Architectures for Stereo Vision

Christian Banz, Holger Blume, and Peter Pirsch

Abstract Stereo vision is an elementary problem for many computer vision tasks. It has been widely studied under the two aspects of increasing the quality of the results and accelerating the computational processes. This chapter provides theoretic background on stereo vision systems and discusses architectures and implementations for real-time applications. In particular, the computationally most intensive part, the stereo matching, is discussed on the example of one of the leading algorithms, the semi-global matching (SGM). For this algorithm two implementations are presented in detail on two of the most relevant platforms for real-time image processing today: Field Programmable Gate Arrays (FPGAs) and Graphics Processing Units (GPUs). Thus, the major differences in designing parallelization techniques for extremely different image processing platforms are being illustrated.

1 Introduction

The field of stereo vision is highly inspired by the capabilities of the human imaging system. It encompasses all aspects of computer vision processing data from stereo image pairs in one way or another. The goal is to estimate 3D information about the observed scene, which can be used for a number of applications such as e.g. distance measurement, 3D reconstruction, and arbitrary view interpolation. Crucial for stereo vision is the task of stereo matching which identifies the projection points of the same 3D real world point in both images of the stereo pair. The location

C. Banz (✉) • H. Blume • P. Pirsch
Institute of Microelectronic Systems, Leibniz University of Hannover Appelstr. 4, 30167 Hannover, Germany
e-mail: christian.banz@alumni.uni-hannover.de; blume@ims.uni-hannover.de; pirsch@ims.uni-hannover.de

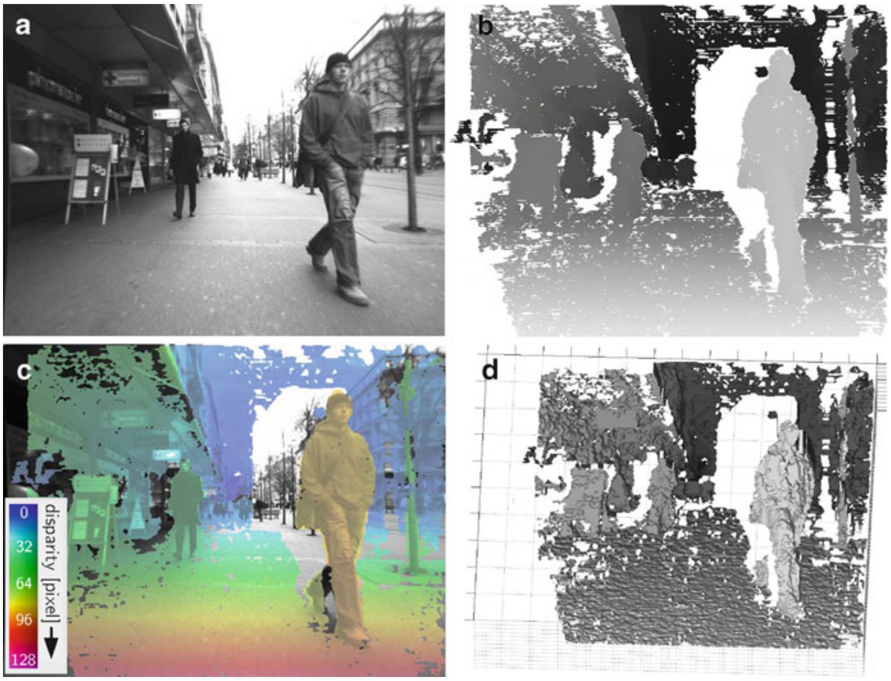

Fig. 1 Results for the stereo correspondence problem: (**a**) left rectified input image (raw input images taken from [20]), (**b**) disparity map after left/right check where *white* denotes disparities marked as invalid, (**c**) false color representation of the disparity map, (**d**) untextured 3D view generated from the disparity map of (**b**)

difference (*the disparity)* in conjunction with a known stereo camera calibration allows to infer the depth information. Figure 1 gives an example.

The importance of stereo matching has been underlined by Szeliski and Scharstein stating that it is "one of the most widely studied and fundamental problems of computer vision" [83]. Active research in this field has resulted in a wide range of disparity estimation algorithms using radically different approaches. Recently, a general taxonomy has been introduced [81] including a comprehensive survey, what resulted in the on-going online *Middlebury benchmark* [80]. Further surveys evaluated different algorithms and variations thereof [39, 86]. Major focus was the quality of the stereo matching in terms of accuracy, density of the disparity map, and robustness.

However, advances in robustness and accuracy were accompanied with significant increases in complexity and computational requirements making the use of specialized implementations for many of today's real-time applications an absolute necessity. Surveys on efficient implementations for selected types of algorithms have been conducted [29, 59, 66, 86] and many more specialized implementations and architectures for individual algorithms and applications have been proposed.

Considering all aspects (algorithmic performance, implementation performance, architectures) a huge design space is unfolded. For embedded systems the choice is invariably on low-power solutions, e.g. based on application specific architectures implemented on FPGAs or ASICs. However, with the recent rise of GPUs for high performance computing, GPUs offer a cost-efficient alternative for stationary systems where power consumption is not an issue.

This chapter addresses high performance disparity estimation considering both, algorithmic and implementation performance. The chapter is structured into an algorithmic and an architectural section; these being Sects. 2 and 3. An introduction to the fundamental principles of the stereo image matching (*epipolar geometry*) and a minimal practical stereo vision system is given in Sect. 2.1. The algorithmic and architecture sections both give a comprehensive overview of recent works. It is followed by a detailed discussion of the semi-global matching algorithm (SGM) [37] (Sect. 2.3) and two exemplary implementations on FPGA (Sect. 3.7) and GPU (Sect. 3.6), respectively.

2 Algorithms

A minimal system for disparity estimation from a real camera setup consists of two processing steps: The first step is camera lens undistortion and *rectification* of non-ideal stereo camera setup (Sect. 2.1) while the second step is the actual stereo matching (Sect. 2.2). All other image preprocessing steps (e.g. noise reduction, equalization) and disparity map post-processing steps (e.g. whole filling, interpolation of pixel with missing stereo information) are optional.

2.1 Epipolar Geometry and Rectification

The objective is to find corresponding pixels in the two images of a stereo camera setup. Due to the underlying epipolar geometry [34, 83] of a stereo camera setup, the search space for corresponding pixels is one-dimensional. As shown in Fig. 2a, for a given pixel in the base image all potential correspondences project onto the *epipolar line* (e_{bm}) in the match image and vice versa. Strictly speaking the possible projections are bound by the *epipole* and the viewing rays for a real-world point at infinity.

For efficient correspondence search implementations a preprocessing step, the *rectification*, is employed. Both images are warped such that epipolar lines in both images are parallel to the scanlines and are row-aligned, i.e. corresponding pixels are in the same horizontal line [34, 103]. Thus, efficient memory access patterns and parallelism over independent scanlines can be obtained. After rectification, the focal axis are parallel to each other and perpendicular to the line joining the two camera centers (*baseline*) and the disparity for points at infinity is 0.

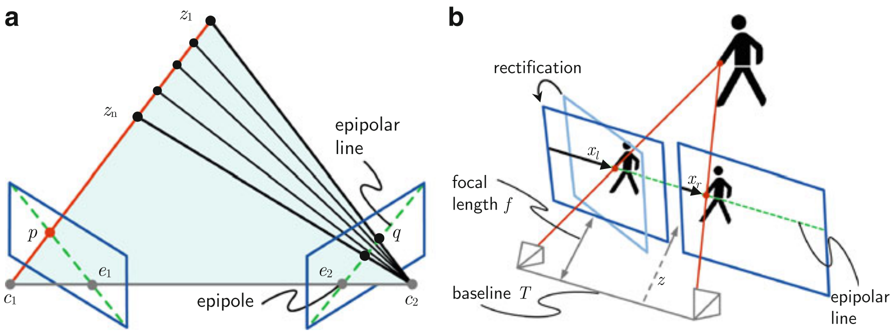

Fig. 2 Epipolar geometry: in (**a**) an unrectified setup and in (**b**) a rectified setup is shown. The rectification process in (**b**) to achieve row-aligned search space is illustrated only for the left projection plane

The rectified stereo setup is shown in Fig. 2b and the disparity is purely horizontal offset $d = (x_l - x_r)$ [pixel]. With the rectified focal length f [pixel], the baseline T [m] of the camera pair, the distance z [m] between the baseline and the 3D point can be calculated as

$$z = \frac{fT}{(x_l - x_r)} = \frac{fT}{d}. \tag{1}$$

This is also referred to as *standard rectified geometry* [83]. Thus, extracting depth information from a stereo camera setup becomes estimating the *disparity map* $d(x,y)$.

In addition to a non-ideal camera setup, stereo vision systems have to handle camera-inflicted image distortions, of which the most common are radial lens distortion, sensor tilting and offset from focal axis [11]. These must be compensated *before* rectification. However, when applying undistortion and rectification to a sequence of input images both steps can be combined. Reverse mapping assigns every pixel in the undistorted and rectified image a sub-pixel accurate origin in the input image. The rectified pixels are obtained using any desired pixel interpolation method. The bilinear interpolation for example, exhibits a reasonable trade-off between image quality and hardware implementation costs. Alternative interpolation methods are spline interpolation, which has higher silicon area requirements, and nearest-neighbor, which does not provide the required resolution for disparity estimation. Intermediate results from the processing steps are shown in Fig. 3.

The displacement vectors for undistortion and rectification are calculated using the intrinsic and extrinsic matrices, the tangential and the radial distortion parameters. These can be obtained by separate camera calibration steps (e.g. [104]) using a calibration pattern, such as a chessboard pattern employed in OpenCV [11]. Alternatively, or additionally, camera self-calibration from scene structure can be employed for particular camera parameters. For latter use in e.g. cars, camera self-calibration or at least updating of the intrinsic parameters from scene structure is mandatory.

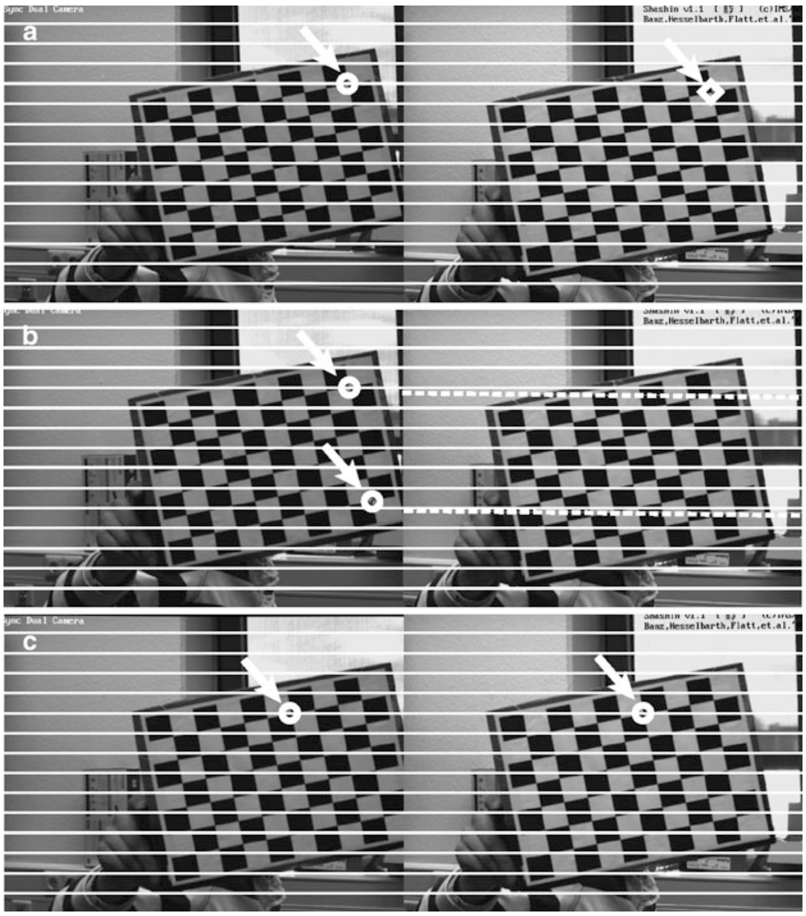

Fig. 3 Image results after undistortion and rectification: (**a**) input images showing that correspondences are not aligned (*circle* and *square*). (**b**) Undistorted images showing the epipolar lines (*dashed lines*) for two exemplary points (*circles*). Here, the effect is minor but the epipolar lines are clearly not aligned to the scanlines (i.e. horizontal pixel rows, *white*). (**c**) Final, undistorted, rectified images with row aligned epipolar lines

