Image Rectification and Matching Solutions Optimized for Dedidated Hardware Accelerators

Cristian-Cosmin Vancea
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
e-mail: Cristian.Vancea@cs.utcluj.ro

Sergiu Nedevschi
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
e-mail: Sergiu.Nedevschi@cs.utcluj.ro

Abstract—**The late explosion of hardware accelerated devices on market has enabled much interest towards fast and energy efficient solutions for the computationally intensive problem of image matching and dense 3D reconstruction. From the perspective of camera projection model the task can be simplified if the images are rectified for epipolar alignment in a preprocessing step. We focus our interest on identifying a rectification model and several classes of matching algorithms as candidate for real-time implementation into embedded devices installed on vehicles. Then we make a thorough analysis of existing solutions and identify the capabilities offered by the hardware underneath. We present our solutions for image rectification and matching in FPGA. We point the benefit from implementing an original caching technique which optimizes access to the image memory when performing rectification. Additionally, we have designed several optimization strategies capable to reduce the hardware costs for image matching. Our system running on an old class FPGA performs image rectification and matching in real-time and classifies between the top best solutions in computational performance and energy efficiency.**

Keywords—stereovision; image rectification; image matching; FPGA; digital design

I. INTRODUCTION

The field of autonomous vehicles requires state-of-the-art solutions in domains such as perception, decision-taking, motion planning and control. Perception plays a significant role in the pipeline by extracting accurate information from the surrounding environment, which is critical for searching a feasible trajectory at the higher decision-taking level. Details such as vehicle dynamics, position of obstacles, road boundaries, represent critical data for accurate and safe in-traffic decisions. With digital world becoming more visual the number of images needed to be processed in real-time is increasing. Such a task is demanding in both, computation and memory. Single-machine environments [1] often lack sufficient computational power and memory, while multi-machine environments add communication and control overhead. For hardware makers the push is to create low-cost systems to ensure real-time analysis and reduce the amount of data traffic. There is increasing demand for smaller, energy-efficient and powerful computational systems that can be placed outside offices, in the end devices, such as smartphones or automated vehicles. That requires not only new materials but also different packaging to ensure high performance with low energy cost and increased density.

In terms of energy efficiency Field Programmable Gate Arrays (FPGA) are superior to high-end Graphics Processing Units (GPU), which offer impressive floating-point performance. Technology is advancing rapidly and with the increasing number of integrated Digital Signal Processing (DSP) units in the FPGA fabric, their floating point capability is continuously improving. The key point for both classes of devices is their high degree of parallelism.

Regarding applications for Advanced Driver Assistance Systems (ADAS), the need to facilitate smarter vision tasks is well concentrated on implementation with hardware accelerators [2][3][4] due to their capabilities for real-time analytics (detection, recognition, classification and tracking), video processing (2D, 2.5D, 3D visualization and reconstruction) and intelligent data transport through I/O components (Ethernet, AVB and CAN).

Nowadays markets share a large variety of components based on LIght Detection And Ranging (LIDAR), radar, ultrasonic, infrared or video bandwidth. These sensors provide data in different formats and one of the most challenging tasks is to correlate measurements into a unique representation viable for processing. Images captured by cameras are represented by pixels, which must be converted into distances expressed in a meaningful world coordinate system. Aiming for rapid and energy efficient 3D reconstruction we make an ample overview of state-of-the-art in hardware optimized solutions for image rectification and matching. We identify the algorithmic models commonly used in the literature and we analyze a consistent list of implementations for various platforms, while considering for possible accuracy flows, their speed, the computational performance and the energy consumption. We contribute with our original architectures implementing image rectification and matching in FPGA. For image rectification we developed an original caching technique which improves the access to image memory. The image matching entails optimization solutions capable to reduce the amount of allocated hardware.

The paper is organized as follows. In Section II we present the analytical model for image rectification and the classes of algorithms for image matching used by a majority of implementations proposed in the literature. In Section III we make a thorough study of existing solutions for image rectification and for image matching, which entail optimization strategies for hardware acceleration. We also refer to our proposed solutions and we initiate a parallel analysis to properly identify the contributing elements. Section IV offers a brief list of conclusions.

II. Standard methods for image rectification and matching

The reconstruction of the surrounding scene, from images captured by cameras at different view angles and positions, has raised the problem of correlating projections associated to the same point in the scene. Referring to a point in one image this process involves an exhaustive search in the other image. Extending to entire image the problem reaches exponential scale. Using the properties of the epipolar constraint for stereo cameras the search space can be restricted to the epipolar line. For a stereo configuration the epipolar lines intersect into a unique point, called the epipolar point, whose position is strictly influenced by the intrinsic parameters (focal distance, skew, principal point) and by the relative extrinsic parameters of the cameras (rotation and translation). For typical stereovision systems in automotive industry the epipolar points lye outside the images. When the epipolar points corresponding to each camera are stretched to infinity the epipolar lines become parallel. If their orientation is horizontal or vertical in both images the pair of cameras align in canonical configuration. In such case the corresponding image projections will always encounter similar position on one of the image axes. This property greatly simplifies the correlation process for several reasons:

- The computation of the epipolar lines is no longer necessary.
- The support window used by matching techniques requires no correction, otherwise used to cancel the cameras' relative orientation.
- When searching for correlations data redundancy can be exploited for reutilization and fast optimization.

A. Image rectification for calibrated systems

The canonical configuration is an ideal case of epipolar constraint not met in practice. The common solution proposed by the scientific community is the process of image rectification which makes the epipolar lines coincide with the scanlines. The images will appear as captured by a canonical configuration obtained through rotating the original cameras while keeping the optical centers fixed [5]. It resumes to computing the rectifying transformations and the Perspective Projection Matrices (PPM) characterizing the new system. Since most stereo matching algorithms assume horizontal epipolar alignment, image rectification became prerequisite for stereo correlation.

Considering the Pinhole model for camera capture a point $\mathrm{X} = [x, y, z]^T$ in the scene projects in the image at coordinates (u, v) according to the following equation:

$$[u, v, 1]^T = (1/\lambda) \cdot \mathrm{A} \cdot (\mathrm{R} \cdot \mathrm{X} + \mathrm{t}), \quad (1)$$

where A is the matrix of intrinsic parameters, R is the camera rotation matrix, t is the translation vector and λ is a scaling factor. A , R and t , respectively are estimated through specialized camera calibration techniques. If the intrinsic parameters A' and the rotation matrix R' are defined for epipolar alignment [6] it is possible to express a mapping from original image coordinates to rectified image coordinates as:

$$[u', v', 1]^T = (1/\lambda') \cdot \mathrm{A}' \cdot \mathrm{R}' \cdot \mathrm{R}^{-1} \cdot \mathrm{A}^{-1} \cdot [u, v, 1]^T. \quad (2)$$

The inverse relation from rectified image coordinates to original image coordinates is called inverse mapping and allows integration of distortion parameters [7] as follows:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathrm{A} \cdot \begin{bmatrix} \hat{x} + \hat{x} m_1 r^2 + p_1(r^2 + 2\hat{x}^2) + 2p_2\hat{x}\hat{y} \\ \hat{y} + \hat{y} m_1 r^2 + p_2(r^2 + 2\hat{y}^2) + 2p_1\hat{x}\hat{y} \\ 1 \end{bmatrix}, \quad (3)$$

where m, p_1, p_2 represent distortion parameters obtained from calibration, $r = \sqrt{\hat{x}^2 + \hat{y}^2}$ and

$$[\hat{x}, \hat{y}, 1]^T = \lambda' \cdot \mathrm{R} \cdot \mathrm{R}'^{-1} \cdot \mathrm{A}'^{-1} \cdot [u, v, 1]^T. \quad (4)$$

Using either direct mapping or inverse mapping, it is possible to generate the rectified image from pixels of original image. If sub-pixel precision is required the rectification can adopt interpolation of pixels from close vicinity.

B. Techniques for dense disparity computation

Depth information is critical for environment perception in autonomous systems. Stereo reconstruction is mostly affected by the accuracy of stereo matching algorithms looking for pixel correspondences in the captured frames. The displacement between matched points (disparity) is directly inferred in the depth estimation. There are three classes of matching algorithms delimited by their characteristics, accuracy and speed [8]: local, global and semi-global.

