Stereo Vision Algorithms for FPGAs

Stefano Mattoccia
Department of Computer Science and Engineering, University of Bologna
stefano.mattoccia@unibo.it

Abstract

In recent years, with the advent of cheap and accurate RGBD (RGB plus Depth) active sensors like the Microsoft Kinect and devices based on time-of-flight (ToF) technology, there has been increasing interest in 3D-based applications. At the same time, several effective improvements to passive stereo vision algorithms have been proposed in the literature. Despite these facts and the frequent deployment of stereo vision for many research activities, it is often perceived as a bulky and expensive technology not well suited to consumer applications. In this paper, we will review a subset of state-of-the-art stereo vision algorithms that have the potential to fit a target computing architecture based on low-cost field-programmable gate arrays (FPGAs), without additional external devices (e.g.*, FIFOs, DDR memories, etc.). Mapping these algorithms into a similar low-power, low-cost architecture would make RGBD sensors based on stereo vision suitable to a wider class of application scenarios currently not addressed by this technology.*

1. Introduction

In recent years, with the widespread diffusion of 3D sensors, there has been increasing interest in consumer and research applications based on dense range data. Some of these sensors provide a depth map and an RGB (or monochrome) image of the sensed scene. For this reason, these devices are often referred to as RGBD (RGB plus Depth) sensors. A well-known and representative example of such devices is the Microsoft Kinect, a cheap and accurate RGBD sensor based on structured light technology. Since its presentation in 2010, it has been deployed in many scientific and consumer applications. This technology relies on a standard colour camera, an infrared projector, and an infrared camera. The projected pattern is sensed by the infrared camera and analysed according to a patented technology in order to infer depth. The Kinect enables the user to obtain accurate depth maps and images at VGA resolution in indoor environments. Another interesting technology that gained popularity in recent years is *Time of Flight* (ToF) technology. In this case, the sensor emits a modulated light; by measuring the time required to receive the bounced light, it is possible to infer depth. This technology also provides a monochrome image of the sensed scene and hence belongs to the class of RGBD sensors. However, compared to the Kinect technology, ToF currently provides a depth map and images at a reduced resolution. In both cases, especially for ToF sensors, there have been attempts to improve their performance by means of sensor fusion techniques (*e.g.* [2]) combining the depth map provided by these sensors with images acquired with registered high-resolution cameras. The Kinect and ToF technologies have specific strengths and limitations [13]; however, both sensors are ill-suited to environments flooded with sunlight (the Kinect in particular becomes useless in these circumstances).

Stereo vision is a well-known technology for inferring depth and, excluding projection-based approaches, it is a passive technology based on standard imaging sensors. Stereo vision systems infer dense depth maps by identifying corresponding projections of the same 3D point sensed by two or more cameras in different positions. This challenging task, often referred to as the *correspondence problem*, can be tackled with many algorithms, and consequently produces different outcomes in terms of accuracy and computational complexity. This means that, in stereo vision, the algorithm aimed at tackling the correspondence problem plays a major role in the overall technology. Of course, stereo vision systems intrinsically provide RGB or monochrome images, and thus belong to the class of RGBD sensors. Compared to active technologies, stereo vision may provide unreliable results in regions where the correspondence problem becomes ambiguous (*e.g.*, in poorly textured regions). However, compared to active technologies, it is suited to both indoor and outdoor environments, as well as to close-range and long-range depth measurements. Finally, being a passive technology, multiple stereo vision sensors sensing the same area do not interfere with each other. Nevertheless, despite its positive aspects and widespread deployment in many research applications in the last few decades, stereo vision is often perceived as a

bulky and expensive technology not suited to mainstream or consumer applications. In this paper, we will try to address this concern by outlining a simple computing architecture based on a low-cost FPGA. We will also review stereo-matching algorithms that have the potential to entirely fit within this constrained architecture without any other external device (e.g., FIFOs, DDR memories, etc.), with the exception of a high-speed communication controller. The topic addressed in this paper is related to on-going research activity aimed at developing a cheap, accurate, self-powered RGBD sensor based on stereo vision technology, deploying as a computing platform only the reconfigurable logic available in standard low-cost FPGAs.

