FPGA Accelerator for Stereo Vision using Semi-Global Matching through Dependency Relaxation

Shashwat Shrivastava*, Ziaul Choudhury*, Shashwat Khandelwal* and Suresh Purini†
Computer Systems Group, International Institute of Information Technology
Hyderabad, India
Email: *{shashwat.shrivastava, ziaul.c, shashwat.khandelwal}@research.iiit.ac.in †suresh.purini@iiit.ac.in

***Abstract*—In this paper, we propose a fully parallel and pipelined architecture for stereo vision on FPGAs using Semi-Global Matching with Census Transform being used underneath. Further, we extend the above streaming architecture so that multiple pixels can be processed in a data parallel fashion. We expose this data parallelism through dependency relaxation. This establishes a trade-off between accuracy and throughput of the hardware. We tested the proposed architecture on Virtex-7 FPGA using KITTI 2012 and KITTI 2015 datasets. On images of resolution 1280x960, with 64 disparity levels, we are able to run our hardware design at 100 MHz. At this frequency, our design is able to process 322 frames per second which is 1.6 times faster than the state-of-the-art SGM implementation on FPGA. Our system can be scaled to a higher resolution image.**

***Index Terms*—Stereo Vision, FPGAs, Semi-Global Matching**

I. Introduction

Stereo vision involves recovering the 3D structure of a scene from a pair of 2D images of the same scene, each acquired using a camera from a different viewpoint in space, at the same time. This requires finding a corresponding pixel from the left image for every pixel in the right image. We simplify this problem by rectifying the stereo pair such that the two corresponding pixels lie on the same scan line. The distance between a pixel in the right image and its corresponding pixel in the left image is called its *disparity*. If we know the focal length of the stereo cameras, then we can compute the depth of a 3D point which got projected onto the corresponding pixel pair. Thus the goal is to construct a disparity map $D(x, y) \in \{0, \cdots, d_{max} - 1\}$, for some constant d_{max}, from a rectified stereo pair of images. This has applications in the fields such as robotics and driver-less cars which require an accurate estimation of the surrounding 3D geometry.

Stereo correspondence algorithms can be broadly classified into local, global and semi-global methods [1], [2]. In local methods, the disparity of pixels in a right image is computed by matching its neighborhood window with that of others on the same scan line from the left image using distance metrics such as sum of squared intensity differences (SSD), sum of absolute intensity differences (SAD) and census transform [3]. In practice, disparity maps computed using local methods end up containing streaky artifacts. Global methods attempt to construct smooth disparity maps by introducing suitable penalties for discontinuities. This is modeled by defining an energy function over the disparity map as follows.

$$E(D) = E_{data}(D) + E_{smooth}(D)$$

The objective is to find a disparity map which minimized the above objective function. The E_{data} function captures the disparity map as computed by local methods and the E_{smooth} function captures the smoothness constraints between the neighborhood pixels. Global methods are NP-hard and there are no known computationally efficient algorithms for the same.

Semi-Global Matching (SGM) methods [2] attempt to approximate the global methods by expressing the 2D smoothness constraints as an aggregation of multiple 1D constraints representing different directions. This makes them computationally tractable while giving a high-quality disparity map. In this paper, we propose a novel parallel pipelined architecture for high throughput stereo vision which is based on the SGM algorithm proposed by Hirschmuller [2]. This algorithm which is described in Section I-A forms the basis for the proposed hardware architecture presented in Section II. The following are the main contributions of our work.

1) We propose a novel parallel pipelined architecture for disparity map computation in a streaming fashion. The architecture is a realization of Semi-Global Matching technique with Census Transform being used as a local method.
2) We extend the aforementioned architecture so that multiple pixels within a same row can be processed in parallel. We call this data parallelism as pixel-level parallelism. We establish a trade-off between accuracy of the disparity map computed and achievable throughput through pixel-level parallelism.