Global approaches define an energy function as summation of a data term and a smoothness term. The data term sums the pixel-wise matching costs. Complex forms of cost aggregation are not necessary. The smoothness term is mostly based on piecewise disparity localization and can be weighted according to prior segmentation results. Other possible terms could be used for penalizing occlusions, treating visibility or enforcing left/right symmetry. There are several strategies to search for the disparity map that minimizes the global energy. Methods based on dynamic programming usually perform 1D scanline optimizations, but encounter negative streaking effects. This was partially solved with tree-based variants [9]. The alternative solution was to use more complex algorithms such as graph-cuts [10] or belief propagation, which perform direct 2D optimization. The layered approach in [11] integrated information from image segmentation: using pixel-wise classification several plane models were defined in disparity space and were further refined iteratively. Another possibility was to infer disparity gradient into the energy function as representation of local surface orientation [12]. A patch-based approach [13] attached plane coefficients to pixels and performed spatial, view and temporal propagation for several consecutive iterations. Even if highly appreciated for accuracy, these methods suffer from slow processing.

Local block-matching algorithms use a restricted area of interest around each point to compare information from both images. The reduced support and the absence of global constraints is more likely to provide errors in featureless regions or repetitive patterns, but favors real-time performance and small memory footprint, which facilitate the run on embedded and mobile devices. With increased window size, the number of mismatches is reduced, unfortunately the detection rate at object boundaries is compromised. Usually, the support neighborhood takes the form of a rectangle [14] which favors frontal parallel surfaces. Allot of effort has been invested into adaptation for slanted surfaces [15]. Various matching metrics were proposed to compute the matching cost [16]. Cost aggregation based on single-block or multi-block computation [17] was implemented for smoothing. Since large aggregation blocks reduce mismatching, but also negatively influence object borders, adaptive and shifting windows were proposed to tackle the problem. Usually, a final disparity refinement is adopted to remove the peaks, to check the consistency or to fill the invalid gaps.

An alternative to previous solutions was proposed by Semi-Global Matching (SGM) approaches with results similar to most expensive methods, but allowing for real-time performance when implemented for smaller images or specialized hardware accelerator devices [18]. It follows dynamic programming, but avoids inappropriate behavior by approximating 2D energy optimization through minimizing on several 1D scanlines oriented at different angles, in polynomial time. The number of proposed scanlines ranges from four [19] to sixteen [20]. Correlation costs and smoothness constitute the basic elements of energy computation. Smoothness was expressed as summation of a regular penalty P_1 for small disparity discontinuities, like changes in object shape, and a higher penalty P_2 for the larger discontinuities found at border crossing. For a given disparity map D and the set of pixels S the general expression [21] of the energy function is:

$$E(D) = \sum_{\mathrm{p}\in S}\left[C\left(\mathrm{p}, D_{\mathrm{p}}\right) + \sum_{\mathrm{q}\in N_{\mathrm{p}}}\left(P_1 \cdot T\left[\left|D_{\mathrm{p}} - D_{\mathrm{q}}\right| = 1\right] + \right.\right.$$
$$\left.\left. + P_2 \cdot T\left[\left|D_{\mathrm{p}} - D_{\mathrm{q}}\right| > 1\right]\right)\right], \tag{5}$$

where N_{p} represents the set of all neighbors on the direction of the scanlines, $C(\cdot,\cdot)$ is the cost function and operator $T[\cdot]$ equals 1 or 0 depending on the argument value, if it is true or false, respectively. A third penalty P_3 based on second order prior along the scanline is suggested [22] to favor smooth transitions. Its values is added to P_1 and P_2. Considering a scanline direction represented by the vector r, the intermediary elements of the cost volume minimizing the energy will be optimized by the following recurrent expression:

$$C_{\mathrm{r}}(\mathrm{p},\Delta) = C(\mathrm{p},\Delta) + \min\Big\{C_{\mathrm{r}}(\mathrm{p}-\mathrm{r},\Delta), C_{\mathrm{r}}(\mathrm{p}-\mathrm{r},\Delta-1) +$$
$$+P_1, C_{\mathrm{r}}(\mathrm{p}-\mathrm{r},\Delta+1) + P_1, \min_d C_{\mathrm{r}}(\mathrm{p}-\mathrm{r},d) + P_2\Big\} -$$
$$-\min_d C_{\mathrm{r}}(\mathrm{p}-\mathrm{r},d). \tag{6}$$

The final cost will be the sum for the entire set of scanlines:

$$C'(\mathrm{p},\Delta) = \sum_{\mathrm{r}} C_{\mathrm{r}}(\mathrm{p},\Delta). \tag{7}$$

For outdoor scenes usually generating slight decalibrations of the system the preferred cost metric is Census Transform calculated on dense or sparse masks (Fig. 1) for the support window. It is invariant to luminosity and contrast differences. It shows robust behavior in difficult conditions or presence of radiometric errors common to production systems. A slight variation, the Center-Symmetric Census Transform [23] has proved more reliable because it does not suffer from noisy central pixels.

The latest efforts are focused on integrating deep-learning techniques at different layers of the traditional pipeline [24]. A special interest is given to end-to-end Convolutional Neural Networks (CNN) [26] which can generate accurate disparity maps directly from image pairs.

III. Implementations Optimized for Hardware Acceleration

Recent innovations in stereovision confront with an emerging need to process the large amount of imagery captured by high-definition cameras. Usually, the applications are deployed on mobile devices which are required to be energy efficient and fast. Taking advantage of certain patterns of memory access exhibited by image processing applications it is possible to estimate the required level of connectivity, scheduling and memory usage, thus being able to differentiate between suitable computing platforms such as DSP, Application Specific Integrated Circuits (ASIC), FPGA, GPU or Central Processing Units (CPU). The next two sections study the implementation of image rectification and matching from the perspective of optimization with specialized hardware acceleration.

A. Hardware solutions for image rectification

The FPGA architecture in [27] performed noise removal, image rectification with bilinear interpolation and matching. For the image rectification part pixels were stored in line buffer units composed of four separate buffers corresponding to odd and even rows and columns. This allowed the simultaneous access to all neighboring pixels of a sub-pixel location. Line buffers were implemented as circulars buffers. The amount of memory allocated made possible to accept up to 64 pixels vertical displacement. The inverse mapping from rectified images to original images was computed on-the-fly. The linear functions at the base used coefficients computed from calibration data and stored within configurable registers. Multiplication and division operations required for bilinear interpolation were implemented with DSP units. The rectification unit was able to provide one rectified image pixel per clock cycle and was implemented on Xilinx Virtex-5 FPGA. The system entailing image rectification and matching was capable to run at 37 fps on 640x480 images.

Image rectification optimized for GPU was proposed in [29]. The algorithm was implemented according to model specifications in [5]. The process was based on inverse mapping Look-Up Tables (LUT) computed for each image. Their contents were computed once, at system startup, based on preliminary calibration data. The option to use precomputed LUT allowed the integration of rectification with distortion removal, down-sampling and ground plane stereo, in a unique image warping. The correspondences were computed with sub-pixel precision, therefore pixel intensities in the rectified image were obtained using bilinear interpolation of the neighboring values. For each captured frame the images were sent to the GPU texture memory which owned a cache facilitating accelerated access to neighboring pixels. Additionally, it included hardware support for fast bilinear interpolation. The processing time was 0.065 ms for 512x383 images and 0.179 ms for 1024x768 images, without considering the transfer time between CPU and GPU.

The FPGA architecture in [30] was capable to perform non-linear image transformation based on finite differences in conjunction with polynomial approximation. Such

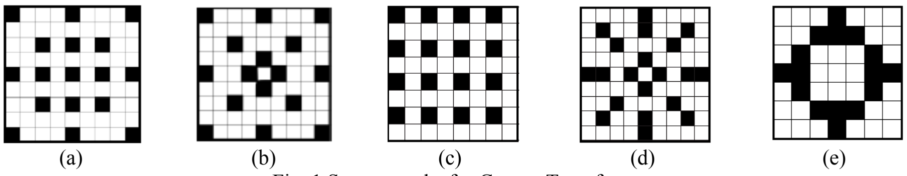
(a) (b) (c) (d) (e)

Fig. 1 Sparse masks for Census Transform

transformations could implement image undistortion and image rectification altogether, with some level of precision loss. The degree of the polynomial functions and the number of bits used for accuracy were customized for expected precision and reduced amount of allocated resources. In practical examples four degree polynomials were able to provide errors less than half a pixel. The implementation on Xilinx Spartan-3E FPGA achieved real-time performance with latency of few image lines.

Following the same principle, but using matrix operations instead of polynomial functions, the implementation in [31] was deployed on Xilinx Virtex 4 FPGA (model XC4VLX60) and reached 367 fps and 120 fps for 640x480 and 1280x720 images, respectively. Based on initial calibration data, images were rectified according to epipolar alignment along with radial and tangential distortion removal. At the cost of 0.13 ms latency, an image buffer covering several image rows was implemented in BRAM units to assure consistent flow between input image and rectification circuitry.