2. Related Work

Dense stereo vision has been a widely researched topic for decades [15] and, due to its highly demanding computational requirements, many different computing platforms (*e.g.*, CPUs, GPUs, DSPs, FPGAs, ASICs, etc.) have been deployed to obtain depth maps in real time. However, some of these computing architectures, such as those based on standard CPUs or GPUs, are currently not well-suited to consumer/embedded applications due to their high power requirements, cost, and size. Computing architectures, such as those based on high-end FPGAs, are often too expensive, while solutions based on custom application-specific integrated circuits (ASICs), despite the limitations regarding their reconfigurability and *time to market* compared to FPGAs, represent a less-expensive solution in large volumes. Finally, we point out that interesting low-power, low-cost, reconfigurable architectures for real-time dense stereo vision are represented by those based on embedded CPUs coupled with integrated DSPs, such as the OMAP platform, as reported in [5]. A recent and detailed review of stereo vision algorithms for different computing architectures can be found in [16]. In this paper, we will consider a simple computing architecture based entirely on low-cost FPGAs that, in our opinion, represent an optimal solution to obtaining compact, low-cost, low-power 3D sensors based on stereo vision.

2.1. Field Programmable Gate Arrays

FPGAs can be configured, and in most cases reconfigured many times, by means of hardware description languages (HDLs) such as VHDL or Verilog. The internal structure of an FPGA consists of a large amount of *logic cells* containing a small amount of elementary logic blocks (*e.g.*, Flip-Flops, multiplexers, and lookup tables). Also distributed into the FPGA are small multi-port memories, often referred to as *block RAM*, with fast access time. Moreover, modern low-cost FPGAs often integrate configurable DSPs for efficient arithmetic operations, clock managers, and high-speed transceivers. All these components can be configured by programmers/designers according to their specific requirements by using HDLs. For instance, considering a Xilinx Spartan 6 Model 45 FPGA, we can find roughly 44,000 logic cells, 116 dual-port block RAMs (18Kb each), 58 DSPs, 4 clock managers, and 358 configurable I/O pins. It is worth noting that the reconfigurable logic of an FPGA can be configured/programmed with HDLs at a higher level of abstraction using a *behavioural* programming methodology. However, mapping computer vision algorithms on the reconfigurable logic is not as simple as mapping the same algorithms on CPUs with traditional high-level languages. Nevertheless, recent years have seen the appearance of effective high-level synthesis (HLS) tools that enable the automatic conversion of code written in a standard programming language, such as C/C++ or Matlab, into HDLs. Despite this positive fact, because of the hardware resources of the reconfigurable logic being constrained, a clear understanding of the overall FPGA architecture and its resources is crucial to writing optimized code with HDLs, as well as with HLS tools. Thanks to its complete reconfigurability, an FPGA can be programmed to massively exploit parallelism, enabling one to obtain the optimal performance/Watt.

2.2. Stereo vision

Stereo vision [15] is a technique aimed at inferring dense or sparse depth maps from two or more views of the same scene observed by two or more cameras. Although increasing the number of cameras has the potential to improve accuracy and reliability, the binocular setup (*i.e.*, deploying two imaging sensors) is frequently employed in practice. Due to the many applications that can take advantage of range data, this topic has received constant research interest in the last few decades, and significant algorithmic improvements have been proposed in recent years [15, 9]. However, most dense stereo vision algorithms are computationally demanding, and parallel computing architectures are in most cases mandatory if one is to obtain dense depth measurements in real time. Not surprisingly, FPGAs have attracted the interest of many researchers working in this field [16].