2.2 Stereo Correspondence

The origins of stereo correspondence were *sparse*, feature-based methods processing only a set of potentially highly discriminative image points. Today, most algorithms are *dense* methods, trying to infer a complete disparity map even for texture-less regions. Dense methods are typically classified into *local* and *global* approaches. However, for both classes of dense methods a common taxonomy and categorization has been introduced in [81]. Generally, a disparity estimation algorithm consists of the four processing steps:

1. Matching cost computation
2. Cost (support) aggregation
3. Disparity computation and optimization
4. Disparity refinement (or post processing)

It is of crucial importance to distinguish between the matching costs, which is the initial similarity measure between two pixels in the base and match image (or left and right image, respectively), and the aggregation method that uses these costs. The results of the matching cost computation are stored in the *disparity space image* $C(x,y,d)$. Cost aggregation of local (or area based) methods is performed on the information based in a local aggregation region (*support region*) from the matching costs $C(x,y,d)$. Global methods on the other hand perform one or more optimization steps on the matching costs often enforcing some kind of smoothness criterion. Depending on the algorithm, steps have varying importance and some might even be omitted. An example will be given in Sect. 2.3.

For matching cost computation a number of different window-based similarity measures can be employed. With rectified input images similarity of potentially corresponding pixels must be computed at location $\mathbf{p} = [x,y]^T$ in the left image and $\mathbf{q} = [x-d,y]^T$ in the right image. Initially often used and inspired by other areas of video processing are sum of squared intensity differences (SSD), sum of absolute differences (SAD), normalized cross correlation (NCC) and their respective zero mean variations. More recently, measures specifically for stereo matching have been proposed. For example, rank and census transform [99] are non-parametric transforms, and are thus robust to a certain amount of intensity differences. A vast number of other measures based on gradients, phase correlations, ordinal measures and dense feature descriptors exist. Entropy based measures (e.g. mutual information [37, 51]) have also been proposed. For those measures that compare absolute difference values, the approach of Birchfeld and Tomasi (BT) [9] can be used to include sampling insensitivity. For a more complete list of similarity measures refer to [83]. Detailed studies on the performance of the similarity measures in conjunction with different aggregation methods have been conducted [39, 81, 90].

The emphasis of local methods is on the cost aggregation step (step 2). A recent, comprehensive comparison of aggregation methods can be found in [86] and of selected methods for GPU implementation in [29]. Support regions can be two- or three-dimensional windows from the disparity space with fixed or adaptive window sizes, shapes, anchor points or weights. Adaptation may e.g. be performed by a full search through multiple windows or from a number of cues, e.g. constant disparity constraints and color-based segmentation. A more complete list may also be found in [83]. After cost aggregation, follows the disparity computation (step 3) of which the most basic form is selecting the disparity with minimal aggregated cost value for each pixel. Local methods are often well suited for hardware implementation due to the implicit parallelism and local data dependencies.

With global methods, the cost aggregation (step 2) is often omitted because the global smoothness constraints, which are enforced by the optimization process during the disparity computation (step 3), perform similar functions [83]. Global methods are often formulated within an energy-minimization framework:

$$E(D) = E_d(D) + \lambda E_s(D). \tag{2}$$

The objective is to find a solution d that minimizes the total energy E for a disparity map D, where E_d is the data term representing how well the solution fits to the input image and E_s represents the smoothness constraints made by the algorithm. These *regularization* or variational formulations are also employed in many others areas of image processing. In stereo processing it is important to formulate $E_s(d)$ to allow for *discontinuity preservation* in the disparity map. Algorithms to find the solution to (2) include belief propagation, graph cuts, and total variation among others [83]. Unfortunately, the problem is NP-hard for many discontinuity preserving if E_s is formulated two-dimensionally [10]. Reducing E_s to one-dimension along the scanlines, allows for independent, parallel *scanline optimization* but suffers from streaking (inconsistency between scanlines). Other global methods are based on *dynamic programming*, which performs global optimization for independent scanlines. Dynamic programming also suffers from streaking, but several works have addressed this problem (e.g. [78]).

For each approach several algorithms have been proposed and minute details influence the performance. As mentioned in Sect. 1, comparative studies have been performed (e.g. [29, 39, 81]) and a widely used benchmark exists [80]. For a stereo vision system with high performance in terms of robustness, accuracy and processing speed, several aspects have to be weighted against each other. While some local methods are more efficiently implementable, they can be challenged by areas with low or repetitive textures due to a high level of ambiguity [39]. Iterative, global minimization methods are often computationally intensive. However, Tombari et al. [86] express, that with sophisticated cost aggregation some local methods yield performance comparable with many global methods. The semi-global optimization strategy [37] is a solution resident in between by accumulating optimization results from multiple independent one-dimensional directions for each pixel. It produces very high quality results, even though not *the* best in the Middlebury benchmark [80]. Further, it is robust and it can be implemented efficiently for a global method. For robustness of the entire disparity estimation a suitable similarity measure must be chosen. Among the more robust measures in [38, 39] were census, rank, ordinal measures, and hierarchical mutual information.

Disparity refinement (step 4) often includes sub-pixel refinement, confidence or integrity checks, and interpolation measures. Since most stereo methods compute disparities at integer level, a *sub-pixel* refinement is necessary for many applications. An easy and computationally efficient way is to fit a curve through the discrete disparity space around the selected disparity. Interpolation functions are investigated in [32]. Several arising issues are discussed in [84]. An often computationally prohibitively expensive alternative is to start the computation with a disparity space already discretized to sub-pixel accuracy.

Foreground objects in the scene occlude different parts of the background when seen from the two camera perspectives. Consequently disparities cannot be computed for these occluded areas of the image due to missing stereo correspondences. This is visible in Fig. 1 by the halos around the foreground objects. It is often desirable to exclude these areas and areas with low confidence from the disparity map and optionally process them with sophisticated hole filling algorithms. Identification of these areas is performed with a *left/right check*, where the disparity maps for the left and right perspective are computed and only matching depth information from both perspectives to a 3D world point is allowed. With respect to the camera-to-camera projection in a rectified stereo pair the constraint for a valid disparity in the base image can be formulated as

$$D_{b,\text{check}}(x,y) = \begin{cases} D_b & \text{if} \quad \left|D_b(x,y) - D_m\left(e_{mb}\left(x, D_b(x,y)\right), y\right)\right| \leq \delta \\ \textit{invalid} & \text{otherwise} \end{cases} \tag{3}$$

with D_b and D_m are the disparity maps from the base and match perspective, respectively.

Further post-processing of the disparity map can be performed using basic median filtering to remove single outliers, peak removal and sophisticated whole filling algorithms, such as surface fitting. However, without a dense, highly accurate initial disparity map, post-processing will not provide reliable disparities.

2.3 Algorithm Example: Semi-global Matching

As a specific example disparity estimation based on the highly relevant and top-performing combination of rank transform [99] and semi-global matching algorithm (SGM) [37] will be used to illustrate the matter of the previous sections. Simultaneously, SGM will be used as a case study for implementations on FPGA and GPU.

The matching costs $C(x,y,d)$ (step 1) are calculated from the rank transform of the base and match image R_b and R_m with absolute difference comparison:

$$C(\mathbf{p},d) = \left|R_b(p_x,p_y) - R_m(p_x-d,p_y)\right|. \tag{4}$$

It is $\mathbf{p} = [p_x,p_y]^T$ the pixel location in the left image. The rank transform is defined as the number of pixels $\mathbf{p}'$ in a square $M \times M$ neighborhood $A(\mathbf{p})$ of the center pixel $\mathbf{p}$ with a luminous intensity I less than $I(\mathbf{p})$

$$R(\mathbf{p}) = \left\|\left\{\mathbf{p}' \in A(\mathbf{p}) \mid I(\mathbf{p}') < I(\mathbf{p})\right\}\right\|. \tag{5}$$

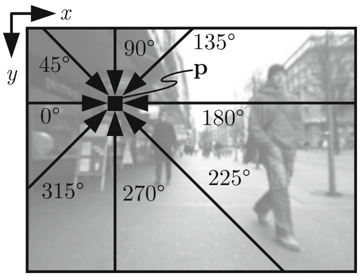

Fig. 4 The path cost aggregation is performed from eight cardinal directions to every pixel

These initial pixel-wise calculated matching costs (i.e. locally calculated) yield non-unique or wrong correspondences due to low texture and ambiguity. Therefore, semi-global matching introduces global consistency constraints by aggregating matching costs along several independent, one-dimensional *paths* from different cardinal directions as shown in Fig. 4. A path $\mathbf{r}$ is formulated recursively by the definition of the path costs $L_r(\mathbf{p},d)$.

$$\begin{aligned} L_r(\mathbf{p},d) = C(\mathbf{p},d) + \min\,[&L_r(\mathbf{p}-\mathbf{r},d)\;, \\ &L_r(\mathbf{p}-\mathbf{r},d-1)+P_1, \\ &L_r(\mathbf{p}-\mathbf{r},d+1)+P_1, \\ &\min_i L_r(\mathbf{p}-\mathbf{r},i)+P_2] - \\ &\min_l L_r(\mathbf{p}-\mathbf{r},l) \end{aligned} \tag{6}$$

The first term, $C(\mathbf{p},d)$, describes the initial matching costs. The second term adds the minimal path costs of the previous pixel $\mathbf{p}-\mathbf{r}$ including a penalty P_1 for disparity changes and P_2 for disparity discontinuities, respectively. Discrimination of small changes $|\Delta d| = 1$ pixel *(px)* and discontinuities $|\Delta d| > 1$ px allows for slanted and curved surfaces on the one hand and preserves disparity discontinuities on the other hand. The last term prevents constantly increasing path costs. For a detailed discussion refer to [37]. P_1 is an empirically determined constant. P_2 can also be an empirically determined constant or can be adapted to the image content. The selection of these penalty functions is investigated in [5] in detail.

Path costs are calculated from several cardinal directions to each pixel, as shown in Fig. 4 and are summed. The aggregated sum costs S are the sum of the path costs

$$S(\mathbf{p},d) = \sum_{\mathbf{r}} L_r(\mathbf{p},d). \tag{7}$$

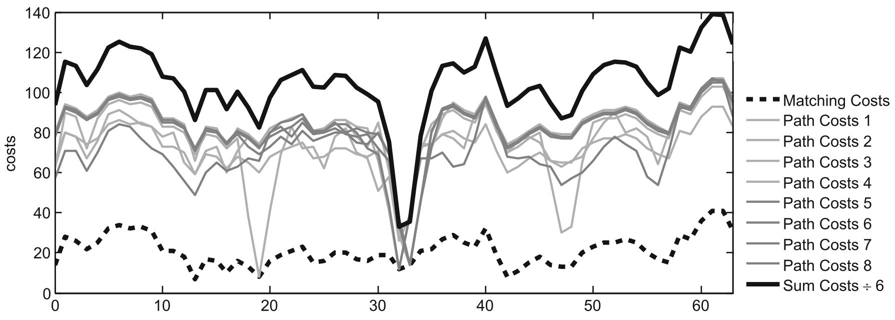

Fig. 5 Effect of path cost aggregation: matching costs, aggregated path costs, and sum costs (scaled by factor 1/6 for better presentation) for the pixel $\mathbf{p} = [183;278]$ of the Teddy test image [82] calculated with SGM

By (6) and (7) SGM aims to approximate the following global energy minimization problem:

$$E(D) = \underbrace{\sum_{\mathbf{p}} C(\mathbf{p}, d)}_{E_d(D)}$$

$$+ \underbrace{\sum_{\mathbf{p}} \left(\sum_{\mathbf{p}' \in A} P_1 \mathrm{T}\left[\left| D_{\mathrm{p}} - D_{\mathrm{p}'} \right| = 1 \right] + \sum_{\mathbf{p}' \in A} P_2 \mathrm{T}\left[\left| D_{\mathrm{p}} - D_{\mathrm{p}'} \right| > 1 \right] \right)}_{E_s(D)} \tag{8}$$

where E_s contains the 2D smoothness constraints on the disparity map. For a derivation of (8) see [37]. The resulting method of the approximation resembles a scanline optimization approach but with excellent regard to interscanline consistencies.