A. Modified SGM using Census Transform

SGM associates for a pixel p, a matching cost $S(p, d)$, if it is assigned a disparity value d. The disparity of a pixel p is then defined as follows.

$$D(p) = \underset{0 \leq d \leq d_{max}}{\operatorname{argmin}} \; S(p, d) \tag{1}$$

978-1-7281-9902-3/20/$31.00 ©2020 IEEE
DOI 10.1109/FPL50879.2020.00057

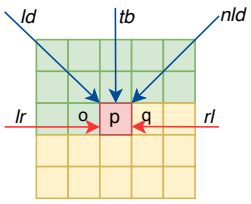


Fig. 1: All the 1D path directions used in disparity map computation in our work.

The matching cost $S(p,d)$ is obtained by aggregating the matching cost from multiple 1D directions. Let $L_r(p,d)$ denotes the matching cost of a pixel p in the direction r if it is assigned a disparity d. Then

$$S(p,d) = \sum_r L_r(p,d) \tag{2}$$

The matching cost $L_r(p,d)$ is in turn defined using the following recursive equation.

$$L_r(p,d) = CT(p,d) + \min \left\{ \begin{array}{c} L_r(p-r,d) \\ L_r(p-r,d\pm 1) + P1 \\ \min\limits_k L_r(p-r,k) + P2 \end{array} \right\} - \min_k L_r(p-r,k) \tag{3}$$

The function CT is computed by applying a census transform on the neighborhood windows at the pixels $p = (x,y)$ and $p' = (x+d,y)$, and then taking a hamming distance between the transformed windows. The census transform of a window is obtained by comparing a pixel with the central pixel and the corresponding cell in the output window is set to 1 if its intensity is greater otherwise to 0. While we used neighborhood windows of dimensions 7x7, Figure 3 illustrates the computation of CT function between two 3x3 windows. P_1 and P_2 are the penalty terms which account for disparity discontinuity along the 1D path. The accuracy of the SGM method is proportional to the number of paths used to estimate disparity. Y. Li et al. [4] used five paths in place of four paths and showed that the accuracy increased by 20%. In this paper, we have also used 5 directions $r \in \{lr, ld, tb, nld, rl\}$ which are depicted in Figure 1. Note that in Equation (3), $p-r$ denotes the previous pixel in the direction r. For example, if $p=(x,y)$, then $p-r$ in lr direction denotes $(x-1,y)$ and in ld direction it would be $(x-1,y-1)$.

II. Proposed Architecture

A. Overall Design

We propose a heterogeneous CPU+FPGA stereo vision system for disparity map computation using Semi-Global Matching. The host CPU streams the stereo pair of images to the FPGA through the PCIe bus. The architecture of the accelerator itself is oblivious to whether the streaming is happening over PCIe bus or directly from a stereo camera through some other protocol. Figure 2 shows the overall architecture of our proposed stereo vision accelerator. For notational simplicity, let $S(p)$ denote the vector $\langle S(p,0), \cdots, S(p,(d_{max}-1))\rangle$ (refer Equation (2)). Analogous definitions hold for $L_r(p)$ and $CT(p)$ (refer Equation (3)). Then for a given pixel, the CT module in Figure 2 is computing the $CT(p)$ vector which involves computing the census transform and hamming distance for multiple disparity levels $[0, d_{max}-1]$. Each module $\mathrm{SGM_r}$ computes the vector $L_r(p)$, for $r \in \{lr, ld, tb, nld, rl\}$. We call them as Semi-Global Matching (SGM) units. The Aggregator unit computes the $S(p)$ vector.

The data flow across these units is maintained through an efficient memory organization, wherein individual modules can read and write to the memory with minimum latency. Using double buffering and overlapping compute with communication, we ensure that our hardware maintains a high compute to communication ratio, thereby preventing any unnecessary stalls. Finally, we resort to a novel parallelism scheme within the SGM units, leading to increased throughput of the hardware. The above description of the hardware defines one Processing Unit (PU). If there are sufficient hardware resources on the FPGA, we can instantiate more than one PU, wherein each PU can compute the disparity of different pixel, thus allowing efficient scale-out. Overall, we achieve pipelined parallelism within a PU and pixel-level parallelism by using multiple such PUs.