3. Target Computing Architecture

Our target computing architecture is depicted in Figure 1. It is based on a single FPGA and aims to obtain dense depth maps at more than 30 fps with WVGA (752×480) stereo pairs and with minimal power requirements (*e.g.*, less than 2.5 Watt provided by a standard USB 2.0 port), size, and costs. Observing the figure, we can see that the two synchronized colour or monochrome imaging sensors are connected, through two *low-voltage differential signaling* (LVDS) channels for clocks and pixels, to the FPGA. This choice, plus the additional LVDS link between the two imaging sensors, enables us to put the sensors and the computing platform in arbitrary positions, even at distances of

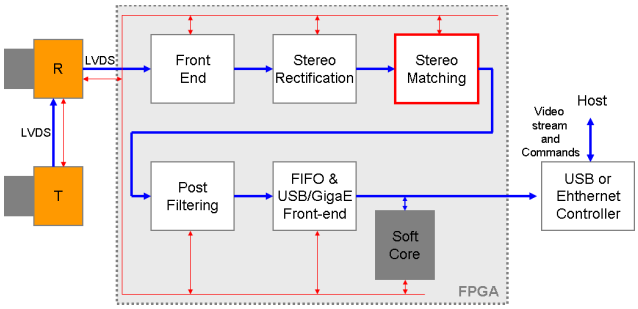

Figure 1. Basic architecture of the target computing platform. The overall design contains the imaging sensors (*e.g.*, at WVGA resolution), a low-cost FPGA (*e.g.*, a Spartan 6 Model 45), and an external high-speed communications controller (*e.g.*, USB 2.0 or 3.0, GigaEthernet). The overall processing pipeline, including the FIFO aimed at handling transfers to/from the high-speed communications controller and a supervisor soft core, is synthesized into the reconfigurable logic of the FPGA.

meters, in order to deal with different setups suited to different application requirements.

Figure 1 also shows the main steps executed by a practical stereo vision system. Once the raw images provided by the image sensors are sent to the FPGA, they are rectified in order to compensate for lens distortions. Moreover, the raw stereo pair is put in *standard form* (*i.e.*, *epipolar lines* are aligned to image *scanlines*). Both steps require a *warping* for each image of the stereo pair, which can be accomplished by knowing the *intrinsic* and *extrinsic* parameters of the stereo camera. Both parameters can be inferred by means of an off-line calibration procedure. Once the rectified images are in standard form, potential corresponding points are identified by the stereo matching module. Unfortunately, since not all of the correspondences found by the previous module are reliable, outlier detection is crucial. This step typically consists of multiple tests aimed at enforcing constraints to the inferred disparity maps (*e.g.*, left-right consistency check, uniqueness constraint), the input images, or the matching costs computed by the previous matching engine. The filtered disparity map is then sent to a FIFO connected to the external communications controller by means of control logic synthesized in the FPGA. The host computer, once it has received the disparity map, will compute depth by triangulation according to the parameters inferred with calibration. Although in this paper we will focus our attention on the stereo matching module, the overall goal consists of mapping all the blocks depicted in Figure 1 into a low-cost FPGA, such as the Xilinx Spartan 6 Model 45 or better. A similar design would allow for small cost, size, weight, power requirements, and reconfigurability. Moreover, the upgrade of the whole project to newer FPGAs (typically cheaper and with better performance in terms of speed and power consumption than the previous generation) is almost straightforward. Finally, we point out that, with the availability of integrated solutions based on reconfigurable logic, plus embedded processors such as the Xilinx Zynq [17], a self-contained FPGA module for the depth map engine would make feasible the design of a fully embedded 3D camera with complete on-board processing.

4. Stereo vision algorithms suited to the constrained computing architecture

A computing architecture similar to that outlined in the previous section poses significant constraints to the computational structure of the algorithms that can be implemented. In fact, considering a representative case study of the Xilinx Spartan 6 FPGA family [17], we can see that the overall block memory available is about 261 KB for the Model 45 (and about 600 KB for the Model 150). This means that, ignoring other requirements, we would not even be able to store a stereo pair at WVGA resolution (about 720 KB) in the 150 device. This observation, plus the limited overall reconfigurable logic available (about 43,000 and 147,000 logic cells for the Model 45 and 150, respectively), dictates that *stream processing* is mandatory. This technique consists of processing pixels as soon as they are available from the imaging sensors, with minimal buffering requirements. Of course, for the same reason, the resulting output cannot be stored in the FPGA and must be sent to the communications controller as soon as it is made available by the processing pipeline. We also point out that another relevant constraint is concerned with the overall reconfigurable logic available for processing (*e.g.*, about 55,000 flip-flops for the Model 45 and 185,000 flip-flops for the Model 150).