Final disparity selection (step 3) is performed by a *winner-take-all* (WTA) approach. The disparity map $D_b(p_x, p_y)$ from the perspective of the base camera is calculated by selecting the disparity with the minimal aggregated costs

$$\min_d S(p_x, p_y, d) \tag{9}$$

for each pixel. For calculating the disparity map from the perspective of the match camera $D_m(q_x, q_y)$, the minimal aggregated costs along the corresponding epipolar lines are selected:

$$\min_d S(q_x + d, q_y, d) \,. \tag{10}$$

Alternatively, SGM can be applied again, but with the other image as base image.

The effect of the path costs aggregation and the disparity selection is illustrated in Fig. 5. The initial matching costs $C(\mathbf{p}, d)$ (dashed line) exhibit a high level of ambiguity. Seven of the eight aggregated paths costs $L_r(\mathbf{p}, d)$ already show distinct

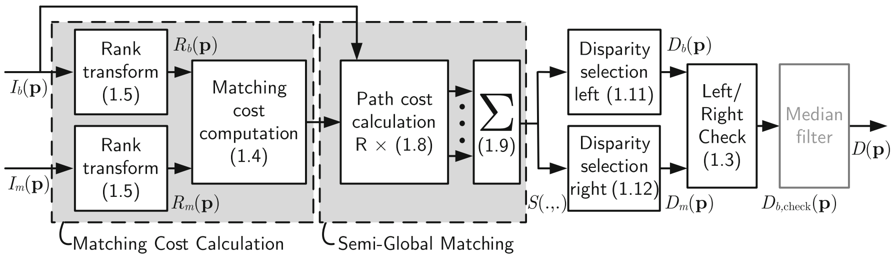

Fig. 6 Processing steps for disparity estimation using rank transform, semi-global matching, and optional median filter. Numbers in parenthesis refer to the respective equations and R denotes the number of paths

minima. The summed path costs $S(\mathbf{p},d)$ (thick black line) clearly identify the minimum at a disparity level of 32 resolving all ambiguities. However, the cost difference for the positions 32 and 33 is minimal indicating that the correct position is located a sub-pel precision.

Finally, left/right check according to (3) and post processing can be applied, e.g. a median filter in its most basic form. An overview of the processing steps is given in Fig. 6.

3 Architectures

The variety of architectures and implementations to compute the stereo correspondence easily rivals the variety of the underlying stereo matching algorithms. Today very efficient implementations for local and global stereo methods are available on FPGAs, ASICs, GPUs, and DSPs. For real-time image throughput, local methods have been and continue to be favored by many researches because of their efficient implementation possibilities. However, with advances in computational power, many global methods are also implementable in real-time.

Early work includes a complete stereo vision system from 1996 featuring rectification and stereo matching with an SSD variant on a custom hardware board consisting of off-the-shelf components and a DSP array [49]. At 30 fps 200×200 images with 5 bit depth resolution could be computed. Other noteworthy early implementations have been presented in 1993 using a DSP array [21] and in 1997 using a single DSP [52]. In [52] also an early overview of implementations is provided. An FPGA array was used in 1997 to implement a census transform stereo matching method [95]. All of these implementations directly compute the disparity information from the matching costs without cost aggregation. Early work includes a complete stereo vision system from 1996 featuring rectification and stereo matching with an SSD variant on a custom hardware board consisting of off-the-shelf components and an DSP array [49]. At 30 fps 200×200 images with

5 bit depth resolution could be computed. Other noteworthy early implementations have been presented in 1993 using a DSP array [21] and in 1997 using a single DSP [52]. In [52] also an early overview of implementations is provided. An FPGA array was used in 1997 to implement a census transform stereo matching method [95]. All of these implementations directly compute the disparity information from the matching costs without cost aggregation.

The following three subsections aim to give an overview of the conducted research on architectures and implementations for disparity estimation. Each subsection focuses on one specific hardware platform. Some key throughput values will be highlighted but without indication of algorithmic performance. A fair comparison must take into account architecture specific features, scalability, algorithmic performance under challenging imaging conditions (which are not present in the standard Middlebury data set), termination criterions on data dependent algorithms (e.g. belief propagation), and varying post processing steps. Thus, a comparison is an extremely complex task and beyond the scope of this chapter. The interested reader may consult the references themselves or one of the comparison studies.

3.1 GPU-Based Implementations

Commodity graphics processing hardware, nowadays superseded by *general purpose graphics processing unit*s (GPUs), have been used quite early to outsource, at first, part of the computation and then the entire stereo matching. For the calculation of disparity maps with an image size of 200×200 pixels and 50 disparity levels (abbreviated $200\times200\times50$), 106 ms were achieved using a variable window SSD method on a NVIDIA GeForce4 in 2003 [76]. Early implementations for scene reconstruction are [77] on a Nvidia GeForce4 and [100] on a ATI Radeon 9700Pro from 2002 and 2003, respectively.

For belief propagation (BP) an efficient technique has been proposed in [22] and implemented for CPUs. It has been extended with occlusion handling and adapted for GPU implementation [12]. The same technique is used in recent implementations [43, 96] reaching 2.75 s for a $640\times480\times33$ image on Nvidia GeForce GTX 280 and 93.98 s on an Intel Core 2 Duo (2.13 GHz). A fast converging hierarchical belief propagation is proposed and implemented in [97] reaching 16 fps for $320\times240\times16$ images. New message passing schemes for BP have been applied in [55] for a GPU and VLSI implementation.

A dynamic programming solution with extensive use of MMX instructions on the CPU using color based cost aggregation has been presented in [23]. In 2005 the dynamic programming optimization step was still slower on the GPU than on the CPU [30] due to limitations of general purpose computation capabilities of the GPU (e.g. branching). Consequently, mixed CPU/GPU implementations performing cost aggregation on the GPU and dynamic programming optimization on the CPU have been presented [30, 56]. Scanline optimization in [101] also shows mixed performance results when comparing GPU versus CPU implementations. Recently,

[48] presented a multi-resolution symmetric dynamic programming variant on a GTX 295 reaching 14 fps for 2048×2048×256 images. A total variation algorithm with GPU implementation has been presented requiring between 15 and 60 s per image [73].

Variants of local methods examining the different techniques of adaptive weights or adaptive support regions have received much attention. Recent local approaches are census based with basic box filter cost aggregation [92] and a local truncated laplacian kernel approximation with adaptive cost aggregation [44]. Locally adaptive support regions have been used and speeded up with bitwise voting in [50]. Further work on local variants with adaptive cost aggregation methods includes [45, 63] and [40]. Instead of adaptive support regions on the input images [61] use edge-preserving filtering on the matching costs. A comparison of six local methods in terms of algorithmic and computational performance on GPUs has been conducted [29]. A plane sweep algorithm with local depth connectivity in order to retain depth discontinuities has been examined in [16].

For SGM various implementations have been presented on a GeForce 8800 Ultra [19] (0.0057 fps at 640×480×128), a Quadro FX5600 [27], a GTX 280 without [31] and with increased depth accuracy [67], and on a Tesla C2050 [4], which is the highest performing implementation with 63 fps for 640×480×128 images. This allows a very interesting retrospective on the evolution of GPUs. Especially some of the new features of Nvidia's compute capability 2.0 graphics cards allow radically different parallelization schemes, which was exploited in [4]. We will have a detailed look at this implementation in Sect. 3.6. Furthermore, a combination of adaptive support regions with a reduced version of SGM is proposed in [62] reaching 10 fps for 450×375×64 images.

3.2 *Dedicated Architectures (FPGA and VLSI)*

For dedicated architectures targeting FPGAs or ASICs, local methods are often favored because of potentially very small designs. This goes as far as to omit the cost aggregation altogether despite the drawbacks in accuracy and robustness. Nevertheless, new cost aggregation concepts have also been investigated and incorporated in hardware. In the following implementations without cost aggregation are indicated with "w/o CA".

Some examples of early architectures using SAD based matching w/o CA are [2, 54, 64]. An SAD based stereo vision system with three cameras has been presented in [98]. Depending on the emphasis of the referenced work, the results vary in throughput and resolution up to 640×480×64 and 31 fps. The so-called Tyzx ASIC for color-image census-based stereo-matching (w/o CA) achieves 200 fps for 512×480 images and 52 disparity levels [93]. It forms the basis of an extended stereo vision system in [94].

Also for recent implementations local methods with and without cost aggregation are still popular. This includes [46] where a census transform (w/o CA) is employed as basis of an entire stereo vision system on an FPGA. Another complete system based on SAD (w/o CA) is presented in [91]. Census with aggregation cues from the original and gradient images is investigated in [1]. Color SAD with a fuzzy logic disparity selection has been proposed and implemented on an FPGA [26]. Methods and architectures using adaptive support weights have been proposed in [14] employing a census variant and in [89] employing an absolute differences variant. In order to reduce the amount of data to be processed [88] works on sobel filtered images, which goes into the direction of sparse matching.

In [60], the architecture of [17], which is based on a local, phase-based method, is extended to large disparity ranges without significant additional hardware cost by adapting an offset of the smaller disparity search window across multiple frames. After large disparity changes, a latency of several frames occurs before correct disparity information can be regained. A bio-inspired method based on gabor filters is introduced in [18].

Among the implementations of dynamic programming approaches a trellis-based implementation, using a single interline consistency constraint has been investigated [68]. A dynamic programming approach based on a maximum-likelihood method is implemented in [78] achieving 64 fps at 640×480 px with 128 disparity levels. And a symmetric dynamic programming variant, similar to the GPU implementation of [48], has been implemented on an FPGA [65].

An FPGA architecture for memory efficient belief propagation for stereo matching has been proposed in [71]. New concepts and architectures for the message passing in BP are proposed [87].

For semi-global matching two architectures have been proposed. The implementation of [25] utilizes a SGM variant with depth adaptive sub-sampling. It achieves 27 fps at 320×200 px and 64 px disparity range. A parameterizable parallelization scheme for SGM and a corresponding FPGA architecture have been proposed in [7, 8]. It achieves, depending on the degree of parallelism, up to 176 fps for VGA images with 128 disparity levels and 4 SGM paths. This architecture will be studied in more detail in Sect. 3.7.

3.3 Other Architectures

The use of programmable architectures besides GPUs has also been investigated in some depth. Mühlmann et al. [66] investigated memory layout schemes for the disparity space and implementations schemes including MMX optimizations for SAD-based matching without cost aggregation (w/o CA).

A number of publications specifically target programmable embedded solutions: An SSD with multiple window selection has been implemented on the ClearSpeed CSX700 architecture (250 MHz, 9 W) which provides massively parallel SIMD in multiple parallel processing elements [41]. The same algorithm has been implemented [79] on the Tilera TILEPro64, which is a MIMD architecture with 64 integer processing cores organized in a two dimensional mesh network running at MHz.