B. CT Function Computation

The streamed input left and right images are stored in their respective line buffers. Since we are using 7x7 windows for computing the census transform, the line buffer sizes are set so that they can hold 7 image rows at a time. This is a parameter in our design and other window sizes are also possible. The CT module computes the $CT(p)$ vector for every pixel p in the right image in a row-major fashion. Using pipelined parallelism across pixels and disparity level parallelism within the computation of a pixel disparity, the hardware maintains a steady throughput of one $CT(p)$ computation per clock cycle.

The CT module reads the 7x7 windows from the line buffers of the left and right images, and computes their census transform. The census transform of a 7x7 window is a 48-bit vector (excluding the center pixel). The bit vectors from the right image are accumulated in a FIFO whose size is equal to the number of disparity levels ($d_{max} = 64$). The bit vectors from the left image are stored in a register of width 64×48 which is equal to 3072. Every time a 48 bit vector comes out from the left image, it will be right shifted into the corresponding register. Once the right FIFO is full, then that means all the census transform vectors required for the computation of the disparity of the first pixel in FIFO are available in the register (refer Figure 3). Then the Hamming Distance module deques the first element from the right FIFO and reads the register to compute all the 64 elements of the $CT(p)$ vector in parallel. Computing each element of the vector involves a 48-bit XOR operation followed by counting the number of 1s. This counting operation for hamming distance computation is performed by a tree structured circuit.

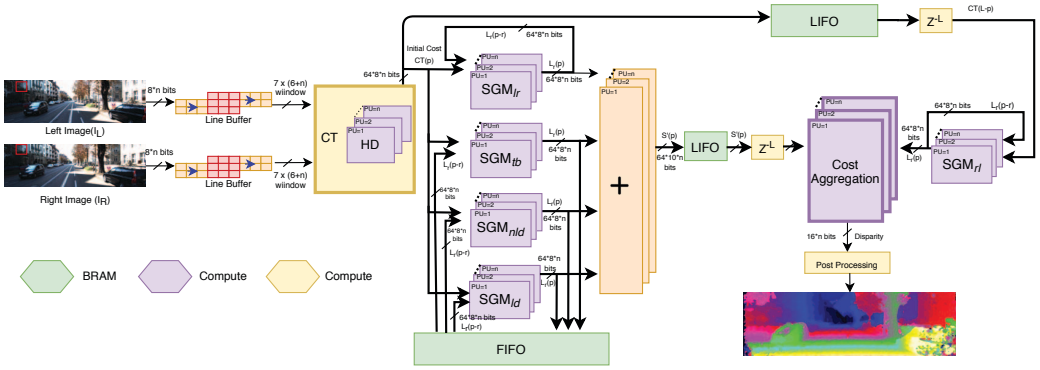


Fig. 2: Overall architecture of the proposed stereo vision accelerator with n PUs for an image with width L.

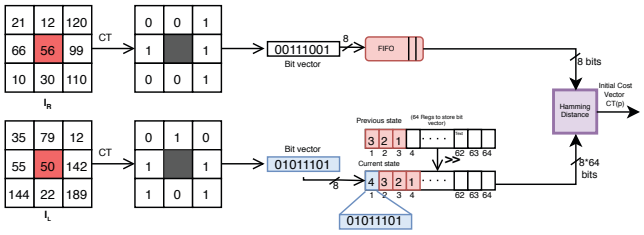


Fig. 3: Architecture of CT PU when $PU = 1$.

C. Architecture of SGM Units

Semi-Global Matching optimizes the path cost along each direction independent of other directions. We can infer this from Equation 3 wherein $L_r(p,d)$ is defined without reference to any other direction $r' \neq r$. Therefore we have five independent SGM units SGM_r for $r \in \{lr, ld, tb, nld, rl\}$, each computing the respective optimized path cost vector $L_r(p)$ (refer Figure 2). Recall that $L_r(p)$ has d_{max} dimensions each corresponding to a disparity level.