Stereo rectification for full High-Definition images (1920x1080) was implemented on GPU based on OpenGL objects such as frame buffer and vertex buffer [32]. The component was running real-time at 70 fps. The strategy was based on precomputed LUT with inverse mapping. At the cost of memory bandwidth usage and storage consumption, the adoption of LUT saved the system from complex computational efforts.

Since the size of rectification LUTs is strictly dependent on image resolution and fractional precision it is very common to encounter situations not comfortable with the amount of on-chip memory (BlockRAM units) available on mid-range FPGA chips. Therefore, external memory (DDR SDRAM, SRAM) is usually preferred by most architectures, but has the disadvantage of higher latency and complex communication protocol. For this reason the solution in [33] proposed a forward mapping scheme with offline compression algorithms to reduce the typical size of LUT to a reasonable storage capacity met in regular low-cost FPGAs. The additional amount of hardware needed for decompression was also negligible. Two LUTs were allocated per image, encoding horizontal and vertical coordinates separately. The fractional part was rounded to nearest available pixel, thus eliminating the possibility for sub-pixel interpolation. The mapping used one-to-one pairing with the sole exception that one pixel in original image could target two adjacent vertical pixels in the rectified image, if the direct mapping was linked to the pixel on top. Such situations were marked as double-targets and the measure was taken to fill some of the voids (data loss consisting in unmapped pixels due to one-to-one mapping restrictions). The LUT was expressed as a set of breakpoints marking double-targets or coordinate changes of maximum one pixel. Each row and each column was reserved a maximum number of possible breakpoints. Such limits were vulnerable to higher distortions, cameras misalignment or combination with down-sampling. The images were parsed horizontally, from top to bottom. Pixels from rectified image with a corresponding position outside the original image were marked as non-targets and were filled with mapped-data from close-surroundings. Due to some latency in the hardware design, each breakpoint from the vertical LUT should have encoded no less than four consecutive pixels for correct decompression. These fine details suggested several sources of possible inconsistency in the resulting rectified images. The proposed architecture implemented on Virtex 5 FPGA (model XCUVP-110T) was able to operate at 347 fps with 1024x768 images and without need for external memory. The LUT was coded as ROM using a consistent amount of BlockRAM. Additionally, the double-target encoding required a minimal amount of buffering on the output. The outcome was less available storage on-chip for the task of disparity estimation. Some limitations were removed with the improved version proposed later for the same FPGA model [34]. Extending the double-target condition from a mapped pixel to all its unmapped neighbors, except the ones at the left and right, the number of voids (data loss) was reduced, and also the number of successive breakpoints, which decreased the risk caused by latency. Even if there was still a small risk for possible breakpoints located at consecutive pixels, the situation was handled by keeping active a constant buffer of the most recent six consecutive breakpoints. Additionally, the one pixel limit imposed on row or column change was extended to two pixels, in order to gain more flexibility. The architecture achieved a supplemental 2.5 times internal memory consumption, while providing correction capabilities for greater cameras misalignment and distortion at 269 fps.

When leveling up the image resolution for cameras with a wide field of view (at 110^0 the distortion is increased) the coordinate differentials at breakpoints may well exceed a two pixels jump. In [35] the breakpoint structure was reconfigured to extend the stepping interval to [-8, 7]. The latency of the compressed LUT memory was reduced to only two clock cycles and the critical case of consecutive breakpoints was solved for one pixel differentials only: as indicated by a special field in the breakpoint descriptor, the current row was incremented/decremented on-the-fly, if needed for the next pixel, without allocating another breakpoint. Additionally, the size of the LUT memory was significantly reduced through variable breakpoint indexing. The fixed number of breakpoints allowed for rows and columns was replaced with a list-like LUT implementation. Indices marking the locations of the first breakpoints in each row or column were stored in a separate array. Hence the breakpoints' memory was completely filled without counting for possible holes. The buffer size reduction was around 58-65% compared to the model in [34] and the implementation on Altera Cyclone V E FPGA (model 5CEBA7F27) supported real-time operation at 50 fps for 1920x1080 resolution.

For situations necessary to avoid memory allocation for LUT the solution in [36] proposed on-the-fly computation of inverse mapping equations with double precision floating point units. The set of operations was based on calibration parameters obtained offline. In the initial step, rectified image coordinates were mapped into the original camera frame. A secondary step performed distortion removal and mapping to original image coordinates. A memory buffer containing several image rows was implemented for efficient access to neighboring pixels after remapping. Bilinear interpolation was used for sub-pixel precision. Real-time was achieved on a Xilinx Virtex 7 FPGA board (model XC7V2000T) at 1280x720 resolution.

Several common lossless methods were modified to enhance performance in an adaptive combination [37]. The solution used forward mapping without sub-pixel interpolation and implemented separate decoding hardware and LUTs for horizontal and vertical coordinates. An initial differential coding, with horizontal direction on rows and vertical direction for first column, was used to transform coordinates into data of low variation, low entropy and reduced bit-length for symbol representation. The obtained form was suitable for optimal compression with either Huffman coding [38] or a modified version or Run-Length Encoding (RLE). The best was chosen for each dataset and camera configuration. The efficiency was analyzed offline by computing the average Inverse Compression Ratio and its variation. Smaller values indicated better performance. In a general term, the Huffman coding, which is based on entropy, demonstrated high stability of results by small variations of Inverse Compression Ratio. The RLE was more optimal due to smaller average of Inverse Compression Ratio, but was less stable. The improvement proposed for RLE aimed to generate breakpoints only when differential codes were passing from the most frequent symbol to any other symbol. A typical breakpoint format was 12-bit wide and contained the less-frequent symbol on four bits (one for sign and three for magnitude) paired with the number of previous occurrences counted for the most frequent symbol. The counter section was limited to seven bits in the standard breakpoint format. It was extended to 11 bits within a second breakpoint format, specially designed for the most frequent symbol. In both cases the 12th bit indicated the model of the breakpoint. Consecutive repetition of breakpoints with extended counter section allowed for unlimited occurrences to be coded for the most-frequent symbol. The adaptive combination reached both stability and performance with an average 7.19% compression ratio, ranging from 1.54% to 11.03%, depending on tested dataset. The implementation on Xilinx Virtex5-VLX330 FPGA reached 261 fps for 1280x720 images.

The inverse mapping approach in [39] proposed LUT size reduction through down-sampling. Horizontal and vertical coordinates of the original map were sampled and stored in low resolution LUTs. When generating the rectified image, normal resolution coordinates were generated from sampled coordinates through separate linear interpolation on horizontal and vertical axes. The solution had an obvious disadvantage due to the mapping accuracy loss resulted from extracting the true position from down-sampled data. For a certain range of configurations the loss was in a reasonable domain. Overall, three interpolation units were required: two for interpolating the horizontal and vertical mapping and a third for interpolating neighboring pixel intensities when estimating the rectified pixel value.

Following the model elaborated in [5] we proposed a pipeline architecture for FPGA [40]. Given its LUT-based implementation for inverse mapping combined with bilinear interpolation we were capable to integrate any of the following operations in a single image warping: rectification, undistortion, down-sampling and ground-plane stereo correction. The transformation is lossless and the semi-sequential access to image pixels in memory is reduced by implementing an original cache model. The implementation on Xilinx VirtexE-V600EFG680 achieved approximately 85 fps for 640x512 images. When ported on Xilinx Virtex4-XC4VLX160 the speed increased to 125fps. Both cases were limited by slow transfer rates with the local memory and with the CPU, which are vendor specific.

B. *Image matching implementations with specialized hardware support*

An early hardware solution [41] providing near real-time dense stereo reconstruction was a 5-camera system calculating 200x200 depth maps at 30 fps. The speed was halved with sub-pixel disparity interpolation. The disparity range was 30 and the fractional precision was supported on three bits. The preprocessing phase performed Laplacian of Gaussian (LoG) and non-linear compression based on histogram equalization. It was followed by image rectification with LUT stored in memory and by interpolation. The matching solution used Sum of Absolute Differences (SAD) as cost metric. The aggregation was performed over results from several camera baselines. The resulting sum over SAD was rewritten to allow hardware optimization. The disparities were searched at minimum costs and refined with parabola sub-pixel interpolation over neighboring costs:

$$f_{para}(\mathrm{p},\Delta_{min}) = \frac{1}{2}\cdot\frac{C(\mathrm{p},\Delta_{min}+1)-C(\mathrm{p},\Delta_{min}-1)}{2C(\mathrm{p},\Delta_{min})-[C(\mathrm{p},\Delta_{min}+1)+C(\mathrm{p},\Delta_{min}-1)]}. \quad (8)$$

The infrastructure was based on off-the-shelf components including ROM, RAM, pipeline registers, convolvers, digitizers and Arithmetic Logic Units (ALU). They were connected through a Versa Module Europa bus (VMEbus) and controlled by a VxWorks real-time processor. The C40 DSP array was self-proprietary and was used for interpolation.