A SAD w/o CA is also investigated on the Tilera TilePro 64 and on many-core CPUs [75]. SAD (w/o CA) for a VLIW processor (Texas Instruments TMS320C6414T, 1.0 GHz) has been shown in [13].

Application specific processors (ASIP) have been investigated in two cases: For semi-global matching an instruction set extension for the Tensilica LX2 DSP template has been proposed [6] reporting 20 fps for 640×480×64 images with reduced number of paths when run at 373 MHz, which is possible with the targeted TSMC 90 nm process. Similarly for SGM, architecture optimizations for a VLIW processor template, the MOAI, have been investigated in [69] reaching 30 fps when running at 400 MHz.

Apart from the original CPU implementation of SGM running at 1.3 s for 450× 375×64 images [36], a variant with depth adaptive sub-sampling has been proposed running at 14 fps for 320×160 images [24].

The cell broadband engine has been utilized for belief propagation and dynamic programming, both taking few seconds to process an image pair [58]. An SAD (w/o CA) implementation on the cell achieves 30 fps for VGA images with 48 disparity levels.

3.4 Comparison Studies

In addition to the algorithmic studies mentioned earlier, studies also taking into account the computational performance have been conducted. An evaluation of cost aggregation for local methods with focus on algorithmic performance and run-time on CPU can be found in [86]. Selected algorithms (various SAD variants, belief propagation, and dynamic programing) have been compared on a CPU in [59]. An evaluation of local algorithms on the GPU has been conducted in [29, 57].

An implementation of belief propagation on GPU and for VLSI has been compared in [55]. And symmetric dynamic programing on GPU and FPGA has been compared in [47]. Comparison of a census based approach (w/o CA) on a DSP (TI C6416), a GPU (GeForce 9800 GT), and a CPU (Intel Core2Quad) has been conducted in [42]. And in [75] SAD (w/o CA) has been studied on a GPU, two multi-core CPUs and the MIMD Tilea architecture. Further, in many of the references in the previous sections the GPU or FPGA implementation is compared to a regular CPU implementation. However, these are too numerous to list them here.

3.5 Current Trends

When targeting real-world applications, an everlasting question is to improve algorithmic performance while reducing computational requirements. This has already been addressed in many of the above references. A recent research direction is to integrate the computation of various information retrieval image processing tasks (e.g. disparity estimation with optical flow). In [28] an algorithm for joint computation of disparity estimation and optical flow is proposed and implemented on the GPU. A holistic architecture for phase based disparity estimation, optical flow, and more is presented [85] and implemented on an FPGA. An holistic architecture for disparity estimation and motion estimation based on SAD is presented in [102].

3.6 Implementation Example: Semi-global Matching on the GPU

An example implementation of the semi-global matching algorithm for GPUs will be given based on the works in [4]. Since GPUs are becoming more and more common, an introduction of the architecture and the terminology will be skipped. Please refer to the Nvidia manuals and [35] for a detailed background on GPU architecture or directly to [4] for a short sketch. The evaluation platform in the following is a Nvidia Tesla C2050 with compute capability 2.0 providing 3 GB DDRRAM global memory with a maximum theoretical bandwidth of 144 GB/s.

3.6.1 Parallelization Principles

Banz et al. [4] formulate the following performance limiting factors for a kernel:

- **Effective memory bandwidth usage** for the payload data which is e.g. reduced by nonaligned, overhead-producing memory access
- **Instruction throughput** defined as the number of instructions performing arithmetics for the core computation and other non-ancillary instructions per unit of time
- **Latency of the memory interface** occurring e.g. when accessing scattered memory locations even if aligned and coalesced, warp-wise access is performed
- **Latency of the arithmetic pipeline** of the ALUs inside the GPU cores if arithmetic instructions depend on each other and can only be executed with the result from the previous instruction

Accordingly, kernels can be memory bound, compute bound or latency bound. Kernels that are not limited by any of the three bounds are ill-adapted for GPU implementation and can be classified as bound by their parallelization scheme.

An efficient parallelization scheme guarantees inherently *aligned* and *coalesced* data access schemes without instruction overhead. Coalesced memory access is the simultaneous memory access to consecutive memory locations of all threads of a warp. It further includes a combination of parallel and sequential processing with independent arithmetic computation steps. An inner (sequential) loop in the otherwise parallel threads working on a set of data that is kept in shared memory or

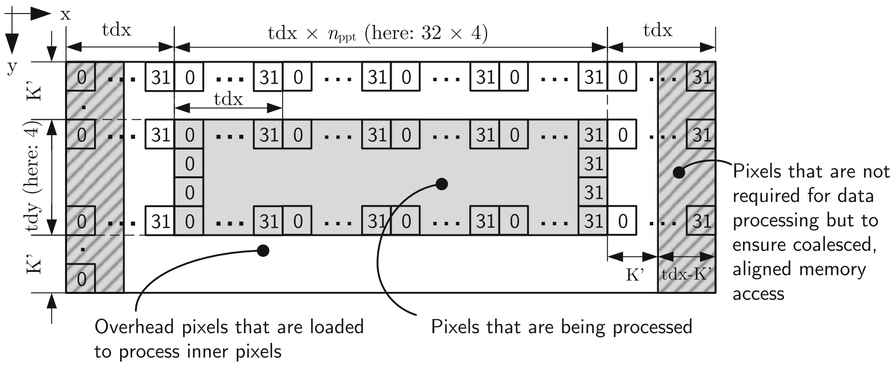
x
y
tdx
tdx × n_{ppt} (here: 32 × 4)
tdx
K'
tdy (here: 4)
K'
tdx-K'
Pixels that are not required for data processing but to ensure coalesced, aligned memory access
Overhead pixels that are loaded to process inner pixels
Pixels that are being processed

Fig. 7 Data fetching and accessing scheme for the 2D filter kernels processing $tdx \cdot n_{\mathrm{ppt}} \times tdy$ kernel windows with a radius of K' where n_{ppt} is the number of pixels processed per thread and the launch configuration, which determines how threads are grouped and executed on the streaming multiprocessors, is $tdx \times tdy$. *Each square* represents a pixel and the number inside is the x-dimension thread ID which fetches the pixel from global memory

register facilitates data reuse, increases the instruction ratio, and keeps the pipeline filled. Further, coherent access schemes are ensured for the memory interface if results are written out with each loop iteration. Apart from an inner loop, executing several warps per streaming multiprocessor increases pipeline utilization.

3.6.2 Rank Transform and Median Filter Kernel

The rank transform and median filter are both non-linear, non-separable 2D image transforms. To generate the result of one output pixel, the data of a local $N \times N$-neighborhood from the input image is required.

The kernel for rank transform and median filter are based on the same principle which is based on the implementation of a separable convolution in [74]. It prefetches data of a two-dimensional spatial locality from global memory into shared memory. Thus, data reuse is maximized because all filter kernels that fully reside in this spatial locality can be processed by a block of threads without additional global memory access. An aligned group of pixels is processed by a two-dimensional block of threads first loading the neighboring center pixels of all kernels. Left and right pixels outside the center area are always loaded with the warp width. Even though this causes minimal data to be loaded which is not used by the current block, it ensures inherent coalesced memory access without instruction overhead or warp divergence. An inner loop allows to process several pixels per thread (n_{ppt}) with a stride of the warp width. Adjusting n_{ppt} and the *launch configuration*, i.e. the number of threads per block in x-dimension (*tdx*) and y-dimension (*tdy*), allows to navigate between the different optimization principles. Figure 7 shows the data layout and thread access scheme.

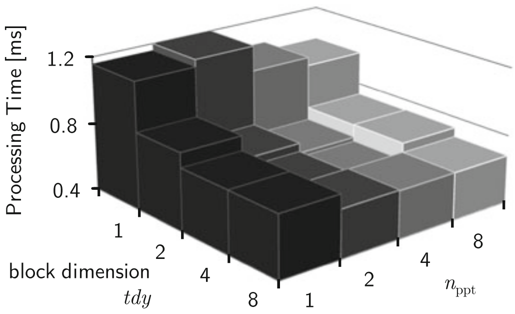

Fig. 8 Performance of the 3×3 median filter: on 1280×960 images as the parallelization configuration changes. Block width is fixed to $tdx = 32$. Best performance is achieved with $tdx \times tdy = 32 \times 4$ and $n_{\mathrm{ppt}} = 4$

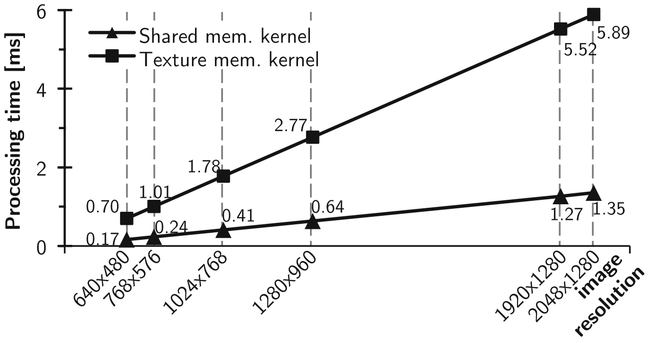

Fig. 9 Performance of the 3×3 median filter: comparison of the texture memory kernel and the proposed shared memory kernel on a Tesla C2050 GPU for the best-performing parallelization configuration

The median filter is always compute bound and performs best with $tdx \times tdy = 32 \times 4$ threads and $n_{\mathrm{ppt}} = 4$. The results of the parameter study for $tdx = 32$ are shown in Fig. 8. Configurations with $n_{\mathrm{ppt}} = 8$ perform slightly worse although redundant memory access is further reduced because of inefficient pipeline utilization. Processing times for a 3×3 median filter (i.e. kernel radius $K' = 1$) are given in Fig. 9 resulting in 0.64 ms for the new shared memory based kernel. For a texture-memory based kernel, which is the most often suggested way of implementing a 2D non-separable filter, processing time is which is 2.77 ms. In comparison, this yields a speed-up of 4.3 when processing a 1280×960 image.

For a 9×9 rank transform (i.e. $K' = 4$) experiments showed that a block size of $tdx \times tdy = 32 \times 4$ with $n_{\mathrm{ppt}} = 4$ yields best performance. A speed up of 4.0 is obtained switching from the texture-based kernel (3.13 ms) to the shared memory kernel (0.78 ms) for 1280×960 images.

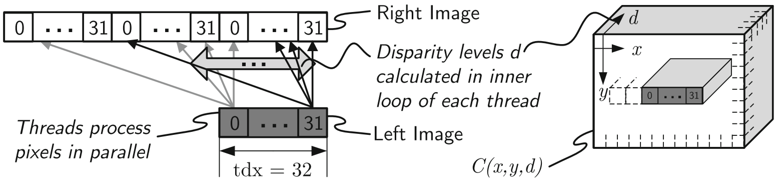
Right Image
Disparity levels d calculated in inner loop of each thread
Threads process pixels in parallel
Left Image
tdx = 32
d
x
y
C(x,y,d)
0
31

Fig. 10 Memory access scheme for calculating the matching costs for *tdx* pixels in parallel in a *tdx*-thread wide warp and $tdy = 1$. The location of the results in the 3D matching cost space is shown. Again the numbers in the *squares* represent the thread ID that fetches the according pixels from global memory

3.6.3 SGM Kernel

For every pixel location $\mathbf{p}$, calculation of the matching cost $C(\mathbf{p}, d)$ according to Eq. (4) results in a vector with one entry for each disparity level d. Thus, the spatial directions (x and y) and the disparity range span the three-dimensional disparity space. The matching cost (*MC*) calculation for every point in this space can be performed independently allowing for parallelization in all three dimensions.