Each of the SGM units take the $CT(p)$ output vector from the CT module as input. The SGM units corresponding to the four directions lr, ld, tb and nld (refer 2) are computed in a streaming fashion as the scanline is read from the line buffer in left to right fashion. We call this as the forward pass. The output of SGM_{ld}, SGM_{tb} and SGM_{nld} units will be stored in the BRAM as they are required while computing the disparity of pixels in the next row. Further, these vectors are summed up to yield partial aggregate vector $S'(p)$ as follows.

$$S'(p) = \sum_{r \in \{lr,ld,tb,nld\}} L_r(p)$$

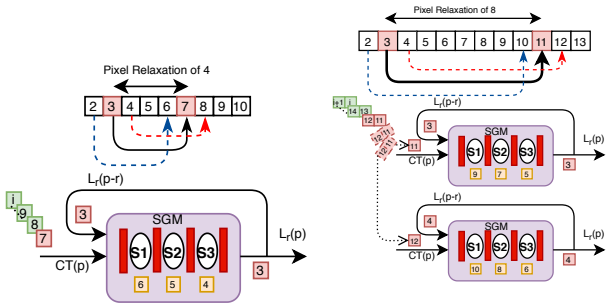
The partial aggregate vector $S'(p)$ is stored in the BRAM for later reference. The SGM_{rl} unit computes the optimized path in the *rl* direction in a backward pass after the current row is completely read. During the time backward pass is happening, simultaneously line buffer filling and forward pass of the next row also begins. Thus the forward pass of the next row and backward pass of the current row occurs in a perfect pipelined fashion with no pipeline stalls whatsoever. Finally, the Cost Aggregator module adds $L_{rl}(p)$ vector to the partial aggregate to generate the full aggregate vector $S(p)$.

$$S(p) = S'(p) + L_{rl}(p)$$

Note from the Figure 2 the partial aggregate vectors computed in the forward pass are stored in Last-In-First-Out (LIFO) data structure to match the vectors generated in the backward pass for complete aggregation. Further, the $CT(p)$ vectors generated by the CT module, for each row, are stored in a LIFO structure so that the SGM_{rl} unit can pop them in the appropriate order while computing $L_{rl}(p)$ vectors. The Cost Aggregator module computed the disparity by finding the index of the minimum value in $S(p)$ (refer Equation (1)) using a tree structure. Note that there is a data dependency from $L_r(p-r)$ to $L_r(p)$. The SGM_{ld}, SGM_{tb} and SGM_{nld} units use BRAM to store the $L_r(\cdot)$ vectors generated while processing the current row so that they can be consumed while processing the next row. This necessitates the usage of a buffer of size equal to row width wherein each element can hold a single $L_r(\cdot)$ vector completely. All the FIFO and LIFO structures in Figure 2 are realized using BRAM blocks on the FPGA.

D. Dependency Relaxation

The SGM_{lr} and SGM_{rl} units handle the dependency from $L_r(p-r)$ to $L_r(p)$ by forwarding their outputs as feedback to themselves through pipelined registers. That means computation of $L_r(p)$ vector for a pixel $p(x,y)$ requires the $L_r(p')$ vector of the pixel $p'(x-1,y)$. Each SGM unit (refer Figure 4a) is realized as a 3 stage pipelined structure and uses a tree structure to compute the minimum of $L_r(p-r,k)$ and calculates Equation 3. So by the time pixel $p(x,y)$ enters the SGM_{lr} pipeline, the $L_r(p')$ vector of its predecessor $p'(x-1,y)$ is not yet available. This will result in stalling the pipeline. We resolve this pipeline stall problem by relaxing the dependency constraint wherein $L_r(p)$ vector is computed by using the $L_r(p'')$ vector of the pixel $p''(x-4,y)$. For example, Figure 4a shows that the $L_r(\cdot)$ vector of pixel 7 is computed using the $L_r(\cdot)$ vector of pixel 3 as against that of pixel 6. In the experimental results section, we show that this dependency relaxation does not adversely affect the accuracy



(a) (b)

Fig. 4: Pixel-level parallelism for PU=1 and PU=2

of the disparity computation. Analogous discussion holds for rl direction.

E. Post Processing Unit

We apply a simple 9x9 median filter on the output disparity map image in a streaming fashion to get rid of the outliers.

