SGM_d, the original two-scan SGM_8 algorithm and the $eSGM$ algorithm. In bold, the strengths of each method.

edge with thanks Andrea Bombino for his contribution to the implementation of the code concerned with the experimental results reported in this paper.

6. REFERENCES

[1] Oliver Jakob Arndt, Daniel Becker, Christian Banz, and Holger Blume. Parallel implementation of real-time semi-global matching on embedded multi-core architectures. In *2013 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, SAMOS 2013, Agios Konstantinos, Samos Island, Greece, July 15-18, 2013*, pages 56–63, 2013.

[2] Christian Banz, Sebastian Hesselbarth, Holger Flatt, Holger Blume, and Peter Pirsch. Real-time stereo vision system using semi-global matching disparity estimation: Architecture and fpga-implementation. In *IC-SAMOS*, pages 93–101, 2010.

[3] L. De-Maeztu, S. Mattoccia, A. Villanueva, and R. Cabeza. Linear stereo matching. In *A13th International Conference on Computer Vision (ICCV2011)*, November 6-13 2011.

[4] Luigi Di Stefano and Stefano Mattoccia. Fast stereo matching for the videt system using a general purpose processor with multimedia extensions. In *Proceedings of the Fifth IEEE International Workshop on Computer Architectures for Machine Perception (CAMP'00)*, CAMP '00, pages 356–, Washington, DC, USA, 2000. IEEE Computer Society.

[5] Geoffrey Egnal and Richard P. Wildes. Detecting binocular half-occlusions: Empirical comparisons of five approaches. *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, 24(8):1127–1133, 2002.

[6] G. Facciolo, C. de Franchis, and E. Meinhardt. Mgm: A significantly more global matching for stereovision. In *British Machine Vision Conference (BMVC)*, 2015.

[7] Stefan K. Gehrig, Felix Eberli, and Thomas Meyer. A real-time low-power stereo vision engine using semi-global matching. In *ICVS*, pages 134–143, 2009.

[8] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The kitti dataset. *Int. J. Rob. Res.*, 32(11):1231–1237, sep 2013.

[9] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[10] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. In *Proceedings of the 11th European Conference on Computer Vision: Part I*, ECCV'10, pages 1–14, Berlin, Heidelberg, 2010. Springer-Verlag.

[11] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(2):328–341, feb 2008.

[12] Heiko Hirschmuller, Maximilian Buder, and Ines Ernst. Memory efficient semi-global matching. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 371–376, 2012.

[13] Heiko Hirschmuller and Daniel Scharstein. Evaluation of stereo matching costs on images with radiometric

differences. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(9):1582–1599, sep 2009.

[14] D. Honegger, H. Oleynikova, and M. Pollefeys. Real-time and low latency embedded computer vision hardware based on a combination of fpga and mobile cpu. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4930–4935, Sept 2014.

[15] Asmaa Hosni, Christoph Rhemann, Michael Bleyer, Carsten Rother, and Margrit Gelautz. Fast cost-volume filtering for visual correspondence and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(2):504 – 511, 2013.

[16] R. Manduchi and C. Tomasi. Distinctiveness maps for image matching. In *Image Analysis and Processing, 1999. Proceedings. International Conference on*, pages 26–31. IEEE, 1999.

[17] Stefano Mattoccia and Matteo Poggi. A passive rgbd sensor for accurate and real-time depth sensing self-contained into an fpga. In *Proceedings of the 9th International Conference on Distributed Smart Cameras*, ICDSC '15, pages 146–151, New York, NY, USA, 2015. ACM.

[18] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[19] Min-Gyu Park and Kuk-Jin Yoon. Leveraging stereo matching with learning-based confidence measures. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[20] Matteo Poggi and Stefano Mattoccia. Learning a general-purpose confidence measure based on o(1) features and asmarter aggregation strategy for semi global matching. In *Proceedings of the 4th International Conference on 3D Vision, 3DV*, 2016.

[21] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42, apr 2002.

[22] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR'03, pages 195–202, Washington, DC, USA, 2003. IEEE Computer Society.

[23] K. Schmid and H. Hirschmuller. Stereo vision and imu based real-time ego-motion and depth image computation on a handheld device. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4671–4678, May 2013.

[24] Akihito Seki and Marc Pollefeys. Patch based confidence prediction for dense disparity map. In *British Machine Vision Conference (BMVC)*, 2016.

[25] Yi Shan, Yuchen Hao, Wenqiang Wang, Yu Wang, Xu Chen, Huazhong Yang, and Wayne Luk. Hardware acceleration for an accurate stereo vision system using mini-census adaptive support region. *ACM Trans. Embed. Comput. Syst.*, 13(4s):132:1–132:24, apr 2014.

[26] Fridtjof Stein. Efficient computation of optical flow using the census transform. In *DAGM-Symposium*, volume 3175 of *Lecture Notes in Computer Science*, pages 79–86, 2004.

[27] Jian Sun, Nan-Ning Zheng, and Heung-Yeung Shum. Stereo matching using belief propagation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(7):787–800, jul 2003.

[28] C. Ttofis and T. Theocharides. Towards accurate hardware stereo correspondence: A real-time fpga implementation of a segmentation-based adaptive support weight algorithm. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, pages 703–708, March 2012.

[29] Xilinx. Zynq-7000 all programmable soc, 2016.

[30] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In *Proceedings of the Third European Conference on Computer Vision (Vol. II)*, ECCV '94, pages 151–158, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.