A straightforward parallelization is to assign each thread with the calculation of one entry in the 3D cost space of $C(\mathbf{p}, d)$. This kernel (`mc_unaligned`) reaches 16.3 ms and 48.6 GB/s which is far from the bandwidth limit due to inefficient, often misaligned memory access, lack of data reuse, and little latency hiding possibilities. This kernel is latency bound which can only be eliminated by a new parallelization scheme.

The new kernel (`mc_proposed`) processes all disparity levels of a group of *tdx* neighboring pixels synchronously in *tdx* threads. The disparity dimension itself is further separated into *tdy* groups each processing d_{range}/tdy disparity levels with an inner loop in the kernel. By adjusting *tdy* thread parallelism is substituted with inner loop complexity. Pixels from the base image are read aligned and coalesced over the *tdx* threads. The required pixels from the right image are loaded in groups of *tdx* aligned, coalesced pixels into the shared memory where they can be accessed and reused by all threads. The parallelization scheme is shown in Fig. 10. Furthermore, only 8 bit precision is required. Since performing arithmetic in non-native GPU data types (i.e. other than 32-bit integer and float) is slow, input images and computation are based on 32-bit integer and type conversion to `uchar` is performed just before writing out the result. Consequently, type conversion to `uchar` is performed just before writing out the result. Choosing *tdx* as a multiple of the warp size (i.e. 32) results in always aligned memory access. This kernel adheres the optimization approach of Sect. 3.6.1 by providing inherently aligned memory access, high data reuse, and efficient use of the arithmetic pipeline. With an obtained performance of 1.8 ms and 111.2 GB/s this is a speed-up of factor 9.2. A performance summary is given in Table 1.

The path costs (*PC*) calculation according to Eq. (6) is performed by individually traversing along each of the eight path directions updating the matching cost values

Table 1 Performance results of the optimized kernels (*MC*: matching costs, *PC*: path costs, *WTA*: winner-takes-all) with optimal launch configuration for computing the semi-global matching algorithm for images with 1280×960 pixels and 128 disparity levels

Kernel	Time (ms)	Bandwidth (GB/s)	Bound by
MC unaligned	16.32	48.6	Parallelization scheme
MC proposed (`uchar4`)	1.80	107.3	Pipeline latency
MC+PC 8 path dir. (sequential)	75.68	20.9	Pipeline latency
MC+PC 8 path dir. (concurrent)	39.81	39.7	Pipeline latency
Sum, WTA left disp. map	15.09	117.4	Memory bandwidth

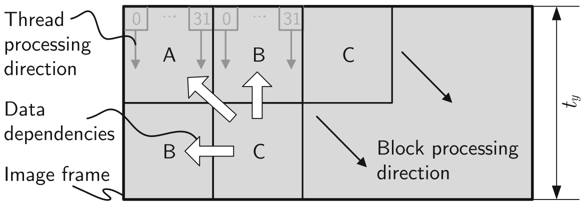

Fig. 11 Image tiling for the 45° path and $t_y = 2$ allowing divergent processing direction and path direction while tiles with the same letter can be processed in parallel. The processing direction ensures coalesced and aligned memory access

and resulting in a new 3D cost space for each path direction. PC calculation must be done sequentially along the respective path direction (e.g. from left to right) because the previous pixel's path costs must be known. The parallel minimum search over the disparity levels has been implemented similar to the parallel reduction scheme from [33]. Although the MCs are common to all PC directions and it seems obvious to separate MC and PC calculation, it is faster to integrate MC and PC calculation and recalculate the MCs on-the-fly for each PC direction. This drastically reduces pressure on the performance-limiting memory bandwidth since the MC data is never transferred via the external memory but can be kept locally in shared memory. All eight path directions are executed concurrently using the CUDA concurrent kernel execution on Fermi architectures.

Due to the coalesced memory access necessity only a group of horizontally neighboring pixels can be efficiently accessed in memory. The path costs kernels must be modified according to their path direction in order to maintain efficient memory access. For each diagonal path direction, processing is separated into rectangular tiles. Within each tile the processing direction is along the image columns, i.e. misaligned to the path direction, but ensuring aligned memory access. Tiles not sharing data dependencies can be processed in parallel as independent thread blocks. This is similar to the intrablock encoding scheme for video streams proposed in [53]. An example of the parallel processing order is shown in Fig. 11. Since block synchronization does not exist on GPUs, correct execution order is established by sequentially launching a kernel for each diagonal tile front (identical letters in Fig. 11) causing some minor time overhead. The seeming alternative

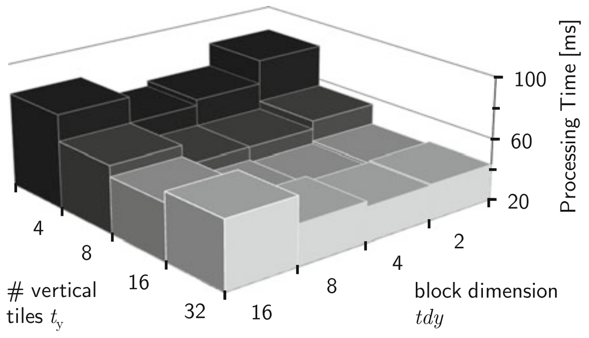

Fig. 12 Impact of the parallelization configuration on the performance of the concurrent path cost calculation for eight paths of the SGM for 1280×960 images and 128 disparity levels. Block width and tile width are both fixed to $tdx = 32$. Best performance is achieved with $tdx \times tdy = 32 \times 4$ (i.e. each inner loop processes 32 disparity levels) and $t_y = 16$

of keeping the processing implementation unchanged but rearranging the data in the memory creates an inherently contradictory situation: if the GPU is used to rearrange the data, the re-sorting causes additional memory access with is not even coalesced.

Again, parameters adjustment allows to navigate between the performance optimization principles. The first parameter (*tdy*) trades thread parallelism against sequential computation in the inner loop for all kernels. The second parameter (t_y) trades the number of parallelly processable blocks versus launch overhead and memory overhead for the four diagonal paths. Figure 12 shows the result of the parameter study. Choosing $tdy = 4$ and $t_y = 16$ results in best performance (39.8 ms and 39.7 GB/s) for a 1280×960 image. If the concurrent kernel execution is not used, performance is approximately halved (75.7 ms and 20.9 GB/s). Both kernel sets, concurrent and sequential, are latency bound.

Summation of the eight path cost spaces (7) and winner-takes-all disparity selection (9) can be performed independently for each pixel allowing for the same parallelization scheme as for the MC calculation. This kernel (`sum_wta`) requires 15.1 ms and is memory bound with 117.4 GB/s.

3.6.4 Performance

The processing time for the complete disparity estimation including rank transform, semi-global matching for eight paths, disparity map generation (without left/right check (3)) and median filtering on a Tesla C2050 Fermi architecture GPU is summarized in Table 2. Overall, a 1280×960 image with 128 disparity levels requires 56.2 ms. The processing times do not include data transfer between host and GPU because it can be effectively hidden using concurrent data transfer when processing image streams. When processing 1280×960 image sets ca. 5 ms additional transfer time is required.

Table 2 Performance results for the entire disparity estimation algorithm using rank transform, semi-global matching and median filtering on a Nvidia Tesla C2050 GPU

Image size	$d_{\text{range}} = 64$	128	256
640×480	9.7 ms	16.0 ms	29.0 ms
1024×768	21.5 ms	35.9 ms	67.1 ms
1280×960	32.9 ms	56.2 ms	105.7 ms

Results are k-mean values over multiple runs and images

3.7 Implementation Example: VLSI Architecture for Semi-global Matching

In this section a parallelization scheme and corresponding VLSI architecture for semi-global matching will be discussed. It is based on the works of [7, 8].

3.7.1 Parallelization

A crucial point for VLSI-implementation is the mapping of the algorithm into a *parallel-processable* and *stream-based* flow that only requires a single-pass across the input images. Further important aspects are *regularity* and *locality* of the architecture that implements this flow [72]. Challenges are imposed by the semi-global matching due to the recursively defined paths and their orientations within the images (see Sect. 2.3), which are not aligned to a stream-based flow.

First, the two-dimensional parallelization concept that enables stream-based processing will be introduced. Afterwards, an extension of the concept into the third dimension is presented, what significantly increases processing speed and throughput. The two-dimensional parallelization concept is shown in Fig. 13 and will be presented for the path directions of 0°, 45°, 90°, 135°. Pixels are processed from left to right along the image row (0° path). After processing pixel $\mathbf{p}_{-1} = [x-1, y]$ of the upper row, all path costs over d of all directions are available in the path costs buffers $\left(\mathrm{z}^{-1}\right)$. Path costs are delayed, according to their path directions of 90° and 45° by one and two *additional* processing steps, respectively. Afterwards, path costs of $L_{45°}(x-3, y, d)$, $L_{90°}(x-2, y, d)$, $L_{135°}(x-1, y, d)$ are available at the output of the path cost buffers. These are exactly those path costs needed for parallel and synchronous calculation of all path costs of all orientations for pixel $\mathbf{p}_2 = [x-2, y+1]$. Synchronous calculation allows direct summation of path costs in a pipeline that returns the aggregated costs S.

Therefore, all paths to the pixels $\mathbf{p}_1 = [x, y]$ *and* $\mathbf{p}_2 = [x-2, y+1]$ are calculated in parallel in a single processing step. This concept is extendable to an arbitrary number of rows. An additional delay by two pixels is introduced for each new row, as illustrated in Fig. 13. Images are separated into image slices of N parallel rows in order to process whole images. Path costs of the last row of an image slice need to be stored and made available to the first row of the next slice.

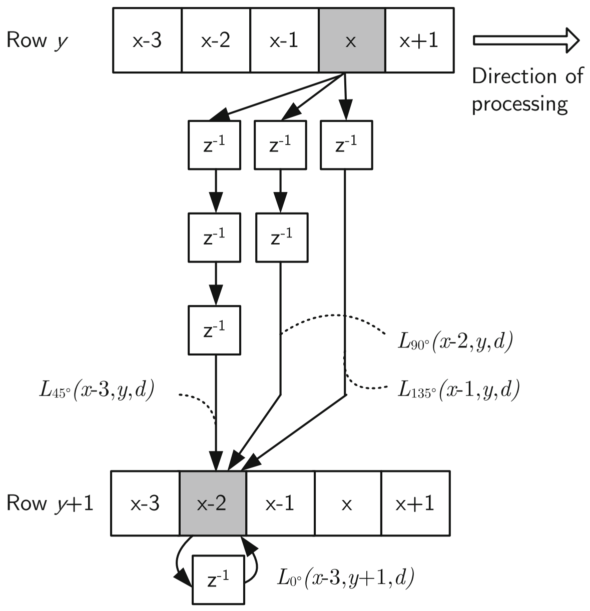
Row y
x-3
x-2
x-1
x
x+1
Direction of processing
z^{-1}
$L_{90^\circ}(x\text{-}2,y,d)$
$L_{45^\circ}(x\text{-}3,y,d)$
$L_{135^\circ}(x\text{-}1,y,d)$
Row y+1
$L_{0^\circ}(x\text{-}3,y+1,d)$

Fig. 13 Synchronized and parallel calculation of the path costs of the four paths L_{0°, L_{45°, L_{90° and L_{135° for the two pixels $\mathbf{p}_1 = [x, y]$ and $\mathbf{p}_2 = [x-2, y+1]$. Each delay element stores the respective path costs over all disparity levels for the duration of one processing step

Generalization of this concept is only limited by the fact that the maximum angle range must be within the half-closed interval $[0, 180^\circ)$. This means that no paths in opposite directions can be directly supported without additional hardware. The two-dimensional parallelization allows regular data accesses of the input images and all intermediate values and will further be referred to as row parallelism. Moreover, this concept is independent of the processing method of the disparity levels, which can be either serial or parallel. Processing the disparity levels in parallel establishes a third dimension of parallelism, which will be referred to as disparity level parallelism. An approach of particular interest for dedicated hardware implementations is not to choose either extreme (none or all disparity levels in parallel) but to process the disparity levels in small groups (e.g. 2, 4, or 8). In this case the size of the path cost buffers, as specified above, remains constant while the throughput increases linearly with the number of parallelized disparity levels. However, some additional logic for the arithmetic computation of n paths in parallel will be required. The increase of logic requirements vs. performance of disparity level parallelism and row level parallelism will be investigated in Sect. 3.7.3.