A system developed on Programmable And Reconfigurable Tool Set computer (PARTS) generated 320x240x24 maps at 42 fps and 22.5 W power consumption [42]. The cost computation was based on Census Transform and Hamming distance. Minimum selection was used to estimate disparities. The entire architecture was developed on 16 Xilinx 4025 FPGA devices connected in a partially torus-like configuration. They had 16 tightly coupled SRAM units.

The small vision module proposed in [43] was implemented on DSP units (model ADSP2181) running at 33 MHz and consuming 600 mW. The maps were sized 160x120x32. The speed performance was only 6 fps, however real-time was achieved when ported on a higher class DSP (model TMS320C60x) running at 200 MHz. The algorithm performed LoG filtering, image rectification, local area normalization, correlation based on SAD, disparity computation through winner-takes-all and post-filtering based on texture confidence and left-right consistency. Sub-pixel interpolation in the final stage provided fractional precision on two bits.

Tyzx Inc. has developed a stereovision machine on ASIC [44] capable of 512x480x52 dense disparity maps at 200 fps. The fast implementation with dense Census Transform allowed 52 simultaneous comparisons when searching for minimum cost disparity. The best match was located with five bits sub-pixel precision. Total power consumption was below 1 W. Initial image rectification was also performed in hardware. Several configurations were provided for different resolutions, with speed being influenced by their frame rate capabilities. In later proposal [45] the algorithm was ported on a mixture hardware of ASIC, FPGA, DSP and PowerPC.

The miniature trinocular system in [46] has been built on the 60 MHz Xilinx XC2V2000 FPGA chip with tightly coupled local SRAM. The performance was 30 fps on a 640x480x64 configuration providing two bits for fractional precision. The speed raised to 120 fps with smaller 320x240 resolution maps. Internally, it performed epipolar rectification, vertical Gaussian and Laplacian filtering, data compression and matching based on Sum of SAD, minimum cost selection for disparity estimation and sub-pixel interpolation with parabola (8). At the expense of using more memory, cost aggregation was optimized by accumulating the costs on columns and rows for data reuse at consecutive disparities. The cameras were fixed to the device with horizontal and vertical baselines.

Constrained by reduced resources in early FPGAs the pipeline architecture proposed in [47] considered replacing simultaneous cost computation for the complete disparity range, with hardware support for a pair of shifting local ranges, called correlation windows. One window was centered on the disparity found in the previous frame in order to exploit temporal information – at higher rates it was estimated the disparities would not suffer major changes at consecutive frames. The other window was continuously sliding on the same row according to flexible customized rules. The optimal disparity suggested by the second correlation window had higher priority if the corresponding cost was better than the best found for the first widow. This way significant changes in disparity could be handled, if not immediately, then within several consecutive frames. The costs were computed in parallel for both windows, at three different scales with down-sampling. The scales used were 1, 2 and 4 and the associated widows were sized 9, 5 and 3 pixels, respectively. The images were initially rectified with bicubic polynomials, in order to map original coordinates to rectified coordinates. The coefficients of the polynomial functions were computed offline using calibration data. A total of 64 scanlines were buffered for each image to assure safe rectification. After rectification the images passed through a Gaussian Pyramid filter and were down-sampled. Three quadrature-pair filters were applied at each scale, oriented at -45^0, 0^0 and 45^0. The results were used to compute Local-Weighted Phase Correlation (LWPC) costs as in [48]. Upon up-sampling from coarser scales using quadrature interpolation in the disparity domain and linear interpolation in horizontal domain, the costs sourced by all scales and orientations were summed up for disparity selection stage. With two similar architectures running concurrently to generate disparities for each image as reference, it was possible to detect mismatches and occlusions through left-right consistency check. The system was able to work at 30 fps on 640x480 images while handling a range of 128 disparities. It was implemented on a platform consisting of four Altera Stratix S80 FPGA consuming 500 mW each.

The pipeline architecture in [49] is a hardware optimized version of the dynamic programming-based maximum likelihood approach introduced in [50]. The implementation on FPGA applied two basic rules: uniqueness assumption, stating each pixel in the left image has one corresponding pixel in the right image, and monotonic ordering assumption over correct matches. The algorithm tried to find an optimal path through a match matrix of indicators highlighting whether pairs of pixels on the same scanline represent occlusions or not. The cost metric was based on squared differences and possible occlusions were penalized. By changing the original code and reducing the matrix size from an entire image to only two scanlines processed in parallel it was possible to achieve a theoretical speed of 36.33 fps on 512x512 images with 16 disparities. A later optimized implementation ran at 64 fps on VGA images with 128 disparities [51].

The solutions so far are representative for the class of local block matching algorithms. Nevertheless, the interest for the more sophisticated SGM approaches has raised with several implementations in FPGA, CPU or GPU devices.

The classic version proposed in [21] and later improved in [20], a hierarchical SGM with Mutual Information, has been optimized for nVidia GeForce 8800 ULTRA GPU [52]. A total of five levels were used, the images being halved for each level. The costs were optimized on eight scanlines (6, 7) with second penalty being adapted by image gradient. After estimating disparities at minimum cost, they were interpolated with parabola (8) for sub-pixel resolution and were smoothed with median filter. Validation was conducted with left-right consistency check on the same cost volume. Disparities found at coarse levels were up-sampled as initial guess for finer resolutions. It was noticed the speed performance did not scale well at low resolutions. Implementation was mainly based on OpenGL/Cg rendering commands. Only the computation for Mutual Information matching table was coded separately on GPU in a vertex shader program. The performance for 640x480 images with 128 disparities reached 4.2 fps and 13 fps for half resolution with half disparity range.

Replacing Mutual Information with Absolute Difference and eliminating the hierarchical approach, the solution in [53] introduced a Cg-based implementation on nVidia GeForce 7900 GTX GPU. The costs were optimized on eight scanlines by accumulation in an RGBA sweep plane texture. A shader pass extracted disparity information and two shaders performed left-right consistency check (based on same cost volume) to detect mismatches and occlusions, which were finally removed through hole filling. Tests on 320x240 images with 64 disparities achieved 8 fps and 13 fps for half resolution and half disparity.

SGM with costs based on the sampling-insensitive absolute difference [54] was optimized for Compute Unified Device Architecture (CUDA) in [55]. A tile of threads was created to avoid memory bottleneck during cost computation. The optimization of cost volume was performed on eight scanlines according to (6, 7), with adaptive penalty P_2. Minimum cost search for disparity estimation was optimized using a naive scan-tree technique. Without left-right consistency check, hole filling and disparity smoothing, the implementation on nVidia Quadro FX 5600 GPU was able to process 450x375 images with 64 disparities at 5.85 fps.

The FPGA solution in [56] was implemented on Xilinx Spartan-3a 3400DSP and was able to generate 680x400x128 dense disparity maps at 25 fps and a power consumption beneath 3 W. Input images were rectified internally and filtered with 3x3 Gaussian. The architecture was implemented to generate two sequential 340x200 disparity maps. One was for images down-sampled with a factor of two and the other for a specified normal resolution region. Only 64 disparities were accounted, but values up to 128 were reached after remapping the initial disparity map to original resolution. For the normal resolution map, values larger than 64 indicated by the other map would replace current results. The matching cost was based on Zero-mean SAD (ZSAD) implemented in parallel for all possible disparities. The SGM optimization was performed in two scans for each map (four in total): one from top-left to bottom-right and a second scan backwards. Each time, the four neighbors left behind were evaluated according to (6) and their summation (7) was stored in external memories. Current scanline costs were saved in local buffers for fast computation. Through pipelining, all 64 disparities were evaluated in one clock cycle. Minimum cost and its neighbors were selected for disparity computation and sub-pixel interpolation with symmetric-V:

$$f_{symV}(\mathrm{p},\Delta_{min}) = \frac{1}{2}\frac{C(\mathrm{p},\Delta_{min}+1)-C(\mathrm{p},\Delta_{min}-1)}{C(\mathrm{p},\Delta_{min})-max\{C(\mathrm{p},\Delta_{min}+1),C(\mathrm{p},\Delta_{min}-1)\}}. \quad (9)$$