In the next sections, we will consider some relevant stereo vision algorithms potentially suited to this constrained target architecture. For this purpose, we will adopt the classification proposed in [15], where algorithms are classified into two major categories, local approaches and global approaches, making some further distinctions when dealing with approaches not completely described by these two broad categories.

4.1. Local approaches

Local approaches do not enforce an explicit smoothness constraint on the target disparity map, and, typically, for each disparity candidate, compute the matching cost by aggregating neighbouring pixels (on rectangular patches referred to as *support* windows, as depicted in Figure 2). Cost aggregation is often explicitly done by summing up the matching costs within the support window, as depicted in Figure 2. However, it is worth noting that some recent approaches implicitly aggregate costs in constant time, independently of the support size [14, 3]. In local algorithms, the best disparity for each point can be identified according to a simple *winner-takes-all* (WTA) strategy finding the

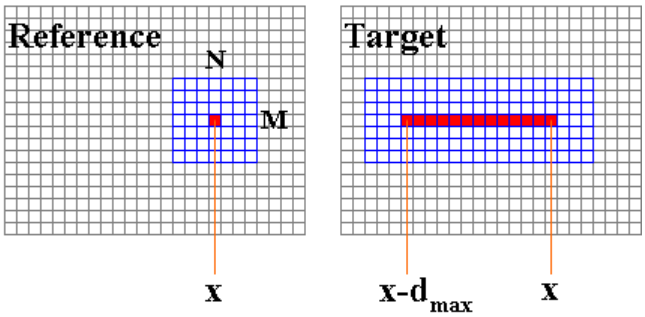

Figure 2. Support windows, of size M $\times$ N, for cost aggregation in local algorithms. In the reference image the support window is centered on point x) while in the target image the support windows are centered on points [x,x-d_{max}].

candidate with the best aggregated cost. For the reasons outlined previously, local algorithms are inherently parallel and, hence, are ideal candidates for implementation in FPGAs. Detailed reviews and evaluations of local stereo algorithms can be found in [15, 9, 16], while an exhaustive review and evaluation of cost functions suited to practical stereo vision systems can be found in [8].

4.1.1 Fixed Window algorithm

In spite of their simplicity and intrinsic parallel nature of local algorithms, even the mapping of the simplest one to the constrained target architecture should be carefully optimized. This algorithm is often referred to as fixed window (FW) or *block matching* and simply sums up all the matching costs within the support window. With a support of size M $\times$ N and a disparity range of $[0, d_{max}]$, the number of arithmetic operations is proportional to M $\times$ N $\times$ $(d_{max} + 1)$. Considering that plausible values for these parameters could be M = 15, N = 15, and d_{max} = 63, the number of arithmetic operations required might easily exceed the hardware resources available in the target reconfigurable logic. However, the number of operations can be significantly reduced by adopting well-known incremental calculation techniques [11]. For example, Figure 3 shows that, for a single image of the stereo pair, the overall cost aggregation for the support depicted in the left side can be obtained more efficiently by deploying a 1D optimization. In fact, the aggregate costs required by the operations in the left side of Figure 3 can be reduced by observing that the overall cost for the central point in red can be obtained by updating the overall cost computed in the previous position along the scanline, adding the aggregated costs in green, and subtracting the aggregated costs in blue. In deploying this 1D optimization strategy, the number of operations is reduced by a factor of M. Nevertheless, a further reduction can be obtained by deploying a 2D incremental scheme that stores intermediate results for each column. In this case, the

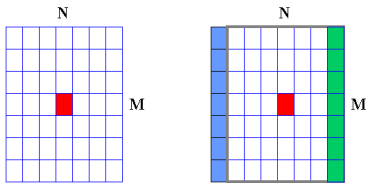

Figure 3. An incremental 1D technique aimed at reducing the number of basic operations for cost aggregation in FW (full-cost computation, left, and 1D optimization, right).

number of operations per window is constant, though at the expense of the memory footprint.