3.7.2 Architecture

The hardware architecture for the entire stereo matching algorithm is given in Fig. 14. Computation of the rank transform of both images and calculation of the data dependent penalty term P_2 is done in parallel and synchronously utilizing the same data path.

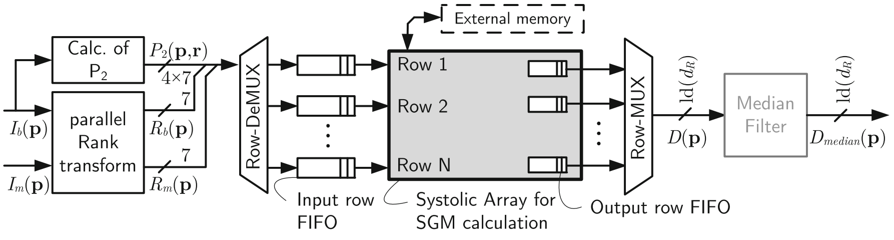

Fig. 14 Hardware architecture for calculation of disparity maps using rank-transform and semi-global matching. The median filter is optional

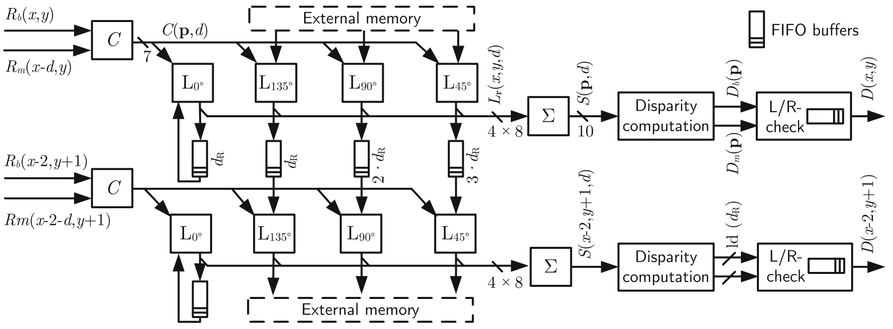

Fig. 15 Hardware architecture of the systolic array for parallel path cost calculation of the semi-global matching for two parallel rows

An N-row buffer provides this data to the systolic array, which calculates the disparities of all N rows in parallel according to the parallelization concept introduced above. As a basic post processing step, a median filter is employed for outlier suppression.

A heterogeneous, completely synchronized systolic array realizes the parallelization concept for the semi-global matching utilizing path directions from 0°, 45°, 90°, 135°. Figure 15 shows the corresponding block diagram without utilization of disparity level parallelism. In this case processing of a pixel **p** is carried out sequentially over all disparities of this pixel. The first processing elements (*C-PEs*) calculate the matching costs $C(\mathbf{p},d)$. Each of the following PEs (*L-PEs*) calculates the path costs L_r along a path **r** according to Eq. (6). The results are buffered in the appropriate path cost buffers. All L-PEs are completely identical and the path orientations are solely defined by the delays introduced by the path cost buffers. Path costs are summed to S and then processed by disparity computation PEs (*D-PEs*). D-PEs locate the minimum, i.e. the correct disparity, for the disparity maps D_b and D_m of the base and match camera, respectively. A final L/R-Check-PE projects the disparity map D_m to the perspective of the base camera, executes the left/right check including occlusion detection, and marks pixels accordingly. A local single row buffer is needed for the projection. It functions simultaneously as an output buffer.

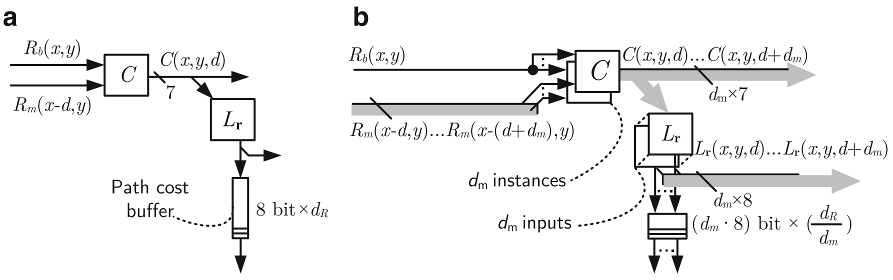
a
$R_b(x,y)$
$R_m(x\text{-}d,y)$
C
$C(x,y,d)$
7
L_r
Path cost buffer
8 bit × d_R
b
$R_b(x,y)$
$R_m(x\text{-}d,y)...R_m(x\text{-}(d+d_m),y)$
C
$C(x,y,d)...C(x,y,d+d_m)$
d_m×7
L_r
$L_r(x,y,d)...L_r(x,y,d+d_m)$
d_m instances
d_m×8
d_m inputs
$(d_m \cdot 8)$ bit × $(\frac{d_R}{d_m})$

Fig. 16 Architectural extension (**b**) of the 2D-systolic array (**a**) for introducing disparity level parallelism where d_m specifies the number of disparity levels processed in parallel

In order to introduce disparity level parallelism in addition to the row level parallelism, the C-PEs and L-PEs are extended to process several consecutive disparity levels in parallel. These groups of parallel disparity levels are processed serially. This leads to an approximately linear increase in throughput. Further, it is area efficient for two reasons. First, additional logic is only required for parts of the processing units. And second, the absolute size of local buffers does not change—only the depth-to-width ratio. This is a major advantage of disparity level parallelism. The architectural extension for disparity level parallelism is shown in Fig. 16.

Boundary treatment for pixels with missing stereo overlap (i.e. $x < d_{\max}$) significantly reduces the number of entries of the cost spaces $C(\mathbf{p},d)$, $L_r(\mathbf{p},d)$, $S(\mathbf{p},d)$, and, consequently, leads to a computing time reduction. For VGA images and a disparity range of 128 px the reduction is 9.9% (without disparity level parallelism).

An external interim memory is required for storing the path costs of the three non-horizontal paths of the last row of an image slice and providing them to the first row of the consecutive image slice. Due to the extremely regular data transfer, obeying the FIFO-principle, and the low transfer rates, external SSRAM and SDRAM-memories can be used. Alternatively, on-chip memory can be considered due to the quite low absolute memory requirements.

3.7.3 Performance

Performance of the complete system and scalability of the SGM unit are analyzed with the minimum clock frequency required to fulfill a fixed throughput constraint. This metric, i.e. the clock frequency normalized for a fixed throughput, allows direct and accurate comparison, and reflects the importance of performance while being independent from varying operating clock frequencies [70]. This also models a typical design constraint of real-world applications, where the required throughput is usually specified by external circumstances (e.g. by the cameras, required depth resolution, etc.). In this case, throughput-normalized metrics for clock frequency,

Table 3 Minimum required clock frequencies of the SGM unit (including rank transform and median filter) for a fixed resolution of 640×480 px with 128 disparity levels at 30 fps and resource usage on a Xilinx Virtex-5 FPGA

	Min. clock frequency (MHz)				LUTs			
$p_r \setminus d_m$	1	2	4	8	1	2	4	8
5	219.6	112.3	58.5	31.6	5,652	6,621	10,110	17,214
10	111.9	57.4	30.0	16.8	11,595	13,398	20,565	34,589
20	58.3	30.2	17.2	13.4	23,379	26,986	41,292	69,578
30	40.7	21.4	14.9	12.4	35,119	40,700	61,930	103,504

The number of parallel rows and parallel disparity levels is denoted p_r and d_m, respectively

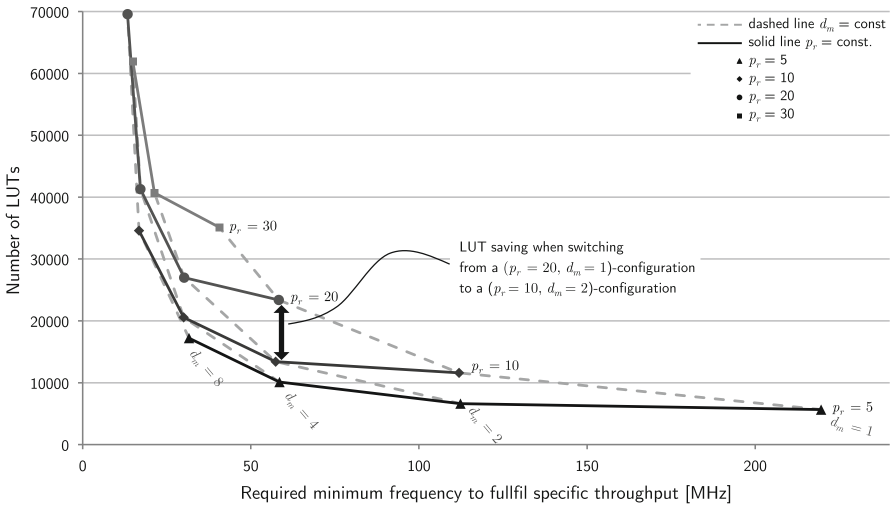

Fig. 17 Required number of LUTs of the systolic array of the SGM unit over the minimum required clock frequency to process 640×480 px with 128 disparity levels at 30 fps. The number of parallel rows and parallel disparity levels is denoted p_r and d_m, respectively. The diagrams effectively show the impact in area and performance when varying row parallelism and/or disparity level parallelism. The *lower left border* in the diagram reflects the Pareto-optimum configuration points

resource usage, power, and latency enable straightforward identification of the Pareto-optimal point of operation. Table 3 provides the results for the SGM unit for a typical parameter set of 640×480 px at 30 fps. As metric for required silicon area, only Virtex5 LUTs are used. For more information (e.g. BRAMs) please refer to [8].

Interesting insights can be gained by studying the row parallelism vs. disparity parallelism trade-off. With increasing degree of parallelism, the SGM unit can be clocked with lower frequencies at the price of higher area requirements. However, there are significant differences between row level parallelism and disparity level parallelism. Each point in Fig. 17 is a specific configuration of the design representing the LUT requirements over the normalized clock frequency. This representation

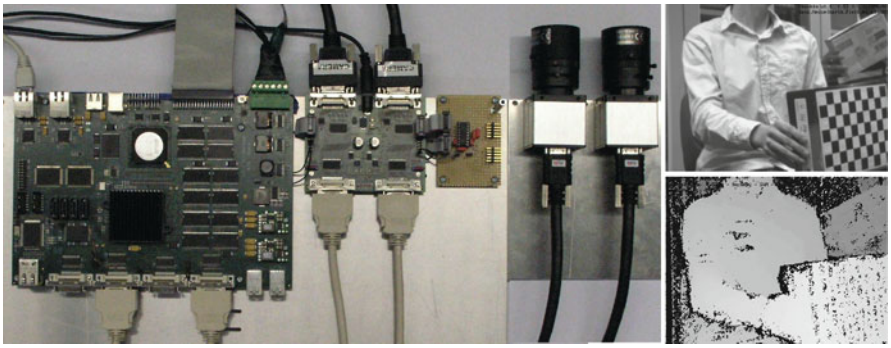

Fig. 18 Hardware setup of the stereo vision system with the system board and the stereo camera rig. On the *right* is the input image of lab scene and the computed raw disparity map before false color visualization and sending to display is conducted

is considerably different from a typical AT-diagram, which would be inadequate for this comparison as it would not reflect the throughput constraint.