A 3x3 median filter was used for refinement and finally, left-right consistency check was performed by re-computing disparities with the same cost volume. Implementation on a

Virtex4 class FPGA allowed an additional symmetric SGM branch implementation for full consistency check, without affecting frame rate. In the CPU implementation [57] the ZSAD was replaced by Census Transform for a dense 5x5 window with 32-bit Hamming Distance. The images were downscaled with a factor of two and four, resulting in implicit 10x10 and 20x20 Census masks. The matching was performed three times in a coarse-to-fine manner. Normal resolution map used 16 disparities, half and quarter resolution maps used 16 and 32 disparities, respectively. The half resolution map covered the disparity range from 16 to 32 and the quarter resolution map covered the rest up to 128. Census computation was parallelized on several cores with line-wise OpenMP. The costs were optimized on eight scanlines. The images were scanned top-left to bottom-right and backwards. Each time four scanlines were integrated in the cost volume optimization with (6, 7). Considering one path, each ray was independently computed through OpenMP parallelization. Moreover, using 128 bit registers with SSE and SSSE optimization it was possible to run this step for 16 concurrent disparities. Finally, disparities were searched at minimum cost and were interpolated for sub-pixel accuracy with Symmetric-V (9), followed by median filtering and consistency check using the same cost volume. The resulting maps were denser, but suffered some blocky effect from down-sampling. Implemented on an Intel Core i7-975ex CPU consuming 130 W the system reached 14.5 fps for 640x320 images and 128 disparities.

A typical problem with FPGA solutions is the amount of external memory usage. Several buffering techniques were proposed in [27] to reduce memory bottleneck. The matching with SGM used Rank Transform as cost metric. The second penalty in (6) was adjusted for each pixel and each scanline according to the rank costs. The image was processed on a normal scan and a backward scan. Each scan optimized the costs on four scanlines, simultaneously. Multiple scanlines were processed in parallel. For each row the resulting optimized costs corresponding to each scanline were stored within three shift registers of different depth and passed to the next row as input. By cascading using such systolic array technique it was possible to process a slice of several scanlines in parallel. The costs resulting from the last scanline were stored in memory as input for the first row in subsequent slice. The disparity was estimated at minimum cost and validated according to uniqueness constraint. The same costs were used to estimate the paired image disparity. Finally, results were filtered with left-right consistency check and refined with median filtering. On a Xilinx Virtex-5 FPGA device input images of 640x480 pixels with 128 disparity range were processed at 37 fps.

A later implementation of SGM with Rank Transform was elaborated for GPU in [28]. Each part of the parallel implementation was carefully analyzed concerning the limitations for instruction and memory throughput. A highly optimized scheme was defined for mapping into available resources. Optimal configurations were set for Rank Transform, median filtering, matching cost computation and scanline optimization. The time for 1024x768x128 disparity maps on nVidia Tesla C2050 GPU was 36 ms (27 fps).

An efficient implementation of SGM and Census Transform [58] took advantage of the two levels of parallelism available in CUDA: coarse-grain parallelism which allows no inter-thread data communication and fine-grain parallelism for sharing data. Each processing step of the pipeline was implemented in a different kernel. The communication was performed using the cost volume. A sparse support window of 9x9 was adopted for cost computation (Fig. 1b). Cost aggregation was employed on 5x5 neighboring blocks. The number of optimization scanlines was reduced to four. This had significant impact on speed improvement while decreasing memory bandwidth. Penalty P_1 was removed to reduce depth scattering at object borders and non-textured areas. Disparities were computed in a winner-takes-all manner. A sinusoidal interpolation functions was proposed for sub-pixel precision. Left-right consistency check, based on a single cost volume, was applied to detect occlusions, finally removed through refinement techniques. The processing speed on 512x383 images with 56 disparities was 52.6 fps for the low class GPU nVidia GeForce GTX-280 and 90.9 fps for the higher class nVidia GeForce GTX-480.

Another CUDA-based optimization for SGM was suggested in [59]. The cost metric was set to Absolute Difference. Eight cost optimization scanlines were used paired into four orientations. Each orientation was assigned a weight in the accumulation (7) and two pairs of penalties, each being used according to local image gradient. A total of 20 parameters were trained using covariance matrix adaptation evolution strategy. During training each set of parameters was validated by the errors obtained with respect to ground-truth data. After estimating disparities at minimum cost, left-right consistency check validated results and median filtering was applied for smoothing. Sub-pixel values were extracted with parabola interpolation (8). Running on nVidia GeForce GTX-480 GPU the algorithm achieved a speed performance of 11.7 fps for 640x480x64 dense disparity maps.

Instead of allocating memory to save the entire cost volume needed for scanline aggregation according to (8), the FPGA solution proposed in [60] suggests having the disparities searched at minimum cost found between all optimization scanlines. Initially, the image was scanned from top-left to bottom-right and optimization was performed on four scanlines. For each scanline the minimum cost was saved in memory along with its index and its neighboring costs that might be used for sub-pixel interpolation. This represents four values for each scanline and 16 values in total. Also, two intermediary locations were added to store the global minimum and its index over all four scanlines. It frees the 16 locations for further use. This amounts to 18 memory locations for each pixel. Practically, the required memory footprint was reduced since classic SGM would have required a number of memory locations equal to the number of disparities. In a subsequent image scan performed backwards the other four scanlines were optimized similarly using the 16 now available memory locations. A third scan was required to choose the right minimum between intermediary values and results from backward scan. Experiments revealed such changes in minimum cost selection were having minimal influence at accuracy level. The cost metric adopted was Census Transform. Implementation on Xilinx Virtex 5 FPGA (model XC5VSX95T) revealed a processing speed of 10 fps on 1024x1024 images with 64 disparities and 33 fps on 640x480 images. When ported on mobile platform equipped with Xilinx Spartan 6 LX150 FPGAs the performance was 25 fps on 512x512 images and same disparity range. Optionally, disparities could have been validated with left-right consistency check at the cost of additional hardware and double the processing time.

A CPU based implementation of SGM enabled parallelization with SIMD instruction set [61]. The Center-Symmetric Census Transform [23] costs were processed in parallel for eight 12-bit pixels using the 128 bit SSE registers. The computation of Hamming Distance was optimized by implementing parallel 4-bit LUTs in logic (PSHUFB instruction from SSSE3), thus avoiding memory bandwidth usage. Some significant improvement was obtained through disparity space compression. The disparity range was subsampled to lower the size of the cost volume. Starting from a fixed disparity (usually the middle of total disparity range) and up, the costs were computed in steps of two or four. The images were not sub-sampled. This affects only close-range reconstruction. Consequently, the algorithm runs a compressed cost volume, however final disparities need to be remapped to original range. Furthermore, the images were divided in several horizontal stripes and the cost volume was generated in parallel for each stripe. To minimize degradation suffered in the scanline accumulation step each stripe was added a small sized upper and lower border, not counted in the final cost volume. Having costs represented on 16 bits it was possible to compute (6) for eight disparities in parallel. Regarding the accumulation in (7) a number of eight scanlines were accounted. Inspired by [57] parallel scanline computation was considered without much success for speedup. Therefore the image was initially scanned from top-left to bottom-right to accumulate four of the scanlines. In a subsequent backwards image scan the rest of the scanlines were added. The minimum cost computation for disparity selection was optimized by computing the minimum for eight positions at once using PHMINPOSUW instruction from SSE4. A second minimum was computed to invalidate a weak disparity when necessary. The same cost volume was used when computing the disparity image from left to right and right to left. Disparities were extracted at sub-pixel level using symmetric-V interpolation (9). They were further refined using median filtering and validated with left-right check. The algorithm tested on a high performance CPU family (Intel Core i7 i7-4960HQ) at 47 W maximum power consumption reached 14 fps on 640x480 images with 128 disparities and two step compression for the upper half range. An improvement of 2.1 fps was achieved when using four step disparity compression. Without compression the cycle per frame was 84 ms (11.9 fps).

Recent efforts were concentrated around neural network implementations. SGM was improved in [25] by having costs computed with a fast CNN. The cost volume was optimized on four scanlines and penalties were adjusted based on image gradient. Disparities were extracted at minimum cost. Sub-pixel accuracy was achieved using parabola interpolation (8) and was further refined with median and bilateral filters. The implementation on one of the latest class GPU (nVidia Titan X) achieved around 16.6 fps for 320x240 images and 32 disparities. For 1240x376 images with 228 disparities the performance decreased to 1.25 fps.