4.2. Algorithms based on adapting weights

Although the FW approach is widely used in practical applications, it is clearly outperformed by more recent approaches based on cost aggregation techniques that aggregate costs according to weights assigned by examining the image content [19, 10, 12, 9]. In these approaches, the overall score is given by a weighted sum of the costs within each support window. The key idea behind this strategy consists of weighting each cost according to its *relevance* with respect to the point under examination (*i.e.*, the central points of the supports).

Many methodologies to assign weights have been proposed in the literature, although one common rationale is that inspired by bilateral filtering [19]. That is, points with *similar* intensity with respect to the central point should be more influential in the weighted sum. Moreover, points *closer* to the central point should also be more relevant. This strategy is similar to the weight computation deployed by bilateral filtering and weights are often computed within the support window of reference and target images (this strategy is often referred to as *joint* or *symmetric*). A first optimization [9, 12] consists of asymmetrically computing weights, examining only the image points belonging to the reference image. Although this strategy significantly reduces weight computation by a factor of d_{max}, the number of operations required for cost aggregation is always proportional to M $\times$ N $\times$ $(d_{max} + 1)$ and may exceed the resources available in the target FPGA. However, simplified yet effective strategies based on the computation of weights and/or costs and/or overall weighted costs only in sampled points may help to further reduce the number of elementary operations per point, maintaining high accuracy. These approaches also exploit massively incremental calculation schemes for cost computation, similar to those outlined for FW. An approach that efficiently computes weights, on a sparse regular grid, and aggregated costs on a block basis, by means of [11], is [10]. In [12], further optimizations have been proposed, including the *pre-selection* of potential candidate disparities and asymmetric weight computations. It is worth noting that local algorithms were recently proposed

[14, 3] that filter matching costs according to the guided filtering technique [6], implicitly performing a weighted cost aggregation in constant time. Such approaches massively exploit incremental calculation techniques [11] potentially suited to the constrained FPGA architecture thanks to the reduced (and constant) number of operations required with respect to explicit cost aggregation approaches inspired by bilateral filtering. Despite these positive facts, the results provided by these constant time algorithms are comparable to those based on explicit cost aggregation and, hence, these algorithms are potentially suited to implementation in the outlined target platform.

4.3. Algorithms based on unconstrained supports

An interesting local approach was proposed in [1]. This technique, by performing multiple 1D cost aggregations constrained by an *information permeability* term (see [1] for details), efficiently enables us to adaptively aggregate costs within unconstrained 2D support windows. The information permeability term weighs and aggregates the matching cost according to the intensity differences between adjacent points. Initially this aggregation is independently performed along horizontal scanlines (from left to right and right to left). Then, a similar approach is applied along vertical directions (from top to bottom and bottom to top) to the summed aggregated matching cost computed along horizontal paths. Although this strategy requires us to store the entire image and matching costs, a simplification of the original approach restricted to subset scanlines (*e.g.*, leftright, rightleft, and topbottom) could be feasible for the constrained target computing architecture.

Finally, a different and effective algorithm that, similarly to the previous method, does not explicitly define a fixed support window was proposed in [18]. In this approach, the matching costs are aggregated using as weights the minimum intensity distance between any two points in the reference image. These weights are stored in a tree structure and, to this aim, a minimum spanning tree (MST) is created containing a number of nodes equal to the number of image points. This enables us to very efficiently and in constant time obtain for each point the aggregated weighted cost computed on the whole image. Nevertheless, in its original formulation, this approach, due to the memory footprint required to store the MST, seems inappropriate to a target computing architecture without being provided external memory devices, such as DDR memory.

4.4. Semi global approaches

Although global algorithms [15] that enforce disparity constraints on the whole image by determining the disparity assignment that minimizes global energy functions often outperform other approaches, their iterative computational structures have high memory footprint. Therefore, they do not seem to be suited to the outlined constrained target architecture.