For a small number of parallel disparity levels, increasing disparity level parallelism is very efficient since it has a significantly smaller influence on the total resource usage than increasing row level parallelism. However, row parallelism is the key concept for stream-based processing and crucial for a high base performance but increases linearly with the number of rows. The full potential of the parallelism approaches is exploited when using a combination of both, i.e. by using a small number of parallel rows and additionally introducing disparity level parallelism up to the configuration that does not yet require additional memory resources. For example, starting from the $(p_r = 10, d_m = 1)$-configuration a performance increase of approximately factor two can be achieved by doubling the number of either parallel rows or disparity levels. Increasing disparity level parallelism does not increase BRAM requirements (not shown, see [8]) and results in a LUT saving of factor 1.8. The major benefit of increasing disparity level parallelism is that local memory requirements remain constant for both, path costs buffers and input/output buffers.

A stereo vision system covering the entire stereo vision process including image acquisition, noise reduction, rectification, disparity estimation, post processing, and visualization has been integrated into a single FPGA. The system has been integrated on a custom build hardware platform show in Fig. 18. This work shows that it is possible to implement an algorithmically extremely high performing disparity matching algorithm in an FPGA with true real-time performance. More details on the implementation can be found in [8].

4 Summary

There has been and continues to be tremendous research in the field of computer vision, both on the algorithmic side and on the hardware side. Nowadays, many implementations for GPUs, FPGAs, ASICs, DSPs, and ASIPs are available. These cover a huge variety of algorithms and design aspects (e.g. algorithmic performance vs. silicon area). The two example implementations on the GPU and the FPGA for semi-global matching based disparity estimation show, that it is possible to realize high quality stereo correspondance search in real-time. The GPU implementation enables SGM processing with eight paths but without left/right check with more than 62 fps of images with a resolution of 640×480 and 128 disparity levels on Nvidia Fermi architecture GPUs. The VLSI architecture is scalable and allows exact adaptation to the particular application. For the same image resolution frame rates of 1.7 fps to 319 fps are achieved at a operating frequency of 133 MHz. Which of both architectures is the more suitable solution depends on the external parameters.

5 Further Reading

A detailed algorithmic overview is provided in the textbook [83] and the surveys [29,81,85]. Epipolar geometry and rectification is covered in [34,103]. The OpenCV library provides many functions for stereo processing [11]. For multi-view stereo and 3D reconstruction [83] is a good starting point.

Dedicated image processing architectures including rectification and many more are covered in [3] and RTL hardware design in [15]. Various kinds of computer architectures including GPUs are found in the newest edition of [35].

References

1. Ambrosch, K., Kubinger, W.: Accurate hardware-based stereo vision. Computer Vision and Image Understanding, Elsevier **114**, 1303–1316 (2010)
2. Arias-Estrada, M., Xicotencatl, J., Brebner, G., Woods, R.: Multiple stereo matching using an extended architecture. Proc. Field-Programmable Logic and Applications **2147**, 203–212 (2001)
3. Bailey, D.G.: Design for embedded image processing on FPGAs. John Wiley & Sons, Singapore (2011)
4. Banz, C., Blume, H., Pirsch, P.: Real-time semi-global matching disparity estimation on the GPU. Proc. IEEE Intl. Conf. Computer Vision Workshops pp. 514–521 (2011)
5. Banz, C., Blume, H., Pirsch, P.: Evaluation of penalty functions for SGM cost aggregation. Intl. Archives of Photogrammetry and Remote Sensing (2012)
6. Banz, C., Dolar, C., Cholewa, F., Blume, H.: Instruction set extension for high throughput disparity estimation in stereo image processing. Proc. IEEE Intl. Conf. Architectures and Processors Application Specific Systems pp. 169–175 (2011)

7. Banz, C., Hesselbarth, S., Flatt, H., Blume, H., Pirsch, P.: Real-time stereo vision system using semi-global matching disparity estimation: Architecture and FPGA-implementation. Proc. IEEE Intl. Conf. Embedded Computer Systems: Architectures, Modeling, and Simulation pp. 93–101 (2010)
8. Banz, C., Hesselbarth, S., Flatt, H., Blume, H., Pirsch, P.: Real-time stereo vision system using semi-global matching disparity estimation: Architecture and FPGA-implementation. Trans. High-Performance Embedded Architectures and Compilers, Springer (2012)
9. Birchfield, S., Tomasi, C.: A pixel dissimilarity measure that is insensitive to image sampling. IEEE Trans. Pattern Analysis and Machine Intelligence **20**(4), 401–406 (1998)
10. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. Proc. IEEE Intl. Conf. Computer Vision **1**, 377–384 (1999)
11. Bradski, G., Kaehler, A.: Learning OpenCV, 1 edn. O'Reilly, Sebastopol (2008)
12. Brunton, A., Chang, S., Roth, G.: Belief propagation on the GPU for stereo vision. Proc. Canadian Conf. Computer and Robot Vision p. 76 (2006)
13. Chang, N., Lin, T.M., Tasi, T.H., Tseng, Y.C., Chang, T.S.: Real-time DSP implementation on local stereo matching. Proc. IEEE Intl. Conf. Multimedia and Expo pp. 2090–2093 (2007)
14. Chang, N., Tasi, T.H., Hsu, B., Chen, Y., Chang, T.S.: Algorithm and architecture of disparity estimation with mini-census adaptive support weight. IEEE Trans. Circuits and Systems for Video Technology **20**(6), 792–805 (2010)
15. Chu, P.P.: RTL hardware design using VHDL: Coding for efficiency, portability, and scalability. Wiley-Interscience, Hoboken and N.J (2006)
16. Cornells, N., van Gool, L.: Real-time connectivity constrained depth map computation using programmable graphics hardware. Proc. IEEE Conf. Computer Vision and Pattern Recognition **1**, 1099–1104 (2005)
17. Darabiha, A., MacLean, W., Rose, J.: Reconfigurable hardware implementation of a phase-correlation stereo algorithm. Machine Vision and Applications, Springer **17**, 116–132 (2006)
18. Diaz, J., Ros, E., Carrillo, R., Prieto, A.: Real-time system for high-image resolution disparity estimation. IEEE Trans. Image Processing **16**(1), 280–285 (2007)
19. Ernst, I., Hirschmüller, H.: Mutual information based semi-global stereo matching on the GPU. Proc. Intl. Symp. Visual Computing **5358**, 228–239 (2008)
20. Ess, A., Leibe, B., Schindler, K., van Gool, L.: A mobile vision system for robust multi-person tracking. Proc. IEEE Conf. Computer Vision and Pattern Recognition pp. 1–8 (2008)
21. Faugeras, O., Viéville, T., Theron, E., Vuillemin, J., Hotz, B., Zhang, Z., Moll, L., Bertin, P., Mathieu, H., Fua, P., Berry, G., Proy, C.: Real-time correlation-based stereo: Algorithm, implementations and applications (1993)
22. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient belief propagation for early vision. Intl. Journal of Computer Vision, Springer **70**, 41–54 (2006)
23. Forstmann, S., Kanou, Y., Jun, O., Thuering, S., Schmitt, A.: Real-time stereo by using dynamic programming. Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshop p. 29 (2004)
24. Gehrig, S., Rabe, C.: Real-time semi-global matching on the CPU. Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshop pp. 85–92 (2010)
25. Gehrig, S.K., Eberli, F., Meyer, T.: A real-time low-power stereo vision engine using semi-global matching. Proc. Intl. Conf. Computer Vision Systems **5815**, 134–143 (2009)
26. Georgoulas, C., Andreadis, I.: A real-time fuzzy hardware structure for disparity map computation. Journal of Real-Time Image Processing, Springer **6**(4), 257–273 (2011)
27. Gibson, J., Marques, O.: Stereo depth with a unified architecture GPU. Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshop pp. 1–6 (2008)
28. Gong, M.: Real-time joint disparity and disparity flow estimation on programmable graphics hardware. Computer Vision and Image Understanding, Elsevier **113**(1), 90–100 (2009)
29. Gong, M., Yang, R., Wang, L., Gong Mingwei: A performance study on different cost aggregation approaches used in real-time stereo matching. Intl. Journal of Computer Vision, Springer **75**, 283–296 (2007)

30. Gong, M., Yang, Y.H.: Near real-time reliable stereo matching using programmable graphics hardware. Proc. IEEE Conf. Computer Vision and Pattern Recognition **1**, 924–931 (2005)
31. Haller, I., Nedevschi, S.: GPU optimization of the SGM stereo algorithm. Proc. IEEE Intl. Conf. Intelligent Computer Communication and Processing pp. 197–202 (2010)
32. Haller, I., Nedevschi, S.: Design of interpolation functions for subpixel-accuracy stereo-vision systems. IEEE Trans. Image Processing **21**(2), 889–898 (2012)
33. Harris: Optimizing parallel reduction in CUDA (2007). Whitepaper included in Nvidia Cuda SDK 4.0
34. Hartley, R.I., Zisserman, A.: Multiple view geometry in computer vision, 2. ed., 7. print. edn. Cambridge Univ. Press, Cambridge (2010)
35. Hennessy, J.L., Patterson, D.A.: Computer architecture: A quantitative approach, 5 edn. Morgan Kaufmann, San Francisco and Calif and Oxford (2011)
36. Hirschmüller, H.: Accurate and efficient stereo processing by semi-global matching and mutual information. Proc. IEEE Conf. Computer Vision and Pattern Recognition **2**, 807–814 (2005)
37. Hirschmüller, H.: Stereo processing by semiglobal matching and mutual information. IEEE Trans. Pattern Analysis and Machine Intelligence **30**(2), 328–341 (2008)
38. Hirschmüller, H., Scharstein, D.: Evaluation of cost functions for stereo matching. Proc. IEEE Conf. Computer Vision and Pattern Recognition pp. 1–8 (2007)
39. Hirschmüller, H., Scharstein, D.: Evaluation of stereo matching costs on images with radiometric differences. IEEE Trans. Pattern Analysis and Machine Intelligence **31**(9), 1582–1599 (2009)
40. Hosni, A., Bleyer, M., Rhemann, C., Gelautz, M., Rother, C.: Real-time local stereo matching using guided image filtering. Proc. IEEE Intl. Conf. Multimedia and Expo pp. 1–6 (2011)
41. Hosseini, F., Fijany, A., Safari, S., Fontaine, J.: Fast implementation of dense stereo vision algorithms on a highly parallel SIMD architecture. Journal of Real-Time Image Processing, Springer pp. 1–15 (2011)
42. Humenberger, M., Zinner, C., Kubinger, W.: Performance evaluation of a census-based stereo matching algorithm on embedded and multi-core hardware. Proc. Intl. Symp. Image and Signal Processing and Analysis pp. 388–393 (2009)
43. Ivanchenko, V., Shen, H., Coughlan, J.: Elevation-based MRF stereo implemented in real-time on a GPU. Workshop Applications of Computer Vision pp. 1–8 (2009)
44. Jiangbo, L., Rogmans, S., Lafruit, G., Catthoor, F.: Real-time stereo correspondence using a truncated separable laplacian kernel approximation on graphics hardware. Proc. IEEE Intl. Conf. Multimedia and Expo pp. 1946–1949 (2007)
45. Jiangbo, L., Zhang, K., Lafruit, G., Catthoor, F.: Real-time stereo matching: a cross-based local approach. Proc. IEEE Intl. Conf. Acoustics, Speech and Signal Processing pp. 733–736 (2009)
46. Jin, S., Cho, J., Pham, X.D., Lee, K.M., Park, S.K., Kim, M., Jeon, J.W.: FPGA design and implementation of a real-time stereo vision system. IEEE Trans. Circuits and Systems for Video Technology **20**(1), 15–26 (2010)
47. Kalarot, R., Morris, J.: Comparison of FPGA and GPU implementations of real-time stereo vision. Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshop pp. 9–15 (2010)
48. Kalarot, R., Morris, J., Gimel'farb, G.: Performance analysis of multi-resolution symmetric dynamic programming stereo on GPU. Proc. Intl. Conf. Image and Vision Computing New Zealand pp. 1–7 (2010)
49. Kanade, T., Yoshida, A., Oda, K., Kano, H., Tanaka, M.: A stereo machine for video-rate dense depth mapping and its new applications. Proc. IEEE Conf. Computer Vision and Pattern Recognition pp. 196–202 (1996)
50. Ke, Z., Jiangbo, L., Qiong, Y., Lafruit, G., Lauwereins, R., van Gool, L.: Real-Time and Accurate Stereo: A Scalable Approach With Bitwise Fast Voting on CUDA. IEEE Trans. Circuits and Systems for Video Technology **21**(7), 867–878 (2011)