When analyzing a vast time frame from the past the race for efficient power consumption was clearly won on FPGA side until recent launch of GPU devices targeting the mobile and automotive market. The implementation in [62] proposed a highly parallel scheme for SGM with Center-Symmetric Census Transform optimized for nVidia Tegra X1 GPU. According to vendor specifications the average Thermal Design Power (TDP) is 10 W. Taking advantage of the shared memory capabilities it was adopted a 2D-tiled parallel scheme to generate bit streams associated to a dense 9x7 Census mask. The costs computed with Hamming Distance were parallelized on 1D tiles. The optimization of cost volume according to (6) was separately optimized for each scanline. Penalty P_2 was set constant. The aggregation (7) took place during the last scanline optimization along with minimum cost search for disparity estimation. The only post-processing step consisted in median filtering, while sub-pixel interpolation and left-right consistency check were missing. For 640x480 images and 128 disparities the implementation with eight cost optimization scanlines was running at 19 fps. Real-time was achieved (42 fps) when optimizing on four scanlines. A recommended 8-scanline implementation was possible within real-time frame only on the higher class nVidia Titan X GPU, running at 237 fps, with a power consumption of 250 W.

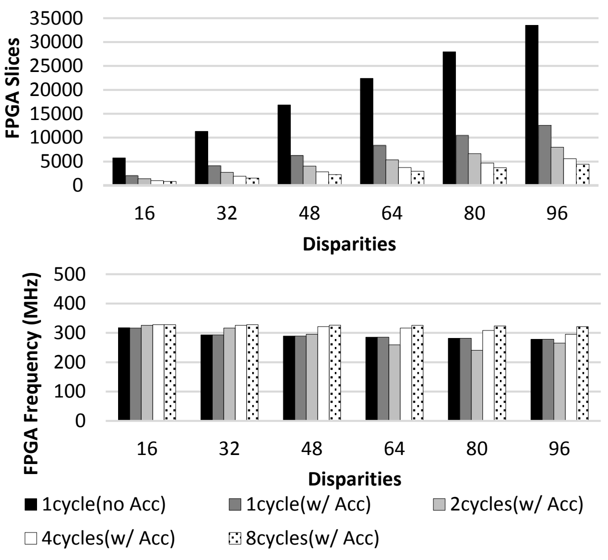

Fig. 2 Impact of the proposed optimizations on the Cost Volume Computation unit – internal running frequency is stable despite hardware reduction.

TABLE I. PERFORMANCE BY THE NUMBER OF CYCLES WHEN GENERATING 640X512X128 DISPARITY MAPS

Cycles	Speed [fps]	PDS [x10^6]	Power [W]	PDS/Watt [x10^6]
1	84	3541.3	4.468	792.6
2	82	3419.7	3.705	923
4	60	2521.5	3.409	739.7
8	39	1653.1	3.306	500
16	23	978.9	3.315	295.3

Our solution designed for FPGA [63] represents a pipeline architecture for local block matching. We proposed an original hardware cost optimization technique based on accumulating costs aggregated on columns of support window. Also, an original multiple cycle approach is adopted for further optimization of the cost volume computation unit (Fig. 2). The fractional precision is computed with the LUT-based method proposed in [64]. It is integrated in the pipeline using a fast and cost effective implementation that allocates only one internal memory unit (BlockRAM). The architecture running both image rectification and matching has achieved 82 fps for a two cycle customization providing 640x512x128 disparity maps. In this configuration the power dissipation was 3.7 W. TABLE I shows the results for implementing other single cycle and multiple cycle configurations. The PDS coefficient represents the number of pixels multiplied by the number of disparities computed per second. It is the result of multiplying the speed with the image resolution and with the disparity range, in order to measures the computational capability. TABLE II shows our 2-cycle implementation is second in computational performance and third in energy efficiency.

C. *Comparative remarks*

The amount of circuitry dedicated to high scale parallelization and pipelining, seconded by top classification in power efficient devices, makes the FPGA one of the most attractive solutions for hardware acceleration at the edge. We adopted this class of hardware to accelerate the computationally intensive image matching task for which rectification is a mandatory preprocessing step.

Image rectification solutions, adopting computation of the correspondences on-the-fly, require floating point arithmetical units to implement the complex equations. The mapping can be forward or inverse. The precision has to be carefully established, if the paired locations need sub-pixel approximation. For simplification and economy of allocated hardware, we decided to have the mapping computed offline and keep the paired image locations in a LUT that is stored into local memory at initialization. Given the necessary size, the memory is allocated on SRAM chips outside the FPGA.

When implementing rectification with sub-pixel location the forward mapping requires larger buffers to store the neighborhood for interpolation. Instead, the proposed solution implements inverse mapping, hence supplemental buffers are not needed because the interpolated pixels are always adjacent at known locations. However, an original caching technique is implemented to reduce the access to the memory storing the input images.

Solutions applying forward mapping had to remove fractional precision, in order to allow LUT compression that can fit the small memory distributed inside the FPGA. A one-to-one mapping may encounter gaps between pixels. With inverse mapping the gaps are eliminated, the precision is maintained and the image is completely rectified.

The reduction of the LUT size with down-sampling, followed by up-sampling of the rectified image to original resolution, encounters information loss. We use down-sampling only when the disparity map should be computed for lower resolutions, without further up-sampling.

There are implementations which separate rectification from undistortion, in distinct computations, hence they use additional hardware while increasing the latency. The rectification model we proposed can combine several preprocessing operations into a unique image warping.

Image matching encounters many implementations, which follow the algorithmic details characterizing different local or semi-global solutions. Not only they approach different classes

TABLE II. STEREO MATCHING SOLUTIONS IMPLEMENTED WITH SPECIALIZED HARDWARE (OUR SOLUTION IN LAST ROW)

Reference	Matching technique	Hardware platform	(Image size) x Disparities	Speed [fps]	PDS [x10^6]	PDS / Watt [x10^6]
Kanade et al. [41]	local with SAD	C40 DSP array, ROM, RAM, ALU	(200x200)x30	30	30	N/A
Woodfill et al. [42]	local with Census	16 x Xilinx 4025 FPGA	(320x240)x24	42	77.4	3.44
Konolige [43]	local with SAD	ADSP2181 DSP module	(160x120)x32	6	3.7	6.14
Woodfill et al. [44] Woodfill et al. [45]	local with Census local with Census	ASIC ASIC, FPGA, PowerPC	(512x480)x52	200	2555.9	2555.9
Jia et al. [46]	local with SAD	Xilinx XC2V2000 FPGA	(640*480)x64	30	589.8	N/A
Masrani et al. [47]	local with hierarchical LWPC	4 x Altera Stratix S80 FPGA	(640*480)x128	30	330	660
Rosenberg et al. [53]	SGM with Absolute Difference	nVidia GeForce 7900 GTX GPU	(320x240)x64 (160x120)x32	8 13	39.3 7.99	0.47 0.1
Sabihuddin et al. [49]	global with Squared Difference	FGPA	(512x512)x16	36.3	152.4	N/A
Sabihuddin et al. [51]	global with Squared Difference	FPGA	(640x480)x128	64	2516.6	N/A
Gibson et al. [55]	SGM with sampling-insensitive	nVidia Quadro FX 5600 GPU	(450x375)x64	5.9	63.7	0.37
Ernst et al. [52]	hierarchical SGM with Mutual Information	nVidia GeForce 8800 ULTRA GPU	(640x480)x128 (320x240)x64	4.2 13	188.7 73	1.08 0.42
Gehrig et al. [56]	SGM with ZSAD	Xilinx Spartan-3a 3400DSP FPGA	(680x400)x128	25	217.6	72.53
Gehrig et al. [57]	- SGM with Census - hierarchical SGM	Intel Core i7-975ex CPU	(640x320)x128 (640x320)x128	4.5 14.5	117 65.3	0.9 0.5
Haller et al. [58]	SGM with Census	nVidia GeForce GTX-280 GPU	(512x383)x56	52.6	577.6	2.45
Banz et al. [27]	SGM with Rank Transform	Xilinx Virtex 5 FPGA	(640x480)x128	37	1454.9	1454.9
Banz et al. [28]	SGM with Rank Transform	nVidia Tesla C2050 GPU	(1024x768)x128	27.8	2798.4	11.76
Pantilie et al. [29] Pantilie et al. [18]	SGM with Census	nVidia GeForce GTX-480 GPU	(512x383)x56	91	999.3	4
Buder [60]	SGM with Census	Xilinx Virtex 5 FPGA Xilinx Virtex 5 FPGA Spartan 6 LX150 FPGA	(640x480)x64 (1024x1024)x64 (512x512)x64	33 10 25	648.8 671.1 419.4	648.08 671.08 838.86
Michael et al. [59]	SGM with Absolute Difference	nVidia GeForce GTX-480 GPU	(640x480)x64	11.7	230	0.92
Spangenberg et al. [61]	SGM with Center-Symmetric Census	Intel Core i7 i7-4960HQ CPU	(640x480)x128	16.1	396.4	8.43
Zbontar et al. [25]	SGM with CNN	nVidia Titan X GPU	(320x240)x32 (1240x376)x228	16.7 1.25	41 132.9	0.16 0.53
Juarez et al. [62]	SGM with Center-Symmetric Census	nVidia Tegra X1 GPU nVidia Titan X GPU	(640x480)x128	19 237	747.1 9319.2	74.71 37.28
Vancea et al. [63]	**local block matching**	**Xilinx Virtex 4 FPGA**	**(640x512)x128**	**82**	**3419.7**	**923**