F. Pixel-level Parallelism through Dependency Relaxation

The architecture described so far illustrates how pipelined parallelism is being used to compute the pixel disparities in a streaming fashion. In order to increase the throughput further, we can process multiple pixels in parallel. We cannot process two pixels $p(x, y)$ and $p'(x, y + 1)$ in parallel due to dependency in computing path costs in the ld, tb and nld directions. However, it is possible for us to process the pixels $p(x, y)$ and $p(x + 1, y)$ in parallel by relaxing the dependency constraints in the horizontal directions lr and rl, as in Section II-D.

In order to process two pixels $p(x, y)$ and $p'(x, y)$ in parallel, we instantiate two PUs, PU_0 and PU_1 along with a pixel distribution logic. Each PU has its own CT function computation unit, SGM units, and Cost Aggregation unit. Figure 2 depicts this by replicating the constituent units in a PU appropriately. The pixel distribution logic, supplies all the odd pixels in a row to PU_0 and even pixels to PU_1. This is illustrated in the Figure 4b. The SGM_{ld}, SGM_{tb} and SGM_{nld} units corresponding to vertical direction will have no problem in maintaining a steady-state pipeline throughput. However, in the horizontal directions lr and rl, the required $L_r(p - r)$ vector may not be readily available for a PU while computing a $L_r(p)$ vector. For example, in Figure 4b, a processing unit computing $L_r(\cdot)$ vector for pixel 11 does not have access to $L_r(\cdot)$ vector of even the pixel 7. However, the $L_r(\cdot)$ vector of pixel 3 can be used as an approximation. This we call as dependency constraint relaxation. In general, when using two PUs, while computing $L_r(p)$ vector of a pixel $p(x, y)$, we use the $L_r(p')$ vector of the pixel $p'(x - 8, y)$. If there are k PUs, then we use the pixel $p'(x-4k, y)$ to compute the $L_r(p)$ vector. Thus we establish a trade-off between throughput and accuracy in this work. We analyze this trade-off through experimental results in Section III.

III. Experimental Results

We synthesized our accelerator on a Virtex-7 690t FPGA connected to an Intel Core-i5 CPU running at 3.0 GHz. The FPGA is connected to the CPU host through a 3.5 Gbps PCIe-8x link. We compare the speed and accuracy of our accelerator running at 100 MHz with existing hardware implementations.

Table I compares the performance of the proposed architecture against other competitive approaches proposed in the literature. The metrics considered for throughput are Frames per Second (FPS) and Million Disparity Estimates per second (MDE/s). The metric considered for performance density is MDE/s per Kilo LUTs (MDE/s/KLUTs). These two metrics are chosen as they are commonly used across all the hardware implementations we refer to in this paper. As can be seen from Table I, our accelerator is able to process 1.6 times more frames per second compared to best performing [4] and [7]. Also, our accelerator achieves a comparable performance density of 118 MDE/s/KLUTs with respect to the others. This is because of the relatively less LUT consumption even at higher degrees of parallelism in our architecture. Next, we discuss the variation in the accuracy of depth maps with increase in the number of PUs. We evaluate our accelerator by processing the KITTI 2012 [14] and KITTI 2015 [15] image datasets.

We report two types of errors, average error and percentage bad pixels. A pixel is considered bad if the pixel differs from the corresponding ground truth's pixel disparity by more than 3. Table II shows error rates due to dependency relaxation that arises out of increase in the number of PUs, pixel relaxation of one represents standard SGM with five paths. Starting with base as $PU = 1$, with every increment in PU, the average drop in average non-occluded error and average occluded error is 0.125 and 0.116 respectively with both data-sets combined. Similarly average drop in percentage bad pixel for non-occluded and occluded error is 2.04% and 1.88% respectively for every increment in PU. Table III compares the percentage of bad pixels for occluded and non-occluded pixels with other hardware implementations. Our accelerator gives more accurate results compared to [7] but a drop of 4.6% is seen in accuracy in comparison with [11].

We next analyze the throughput gain with the increase in the number of processing units (PUs) for different frame sizes with 64 disparity levels, see Table IV. As can be seen, the FPS increase is proportional to the number of PUs in the hardware. For example, for a frame of size 1226x370, the FPS value increases from 209 to 834 as we scale from 1 to 4 PUs respectively.