Nevertheless, a subclass of these algorithms that enforce disparity constraints on 1D domains by means of dynamic programming or scanline optimization (SO) represents a viable and effective alternative to local approaches for the target computing architecture. In particular, the semi-global matching algorithm [7] computes multiple energy terms by means of the SO technique, independently enforcing 1D smoothness constraints along different paths (typically 8 or 16). These energy terms are then summed up and the best disparity is determined by means of a WTA strategy. SGM is very effective and is deployed in many practical applications. However, in its original formulation, due to its high memory footprint, it is not suited to a computing architecture without a large amount of external memory. Nevertheless, the SGM algorithm becomes suitable in deploying a subset of the original paths (*e.g.*, only four paths) with acceptable performance degradation for our target platform.

5. Experimental results

In this section, we report preliminary experimental results concerned with the implementation of three stereo vision algorithms - belonging to the three classes defined in the previous section - in the outlined computing architecture made of a single FPGA without additional external devices, with the exception of the high-speed communications controller as depicted in Figure 1. Each of these algorithms, as well as all of the other blocks depicted in the figure, was mapped on a Spartan 6 Model 45 FPGA and delivers depth maps at more than 30 fps when processing stereo pairs at WVGA resolution. Specifically, the algorithms currently implemented in the target FPGA are: FW, a modification of [1] using only two paths and a modification of [7] using four paths. Each implementation also includes image rectification, a pre-filtering step based on the Sobel filter, and a post-processing step aimed at filtering outliers by detecting uniform regions as well as by detecting violations of the uniqueness constraint.

For evaluation purposes we provide in Figure 4 experimental results concerned with the three implemented algorithms processing frame ♯66 of the KITTI dataset [4]. Observing the figure, where in the disparity maps warmer colours encode points close to the camera and colder colours farther points, we can see that all three algorithms enable us to obtain dense and fairly accurate disparity maps of this challenging stereo pair. Observing the trees in the right side of the reference frame, we can notice that the SGM algorithms seems less noisy compared to the local algorithms. In the same figure we can also notice occlusions and uniform regions (*e.g.* shadows) detected by the post-filtering modules.

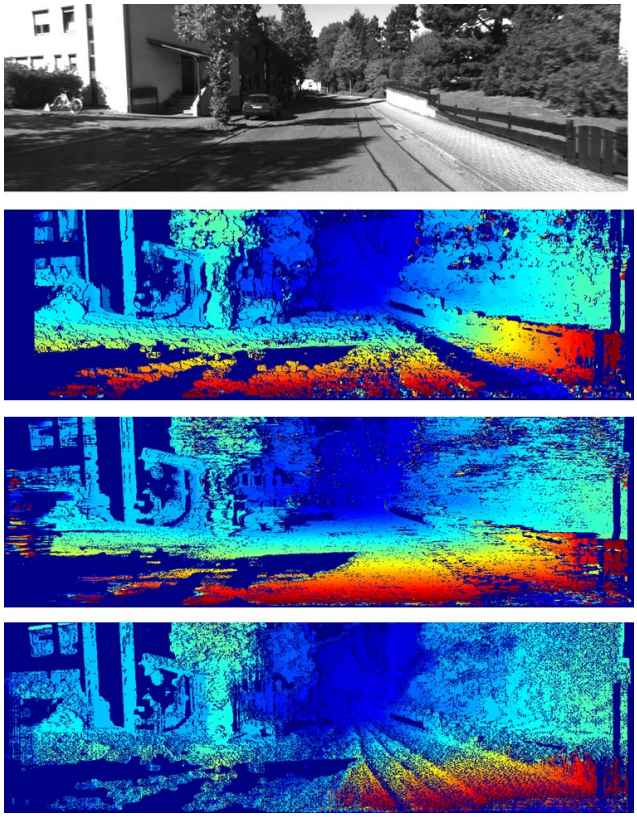

Figure 4. Preliminary experimental results for three algorithms implemented in the target computing architecture (Spartan 6 Model 45). Results are concerned with frame ♯66 of the KITTI dataset [4], using a simple Sobel filter as a pre-filtering step. From top to bottom: rectified reference image, disparity map computed by the FW implementation, disparity map computed by a modified version of the [1] algorithm using only two paths, and disparity maps computed by a modified version of the SGM [7] algorithm using four paths.