of algorithms, but they also entail different customizations within the same class. The differences are related to algorithmic issues concerning the cost function, the cost aggregation, the refinement techniques, the disparity resolution, the number of optimization paths or the number of stripes being processed in parallel. Apart from these algorithmic details, we propose optimization strategies for pipelining, capable to reduce hardware costs without affecting the clock rate at runtime (Fig. 2). In our exemplification, the optimizations are applied for a local block matching algorithm.

According to TABLE II it can be noticed a strong preference towards implementation on GPU and FPGA platforms, the latter being capable of real-time performance with suitable power efficiency for integration into mobile and embedded devices. There are several low-level aspects that must be considered when designing with FPGAs: area utilization measured by allocated FPGA logic, maintenance of critical path length to allow high running frequency, balance between memory usage and bandwidth. While concentrating on parallelization and pipelining at higher level most solutions do not offer low level details for the architecture they propose. Low level optimization requires strong knowledge of the underlying hardware support and fine details for efficient synthesizing, planning, routing and resource mapping.

IV. Conclusions

We have analyzed the most common models and strategies used for image rectification and image matching. The advantages and disadvantages have been presented for each identified class. A thorough study over the last two decades demonstrate intensive preoccupation for real-time performance and low power consumption solutions. We conclude that the most appreciated platforms for optimization and hardware acceleration are represented by GPU and FPGA, with categorical advantage for FPGA, when the power consumption is an important factor.

The analysis includes two pipeline architectures we proposed for image rectification and for image matching in FPGA. The image rectification model adopted is lossless and the implementation is capable to perform accurate image rectification in real-time. The access to the images stored in memory is reduced by incorporating an original caching technique. The optimization techniques we proposed for image matching decreased the hardware costs. The experiments demonstrate the efficiency of the proposed solution in matters of hardware utilization. Our architecture performs rectification and matching in real-time and registers one of the top scores in computational performance and energy efficiency.

Acknowledgment

This work has been supported by a grant of Romanian Ministry of Research and Innovation, CNCS - UEFISCDI, project number PN-III-P4-ID-PCE-2016-0727, within PNCDI III.

References

[1] S. Nedevschi, R. Dănescu, T. Mariţa, F. Oniga, C. Pocol, S. Bota, C. Vancea, “A Sensor for Urban Driving Assistance Systems Based on Dense Stereovision”, Stereo Vision, Ch. 14, pp. 235-259, InTech, November 2008.

[2] Available, 2018: http://www.xilinx.com/applications/automotive.html

[3] Available, 2018: https://www.nvidia.com/en-us/self-driving-cars

[4] Available, 2018: https://www.intel.com/content/www/us/en/automotive/products/programmable/overview.html

[5] C. Vancea, S. Nedevschi, “Analysis of different image rectification approaches for binocular stereovision systems”, Proceedings of IEEE 2nd International Conference on Intelligent Computer Communication and Processing, vol. 1, pp. 135-142, September 2006.

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

[7] Z. Zhang, “A flexible new technique for camera calibration”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, pp. 1330-1334, November 2000.

[8] D. Scharstein, R. Szeliski, “A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms”, International Journal of Computer Vision, vol. 47, no. 1, pp. 7-42, April 2002.

[9] O. Veksler, “Stereo correspondence by dynamic programming on a tree”, Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, pp. 384-390, June 2005.

[10] J. Kim, V. Kolmogorov, R. Zabih, “Visual correspondence using energy minimization and mutual information”, Proceedings of 9th IEEE International Conference on Computer Vision, pp. 1033-1040, October 2003.

[11] M. Bleyer, M. Gelautz, “A layered stereo matching algorithm using image segmentation and global visibility constraints”, ISPRS Journal of Photogrammetry and Remote Sensing, vol. 59, no. 3, pp. 128-150, May 2005.

[12] G. Li, S. Zucker, “Stereo for slanted surfaces: First order disparities and normal consistency”, Proceedings of 5th International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition, vol. 3757, pp. 617-632, November 2005.

[13] M. Bleyer, C. Rhemann, C. Rother, “PatchMatch Stereo-Stereo Matching with Slanted Support Windows”, Proceedings of the British Machine Vision Conference, vol. 11, pp. 1-11, September 2011.

[14] S. Nedevschi, R. Dănescu, D. Frenţiu, T. Mariţa, F. Oniga, C. Pocol, R. Schmidt, T. Graf, “High Accuracy Stereo Vision System for Far Distance Obstacle Detection”, Proceedings of 2004 IEEE Intelligent Vehicles Symposium, pp. 292-297, June 2004.

[15] M. P. Mureşan, S. Nedevschi, R. Dănescu, “Patch warping and local constraints for improved block matching stereo correspondence”, Proceedings of IEEE 12th International Conference on Intelligent Communication and Processing (ICCP), pp. 321-327, September 2016.

[16] M. P. Mureşan, M. Negru, S. Nedevschi, “Improving local stereo algorithms using binary shifted windows, fusion and smoothness constraint”, Proceedings of IEEE 11th International Conference on Intelligent Computer Communication and Processing (ICCP), pp. 179-185, September 2015.

[17] N. Einecke, J. Eggert, “A multi-block-matching approach for stereo”, Proceedings of IEEE Intelligent Vehicles Symposium, pp. 585-592, June 2015.

[18] C. Pantilie, S. Nedevschi, “SORT-SGM: Subpixel optimized real-time semiglobal matching for intelligent vehicles”, IEEE Transactions on Vehicular Technology, vol. 61, no. 3, pp. 1032-1042, March 2012.

[19] I. Haller, C. Pantilie, F. Oniga, S. Nedevschi, “Real-time semi-global dense stereo solution with improved sub-pixel accuracy”, Proceedings of 2010 IEEE Intelligent Vehicles Symposium (IV), pp. 369-376, June 2010.

[20] H. Hirschmüller, “Stereo processing by semiglobal matching and mutual information”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 2, pp. 328-341, February 2008.

[21] H. Hirschmüller, “Accurate and efficient stereo processing by semi-global matching and mutual information”, Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, pp. 807-814, June 2005.

[22] S. Hermann, R. Klette, E. Destefanis, “Inclusion of a second-order prior into semi-global matching”, Advances in Image and Video Technology, pp. 633-644, January 2009.

[23] R. Spangenberg, T. Langner, R. Rojas, “Weighted semi-global matching and center-symmetric census transform for robust driver assistance”, Proceeding of 15th International Conference on Computer Analysis of Images and Patterns, pp. 34-41, August 2013.

[24] J. Zbontar, Y. LeCun, “Computing the stereo matching cost with a convolutional neural network”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1592-1599, June 2015.

[25] J. Zbontar, Y. LeCun, “Stereo matching by training a convolutional neural network to compare image patches”, Journal of Machine Learning Research, vol. 17, no. 1, pp: 2287-2318, January 2016.

[26] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, A. Bry, “End-to-End Learning of Geometry and Context for Deep Stereo Regression”, Proceedings of IEEE International Conference on Computer Vision (ICCV), pp. 66-75, October 2017.

[27] C. Banz, S. Hesselbarth, H. Flatt, H. Blume, P. Pirsch, “Real-time stereo vision system using semi-global matching disparity estimation: Architecture and FPGA-implementation”, Proceedings of 2010 International Conference on Embedded Computer Systems (SAMOS) , pp. 93-101, July 2010.

[28] C. Banz, H. Blume, P. Pirsch, “Real-time semi-global matching disparity estimation on the GPU”, Proceedings of 2011 IEEE International Conference on Computer Vision (ICCV) Workshops, pp. 514-521, November 2011.