51. Kim, J., Kolmogorov, V., Zabih, R.: Visual correspondence using energy minimization and mutual information. Proc. IEEE Intl. Conf. Computer Vision pp. 1033–1040 (2003)
52. Konolige, K.: Small vision systems: Hardware and implementation. Proc. Intl. Symp. Robotic Research (1997)
53. Kung, M., Au, O., Wong, P., Chun, H.L.: Block based parallel motion estimation using programmable graphics hardware. Proc. Intl. Conf. Audio , Language and Image Processing pp. 599–603 (2008)
54. Lee, S.H., Yi, J., Kim, J.S.: Real-time stereo vision on a reconfigurable system. Proc. Intl. Conf. Embedded Computer Systems: Architectures, Modeling, and Simulation Workshops **3553**, 299–307 (2005)
55. Liang, C., Cheng, C., Lai, Y., Chen, L., Chen, H.: Hardware-efficient belief propagation. IEEE Trans. Circuits and Systems for Video Technology **21**(5), 525–537 (2011)
56. Liang, W., Miao, L., Minglun, G., Ruigang, Y., Nister, D.: High-quality real-time stereo using adaptive cost aggregation and dynamic programming. Proc. Intl. Symp. 3D Data Processing, Visualization, and Transmission pp. 798–805 (2006)
57. Liang, W., Mingwei, G., Minglun, G., Ruigang, Y.: How far can we go with local optimization in real-time stereo matching. Proc. Intl. Symp. 3D Data Processing, Visualization, and Transmission pp. 129–136 (2006)
58. Liu, J., Xu, Y., Klette, R., Chen, H., Vaudrey, T.: Disparity Map Computation on a Cell Processor (2009)
59. van der Mark, W., Gavrila, D.: Real-time dense stereo for intelligent vehicles. IEEE Trans. Intelligent Transportation Systems **7**(1), 38–50 (2006)
60. Masrani, D., MacLean, W.: A real-time large disparity range stereo-system using FPGAs. Proc. Intl. Conf. Computer Vision Systems p. 13 (2006)
61. Mattoccia, S., Viti, M., Ries, F.: Near real-time Fast Bilateral Stereo on the GPU. Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshop pp. 136–143 (2011)
62. Mei, X., Sun, X., Zhou, M., Jiao, S., Wang, H., Zhang, X.: On building an accurate stereo matching system on graphics hardware. Proc. IEEE Intl. Conf. Computer Vision Workshops pp. 467–474 (2011)
63. Minglun, G., Ruigang, Y.: Image-gradient-guided real-time stereo on graphics hardware. Proc. Intl Conf. 3D Digital Imaging and Modeling pp. 548–555 (2005)
64. Miyajima, Y., Maruyama, T.: A real-time stereo vision system with FPGA. Proc. Intl. Conf. Field Programmable Logic And Application **2778**, 448–457 (2003)
65. Morris, J., Jawed, K., Gimel'farb, G., Khan, T.: Breaking the 'Ton': Achieving 1% depth accuracy from stereo in real time. Proc. Intl. Conf. Image and Vision Computing New Zealand pp. 142–147 (2009)
66. Mühlmann, K., Maier, D., Hesser, J., Manner, R.: Calculating dense disparity maps from color stereo images, an efficient implementation. Intl. Journal of Computer Vision, Springer **47**(1–3), 79–88 (2002)
67. Pantilie, C., Nedevschi, S.: SORT-SGM: Subpixel optimized real-time semiglobal matching for intelligent vehicles. IEEE Trans. Vehicular Technology **61**(3), 1032–1042 (2012)
68. Park, S., Jeong, H.: Real-time stereo vision FPGA chip with low error rate. Proc. Intl. Conf. Multimedia and Ubiquitous Engineering pp. 751–756 (2007)
69. Paya Vaya, G., Martin Langerwerf, J., Banz, C., Giesemann, F., Pirsch, P., Blume, H.: VLIW architecture optimization for an efficient computation of stereoscopic video applications. Proc. Intl. Conf. Green Circuits and Systems pp. 457–462 (2010)
70. Paya-Vaya, G., Martin-Langerwerf, J., Pirsch, P.: A multi-shared register file structure for VLIW processors. Journal of Signal Processing Systems, Springer **58**(2), 215–231 (2010)
71. Perez, J., Sanchez, P., Martinez, M.: High memory throughput FPGA architecture for high-definition Belief-Propagation stereo matching. Proc. Intl. Conf. Signals, Circuits and Systems pp. 1–6 (2009)
72. Pirsch, P.: Architectures for digital signal processing. John Wiley & Sons, Inc., Chichester (2008)

73. Pock, T., Schoenemann, T., Graber, G., Bischof, H., Cremers, D.: A convex formulation of continuous multi-label problems. Proc. European Conference on Computer Vision **5304**, 792–805 (2008)
74. Podlozhnyuk, V.: Image Convolution with CUDA (2007). Whitepaper included in Nvidia Cuda SDK 4.0
75. Ranft, B., Schoenwald, T., Kitt, B.: Parallel matching-based estimation - a case study on three different hardware architectures. Proc. IEEE Intelligent Vehicles Symposium pp. 1060–1067 (2011)
76. Ruigang, Y., Pollefeys, M.: Multi-resolution real-time stereo on commodity graphics hardware. Proc. IEEE Conf. Computer Vision and Pattern Recognition **1**, I–211–I–217 (2003)
77. Ruigang, Y., Welch, G., Bishop, G.: Real-time consensus-based scene reconstruction using commodity graphics hardware. Proc. Pacific Conf. Computer Graphics and Applications pp. 225–234 (2002)
78. Sabihuddin, S., Islam, J., MacLean, W.: Dynamic programming approach to high frame-rate stereo correspondence: A pipelined architecture implemented on a field programmable gate array. Proc. Canadian Conf. Electrical and Computer Engineering pp. 001,461–001,466 (2008)
79. Safari, S., Fijany, A., Diotalevi, F., Hosseini, F.: Highly parallel and fast implementation of stereo vision algorithms on MIMD many-core Tilera architecture. Proc. IEEE Aerospace Conference pp. 1–11 (2012)
80. Scharstein, D., Szeliski, R.: The Middlebury Stereo Pages. http://vision.middlebury.edu/stereo/
81. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Intl. Journal of Computer Vision, Springer **47**(1), 7–42 (2002)
82. Scharstein, D., Szeliski, R.: High-accuracy stereo depth maps using structured light. Proc. IEEE Conf. Computer Vision and Pattern Recognition **1**, I–195–I–202 (2003)
83. Szeliski, R.: Computer vision: Algorithms and applications. Springer, London and New York (2011)
84. Szeliski, R., Scharstein, D.: Sampling the disparity space image. IEEE Trans. Pattern Analysis and Machine Intelligence **26**(3), 419–425 (2004)
85. Tomasi, M., Vanegas, M., Barranco, F., Daz, J., Ros, E.: Massive parallel-hardware architecture for multiscale stereo, optical flow and image-structure computation. IEEE Trans. Circuits and Systems for Video Technology **22**(2), 282–294 (2012)
86. Tombari, F., Mattoccia, S., Di Stefano, L., Addimanda, E.: Classification and evaluation of cost aggregation methods for stereo correspondence. Proc. IEEE Conf. Computer Vision and Pattern Recognition pp. 1–8 (2008)
87. Tseng, Y.C., Chang, T.S.: Architecture design of belief propagation for real-time disparity estimation. IEEE Trans. Circuits and Systems for Video Technology **20**(11), 1555–1564 (2010)
88. Ttofis, C., Hadjitheophanous, S., Georghiades, A., Theocharides, T.: Edge-directed hardware architecture for real-time disparity map computation. IEEE Trans. Computers **PP**(99), 1 (2012)
89. Ttofis, C., Theocharides, T.: Towards accurate hardware stereo correspondence: A real-time FPGA implementation of a segmentation-based adaptive support weight algorithm. Proc. Conf. Design, Automation & Test in Europe pp. 703–708 (2012)
90. Vaish, V., Levoy, M., Szeliski, R., Zitnick, C., Sing, B.K.: Reconstructing occluded surfaces using synthetic apertures: Stereo, focus and robust measures. Proc. IEEE Conf. Computer Vision and Pattern Recognition **2**, 2331–2338 (2006)
91. Villalpando, C., Morfopolous, A., Matthies, L., Goldberg, S.: FPGA implementation of stereo disparity with high throughput for mobility applications. Proc. IEEE Aerospace Conference pp. 1–10 (2011)
92. Weber, M., Humenberger, M., Kubinger, W.: A very fast census-based stereo matching implementation on a graphics processing unit. Proc. IEEE Intl. Conf. Computer Vision Workshops pp. 786–793 (2009)

93. Woodfill, J., Gordon, G., Buck, R.: Tyzx DeepSea high speed stereo vision system. Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshop p. 41 (2004)
94. Woodfill, J., Gordon, G., Jurasek, D., Brown, T., Buck, R.: The Tyzx DeepSea G2 vision system, a taskable, embedded stereo camera. Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshop p. 126 (2006)
95. Woodfill, J., Herzen, B.v.: Real-time stereo vision on the PARTS reconfigurable computer. Proc. IEEE Symp. FPGAs for Custom Computing Machines pp. 201–210 (1997)
96. Xu, Y., Chen, H., Klette, R., Liu, J., Vaudrey, T.: Belief propagation implementation using CUDA on an Nvidia GTX 280. Proc. Advances in Artificial Intelligence **5866**, 180–189 (2009)
97. Yang, Q., Wang, L., Yang, R., Wang, S., Liao, M., Nister, D.: Real-time global stereo matching using hierarchical belief propagation. Proc. The British Machine Vision Conference pp. 989–998 (2006)
98. Yunde, J., Xiaoxun, Z., Mingxiang, L., Luping: A miniature stereo vision machine (MSVM-III) for dense disparity mapping. Proc. IEEE Intl. Conf. Pattern Recognition **1**, 728–731 (2004)
99. Zabih, R., Woodfill, J.: Non-parametric local transforms for computing visual correspondence. Proc. European Conference on Computer Vision pp. 151–158 (1994)
100. Zach, C., Klaus, A., Hadwiger, M., Karner, K.: Accurate dense stereo reconstruction using graphics hardware. Proc. EUROGRAPHICS pp. 227–234 (2003)
101. Zach, C., Sormann, M., Karner, K.: Scanline optimization for stereo on graphics hardware. Proc. Intl. Symp. 3D Data Processing, Visualization, and Transmission pp. 512–518 (2006)
102. Zatt, B., Shafique, M., Bampi, S., Henkel, J.: Multi-level pipelined parallel hardware architecture for high throughput motion and disparity estimation in Multiview Video Coding. Proc. Conf. Design, Automation & Test in Europe pp. 1–6 (2011)
103. Zhang, Z.: Determining the epipolar geometry and its uncertainty: A review. Intl. Journal of Computer Vision, Springer **27**(2), 161–195 (1998)
104. Zhang, Z.: A flexible new technique for camera calibration. IEEE Trans. Pattern Analysis and Machine Intelligence **22**(11), 1330–1334 (2000)