6. Conclusions

In this paper, we have reviewed stereo vision algorithms suited to a simple computing architecture made of a single low-cost FPGA without additional external devices. These stereo vision algorithms provide accurate depth maps in real time at a WVGA resolution enabling us to to obtain a low-power, low-cost RGBD stereo vision sensor self-contained in an FPGA. Future work is aimed at mapping algorithms for feature detection and description as well as at processing images at higher resolution deploying the same computing platform outlined in this paper.

Acknowledgments

I'd like to thank Ilario Marchio, Marco Casadio, Stefano Bruciati, Michael Cavina, Simone Calisesi and Marco Destro for the experimental results reported in this paper.

References

[1] C. Cigla and A. A. Alatan. Efficient edge-preserving stereo matching. In *ICCV 2001 Workshops*, pages 696–699, 2011.

[2] C. Dal Mutto, P. Zanuttigh, S. Mattoccia, and G. Cortelazzo. Locally consistent tof and stereo data fusion. In *2nd Workshop on Consumer Depth Cameras for Computer Vision*, ECCV'12, pages 598–607, 2012.

[3] L. De-Maeztu, S. Mattoccia, A. Villanueva, and R. Cabeza. Linear stereo matching. In *ICCV 2011*, pages 1708–1715, 2011.

[4] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR 2012*, Providence, USA, June 2012.

[5] S. Goldberg and L. Matthies. Stereo and imu assisted visual odometry on an omap3530 for small robots. In *ECVW 2011*, pages 169–176, 2011.

[6] K. He, J. Sun, and X. Tang. Guided image filtering. In *ECCV 2010*, pages 1–14, 2010.

[7] H. Hirschmüller. Stereo processing by semiglobal matching and mutual information. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(2):328–341, 2008.

[8] H. Hirschmüller and D. Scharstein. Evaluation of stereo matching costs on images with radiometric differences. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(9):1582–1599, 2009.

[9] A. Hosni, M. Bleyer, and M. Gelautz. Secrets of adaptive support weight techniques for local stereo matching. *Computer Vision and Image Understanding*, 117(6):620–632, 2013.

[10] S. Mattoccia, S. Giardino, and A. Gambini. Accurate and efficient cost aggregation strategy for stereo correspondence based on approximated joint bilateral filtering. In *ACCV 2009*, pages 23–27, 2009.

[11] M. Mc Donnel. Box-filtering techniques. *Computer Graphics and Image Processing*, 17:65–70, 1981.

[12] D. Min, J. Lu, and M. N. Do. A revisit to cost aggregation in stereo matching: How far can we reduce its computational redundancy? In *ICCV 2011*, pages 1567–1574, 2011.

[13] C. D. Mutto, P. Zanuttigh, and G. M. Cortelazzo. *Time-of-Flight Cameras and Microsoft Kinect(TM)*. Springer Publishing Company, Incorporated, 2012.

[14] C. Rhemann, A. Hosni, M. Bleyer, C., and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. In *CVPR 2011*, pages 3017–3024, 2011.

[15] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 2010.

[16] B. Tippetts, D. J. Lee, K. Lillywhite, and J. Archibald. Review of stereo vision algorithms and their suitability for resource-limited systems. *Journal of Real-Time Image Processing*, 2013.

[17] Xilinx. www.xilinx.com. `www.xilinx.com`.

[18] Q. Yang. A non-local cost aggregation method for stereo matching. In *CVPR 2012*, pages 1402–1409, 2012.

[19] K. Yoon and I. Kweon. Adaptive support-weight approach for correspondence search. *IEEE Trans. PAMI*, 28(4):650–656, 2006.