[29] C. Pantilie, I. Haller, M. Drulea, S. Nedevschi, “Real-time image rectification and stereo reconstruction system on the GPU”, Proceedings of 10th International Symposium on Parallel and Distributed Computing (ISPDC), pp. 79-85, December 2011.

[30] M. Pohl, M. Schaeferling, G. Kiefer, P. Petrow, E. Woitzel, F. Papenfuß, “An efficient and scalable architecture for real-time distortion removal and rectification of live camera images”, Proceedings of 2012 International Conference on Reconfigurable Computing and FPGAs (ReConFig), pp. 1-7, December 2012.

[31] P. Zicari, “Efficient and high performance FPGA-based rectification architecture for stereo vision”, Microprocessors and Microsystems, vol. 37, no. 8, pp. 1144-1154, November 2013.

[32] P. P. Shete, D. M. Sarode, S. K. Bose, “A real-time stereo rectification of high definition image stream using GPU”, Proceedings of 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI) , pp. 158-162, September 2014.

[33] A. Akin, I. Baz, L. M. Gaemperle, A. Schmid, Y. Leblebici, “Compressed look-up-table based real-time rectification hardware”, Proceeding of IFIP/IEEE 21st International Conference on Very Large Scale Integration (VLSI-SoC), pp. 272-277, October 2013.

[34] A. Akin, L. M. Gaemperle, H. Najibi, A. Schmid, Y. Leblebici, “Enhanced Compressed Look-up-Table Based Real-Time Rectification Hardware”, In IFIP Advances in Information and Communication Technology, vol. 461 (VLSI-SoC: At the Crossroads of Emerging Trends), pp. 227-248, Springer Cham, November 2015.

[35] S. N. Hung, J. Lee, B.J. You, “Real-time stereo rectification using compressed look-up table with variable breakpoint indexing”, Proceedings of IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society, pp. 4814-4819, October 2016.

[36] D. Han , J. Choi, S. J. Yoo, S. W. Baik, H. C. Shin, “The Design of HD Image Rectification Architecture Using Floating Point IP”, Proceeding of 2013 International Conference on Future Software Engineering and Multimedia Engineering (ICFM), IERI Procedia, vol. 6, pp. 39-44, 2014.

[37] J. H. Kim, J. G. Kim, J. K. Oh, S. M. Kang, J. D. Cho, “Efficient Hardware Implementation of Real-time Rectification using Adaptively Compressed LUT”, Journal of Semiconductor Technology and Science, vol. 16, no. 1, pp. 44-57, February 2016.

[38] D. Huffman, “A method for the construction of minimum-redundancy codes”, Proceedings of the IRE 40, no. 9, pp. 1098-1101, September 1952.

[39] P. Di Febbo, S. Mattoccia, C. Dal Mutto, “Real-time image distortion correction: Analysis and evaluation of FPGA-compatible algorithms”, Proceedings of 2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig), pp. 1-6, December 2016.

[40] C. Vancea, S. Nedevschi, “LUT-based image rectification module implemented in FPGA”, Proceedings of IEEE 3rd International Conference on Intelligent Computer Communication and Processing (ICCP), pp. 147-154, September 2007.

[41] T. Kanade, A. Yoshida, K. Oda, H. Kano, M. Tanaka, “A stereo machine for video-rate dense depth mapping and its new applications”, Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 196-202, June 1996.

[42] J. Woodfill, B. Von Herzen, “Real-time stereo vision on the PARTS reconfigurable computer”, Proceedings of the 5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, pp. 201-210, April 1997.

[43] K. Konolige, “Small vision systems: Hardware and implementation”, In Robotics research, pp. 203-212, Springer London, 1998.

[44] J. I. Woodfill, G. Gordon, R. Buck, “Tyzx DeepSea high speed stereo vision system”, Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshop (CVPRW), pp. 41-45, June 2004.

[45] J. I. Woodfill, G. Gordon, D. Jurasek, T. Brown, R. Buck, “The Tyzx DeepSea G2 Vision System, A Taskable, Embedded Stereo Camera”, Proceedings of Conference on Computer Vision and Pattern Recognition Workshop (CVPRW), pp. 126-132, June 2006.

[46] Y. Jia, X. Zhang, M. Li, L. An, “A miniature stereo vision machine (MSVM-III) for dense disparity mapping”, Proceedings of the 17th International Conference on Pattern Recognition (ICPR), vol. 1, pp. 728-731, August 2004.

[47] D. Masrani, W. J. MacLean, “A real-time large disparity range stereo-system using FPGAs”, Proceedings of IEEE International Conference on Computer Vision Systems (ICVS), pp. 13-13, January 2006.

[48] D. Fleet, “Disparity from local weighted phase-correlation”, IEEE International Conference on Systems, Man, and Cybernetics, 1994. Humans, Information and Technology, vol. 1, pp. 48-54, October 1994.

[49] S. Sabihuddin, W. J. MacLean, “Maximum-likelihood stereo correspondence using field programmable gate arrays”, Proceedings of the 5th International Conference on Computer Vision Systems (ICVS), March 2007.

[50] I. Cox, S. Hingorani, S. Rao, B. Maggs, “A maximum likelihood stereo algorithm”, Computer Vision and Image Understanding, vol. 63, no. 3, pp. 542-567, May 1996.

[51] S. Sabihuddin, J. Islam, W. J. MacLean, “Dynamic programming approach to high frame-rate stereo correspondence: A pipelined architecture implemented on a field programmable gate array”, Proceedings of Canadian Conference on Electrical and Computer Engineering (CCECE) 2008, pp. 001461-001466, May 2008.

[52] I. Ernst, H. Hirschmüller, “Mutual information based semi-global stereo matching on the GPU”, Proceedings of the 4th International Symposium on Advances in Visual Computing, pp. 228-239, December 2008.

[53] I. Rosenberg, P. Davidson, C. Muller, J. Han, “Real-time stereo vision using semi-global matching on programmable graphics hardware”, Proceedings of 33rd International Conference and Exhibition on Computer Graphics and Interactive Techniques – ACM SIGGRAPH 2006 Sketches, Article no. 89, August 2006.

[54] S. Birchfield, C. Tomasi, “Depth discontinuities by pixel-to-pixel stereo”, International Journal of Computer Vision, vol. 35, no. 3, pp. 269-293, December 1999.

[55] J. Gibson, O. Marques, “Stereo depth with a unified architecture GPU”, Proceedings of Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1-6, June 2008.

[56] S. Gehrig, F. Eberli, T. Meyer, “A real-time low-power stereo vision engine using semi-global matching”, Proceedings of International Conference on Computer Vision Systems, pp. 134-143, October 2009.

[57] S. Gehrig, C. Rabe, “Real-time semi-global matching on the CPU”, Proceedings of 2010 Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 85-92, June 2010.

[58] I. Haller, S. Nedevschi, “GPU optimization of the SGM stereo algorithm”, Proceedings of International Conference on Intelligent Computer Communication and Processing, pp. 197-202, August 2010.

[59] M. Michael, J. Salmen, J. Stallkamp, M. Schlipsing, “Real-time stereo vision: Optimizing semi-global matching”, Proceedings of IEEE Intelligent Vehicles Symposium (IV), pp. 1197-1202, June 2013.

[60] M. Buder, “Dense realtime stereo matching using a memory efficient Semi-Global-Matching variant based on FPGAs”, Proceedings of Real-Time Hardware, May 2012.

[61] R. Spangenberg, T. Langner, S. Adfeldt, R. Rojas, ”Large scale Semi-Global Matching on the CPU”, Proceedings of 2014 IEEE Intelligent Vehicles Symposium, pp. 195-201, June 2014.

[62] D. H.-Juarez, A. Chacón, A. Espinosa, D. Vázquez, J. C. Moure, A. López, “Embedded real-time stereo estimation via Semi-Global Matching on the GPU”, Proceedings of International Conference on Computational Science (ICCS), Procedia Computer Science, vol. 80, pp. 143-153, June 2016.

[63] C.-C. Vancea, S. Nedevshi, “FPGA-based stereo vision hardware for generating dense disparity maps”, Proceedings of IEEE 12th International Conference on Intelligent Communication and Processing (ICCP), pp. 225-232, September 2016.

[64] C.-C. Vancea, V.-C. Miclea, S. Nedevschi, “Improving stereo reconstruction by sub-pixel correction using histogram matching”, Proceedings of 2016 IEEE Intelligent Vehicles Symposium (IV), pp. 335-341, June 2016.