Resource utilization is another important metric while evaluating hardware architectures. Table V, provides a comparison of resource consumption with respect to KLUTs and BRAMs with increasing PUs. As can be seen, when the hardware is scaled from 1 PU to 4 PUs, KLUTs and BRAM increase by 3.77 and 3.72 times respectively. This increment is linear with respect to the PU increase. Also the scaling-up results in an increase of FPS by 3.99 times.

TABLE I: Performance comparison of the proposed hardware accelerator with other state-of-the-art approaches.

Study	Algorithm	Image Size	Disparity Levels	Platform	FPS	MDE/s	MDE/s/KLUTs
Cambuim [5]	SGM	1024 x 768	128	Cyclone-4	127	12784	126.46
Li [4]	SGM	1280 x 960	64	Altera	197	15,493	161.2
Zhang [6]	Local	1920 x 1080	128	Kintex-7	60	15925	299
Schumacher [7]	SGM	1242 x 375	160	Virtex-5	199	14830	135.96
Ttofis [8]	Local	1280 x 720	64	Kintex-7	60	3538	26.8
Puglia [9]	Global	1024 x 768	64	Virtex-7	30	1510	5.19
Rahnama [10]	ELAS	1242 x 375	64	Zynq-7	47	1401	14.73
J. Wang [11]	SGM	1280 x 960	75	Zynq-7	31	2857	145.74
Wang [12]	SGM	1600 x 1200	128	Stratix-V	42.16	10472	47.2
Banz et al. [13]	SGM	640 x 480	128	Virtex-5	37	1455	21.2
Proposed (PU=4)	**SGM**	**1280 x 960**	**64**	**Virtex-7**	**322**	**25056**	**118.6**

TABLE II: Average error and percentage bad pixel of proposed architecture with increasing pixel relaxation on KITTI dataset.

Pixel Relaxation	KITTI 2012				KITTI 2015			
	Non-Occluded		Occluded		Non-Occluded		Occluded	
	Avg error	% bad pixel	Avg error	% bad pixel	Avg error	% bad pixel	Avg error	% bad pixel
1	4.43	14.6	5.28	16.1	2.76	11.6	3.17	12.8
4	4.27	15	5.06	16.5	2.73	12.5	3.09	13.4
8	4.38	16.3	5.14	17.9	2.9	14.18	3.24	15.2
12	4.49	17.6	5.2	19.1	3.05	15.7	3.39	16.7
16	4.59	18.7	5.35	20.2	3.16	20.8	3.5	21.02

TABLE III: Comparison of percentage bad pixels on KITTI dataset with other implementations.

Study	Platform	KITTI 2012		KITTI 2015	
		Non-Occ	Occ	Non-Occ	Occ
Wang [11]	Zynq-7	9.55%	10.94%	8.55%	8.91%
PU=1 (Proposed)	Virtex-7	15%	16.5%	10.5%	11.4%
Rahnama [10]	Zynq-7	16.3%	-	-	-
Schumacher [7]	Virtex-7	16.15%	18.06%	-	-

TABLE IV: FPS for different image sizes with increasing PUs.

PU	1226 x 3000	1226 x 960	1226 x 370
1	25.8	80.7	209.1
2	51.6	161.3	417.8
3	77.4	241.9	626.7
4	103.2	322.1	834.3

IV. Related Work

C. Banz et al. [13] introduced row-level parallelism using systolic array [16] based architecture running at 30 FPS for 640x480 pixel images with 128-disparity range. W. Wang et al. [12] processed 1024x768 pixel images with 96 disparity levels at 67 FPS on Altera Stratix-IV. They proposed a hybrid row-level parallelism scheme where they partitioned each row into segments and also reduced disparity level parallelism to achieve real-time processing speed. A drastic increase in memory consumption up to 6.3 MB was observed due to inconsistency in the data streaming for cross based cost aggregation with the SGM optimization. Y. Li et al. [4] used an architecture that runs at high frequency and achieves FPS of 197 and uses row-level parallelism proposed by Banz [13]. The latest work by J. Wang et al. [11] proposed a low resource architecture which processed 1280x960 pixel images with 75 disparity levels at 31 FPS. They used parallelism proposed by Banz [13] after downsampling the image and calculating disparity for only 50 levels to save on resources. This resulted in a drop of both accuracy and throughput as their architecture required four clock cycles to produce disparity of each pixel.

TABLE V: Comparison of resource utilization and performance for 1226x960 resolution image on Xilinx Virtex-7.

PU	MDE/s	KLUTs	MDE/KLUT	BRAM
1	6279.53	56.008 (13%)	112.12	172 (12%)
2	12548.85	99.06 (23%)	126.68	328.5 (22%)
3	18822.55	163.388 (38%)	115.20	486 (33%)
4	25056.96	211.26 (49%)	118.61	641.5 (43%)

V. Conclusions

In this paper, we proposed a parallel pipelined hardware accelerator for computing disparity map. Our stereo vision algorithm is based on Census Transform and Semi-Global Matching. An important contribution of our work is a novel pixel-level data parallelism which establishes a trade-off between accuracy and throughput. Overall, our hardware architecture processes 1.6 times more FPS compared to the existing state of the art SGM implementations on FPGAs. The stereo vision accelerator achieved 322 FPS for 1280x960 resolution images with 64 disparity levels on a Virtex-7 FPGA. Our architecture is scalable and can be tailored according to the requirement of FPS, accuracy and availability of resources.

REFERENCES

[1] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1, pp. 7–42, 2002. [Online]. Available: https://doi.org/10.1023/A:1014573219977

[2] H. Hirschmuller, "Stereo processing by semi-global matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, Feb 2008.

[3] J. Banks, M. Bennamoun, and P. Corke, "Non-parametric techniques for fast and robust stereo matching," in *TENCON'97 Brisbane-Australia. Proceedings of IEEE TENCON'97. IEEE Region 10 Annual Conference. Speech and Image Technologies for Computing and Telecommunications (Cat. No. 97CH36162)*, vol. 1. IEEE, 1997, pp. 365–368.

[4] Y. Li, C. Yang, W. Zhong, Z. Li, and S. Chen, "High throughput hardware architecture for accurate semi-global matching," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2017, pp. 641–646.

[5] L. F. S. Cambuim, J. P. F. Barbosa, and E. N. S. Barros, "Hardware module for low-resource and real-time stereo vision engine using semi-global matching approach," in *2017 30th Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2017, pp. 53–58.

[6] X. Zhang, H. Sun, S. Chen, L. Song, and N. Zheng, "Nipm-swmf: Toward efficient fpga design for high-definition large-disparity stereo matching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 5, pp. 1530–1543, May 2019.

[7] F. Schumacher and T. Greiner, "Matching cost computation algorithm and high speed fpga architecture for high quality real-time semi global matching stereo vision for road scenes," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct 2014, pp. 3064–3069.

[8] C. Ttofis and T. Theocharides, "High-quality real-time hardware stereo matching based on guided image filtering," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2014, pp. 1–6.

[9] L. Puglia, M. Vigliar, and G. Raiconi, "Real-time low-power fpga architecture for stereo vision," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 11, pp. 1307–1311, 2017.

[10] O. Rahnama, D. Frost, O. Miksik, and P. H. S. Torr, "Real-time dense stereo matching with elas on fpga-accelerated embedded devices," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2008–2015, July 2018.

[11] J. Wang, Z. Li, L. Yao, S. Chen, and F. Wu, "Low-resource hardware architecture for semi-global stereo matching," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2019, pp. 1–4.

[12] W. Wang, J. Yan, N. Xu, Y. Wang, and F. Hsu, "Real-time high-quality stereo vision system in fpga," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 10, pp. 1696–1708, Oct 2015.

[13] C. Banz, S. Hesselbarth, H. Flatt, H. Blume, and P. Pirsch, "Real-time stereo vision system using semi-global matching disparity estimation: Architecture and fpga-implementation," in *2010 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*, July 2010, pp. 93–101.

[14] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[15] M. Menze, C. Heipke, and A. Geiger, "Joint 3d estimation of vehicles and scene flow," in *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015.

[16] R. P. Hughey, "Programmable systolic arrays," USA, Tech. Rep., 1991.