REAL-TIME HYPERSPECTRAL STEREO PROCESSING FOR THE GENERATION OF 3D DEPTH INFORMATION

Nina Heide, Christian Frese, Thomas Emter, and Janko Petereit

Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB,
Fraunhoferstr. 1, 76131 Karlsruhe, Germany

ABSTRACT

We present a local stereo matching method for hyperspectral camera data, allowing multiple usage of camera hardware and imaging data such as for object classification or spectral analysis and multichannel input to the correspondence problem. The matching process combines correlation-based similarity measures for pixel windows utilizing all 16 spectral channels followed by a consistency check for disparity selection. We evaluate stereo-processing methods focusing on effectiveness and runtime of the processing on a CPU and analyze parallelization possibilities. Based on the results of the evaluation on the CPU, we implement the optimized stereo matching for images with 16 channels on a graphics processing unit (GPU) utilizing the Compute Unified Device Architecture (CUDA). The parallel processing of the calculation steps to obtain the disparity image on the GPU achieves more than 27× speed up, resulting in calculation and post-processing of hyperspectral images with 8 – 13 Hz, depending on the selection of maximum disparity. The 3D reconstruction achieves a mean square error of 0.0267 m^2 in distance measurements from 5 – 10 m.

***Index Terms*—** hyperspectral imaging, stereo vision, real-time stereo matching, GPU processing

1. INTRODUCTION

The reconstruction of the environment is an essential requirement, for instance in the perception for autonomous robotic systems. To capture the environment for localization, mapping and execution of autonomous tasks, active or passive sensors can be used. Besides light and radio detection and ranging (LiDAR, RADAR), the structure of the surrounding area can also be captured by stereo vision. Passive stereo vision allows the 3D reconstruction of the environment with ambient lighting. Hyperspectral cameras enable the usage of the same hardware and data for multiple purposes as 3D perception, object classification and spectral analysis. For 3D perception, the analysis of light intensity in different spectral channels supports correlation-based similarity measures. Correlation-based, local stereo matching produces a dense 3D reconstruction and facilitates parallel processing of image data on GPUs, which significantly speeds up the calculation of disparity images.

Our aim is to passively perceive the environment in 3D using hyperspectral cameras for the purpose of autonomous driving in outdoor environments [1]. We extend the local stereo matching proposed by Jiao et al. [2] to hyperspectral images, considering a significantly larger amount of image information for depth reconstruction than in RGB images. Optimization is performed on a CPU for high quality disparity images and low runtime comparing hyperspectral data to 3D LiDAR data captured on technology demonstrators in unstructured outdoor environments. To achieve real-time 3D reconstruction we implement our optimized algorithm on a GPU performing adjustments for parallel processing.

2. RELATED WORK

Stereo matching techniques divide into feature-based techniques producing sparse point clouds [3, 4], and correlation-based techniques generating considerably denser point clouds [5].

The correlation-based, local stereo matching method for RGB images proposed by Jiao et al. [2] consists of calculating an initial disparity using the combination and aggregation of cost functions followed by several post-processing steps. The cost values from Modified Colour Census Transform (CCT), sum of absolute differences (SAD) and gradient differences (SGD_x, SGD_y) are combined with regard to their assigned weight and aggregated using the Guided Filter Algorithm (GF) [6] choosing the disparity with the lowest costs. In post-processing, disparities are recalculated with interchanged input pictures, accepting only disparities passing this left-right consistency check (LRC). Inconsistent values are corrected by cross-region based voting (CBIV) [7] and detection of remaining artifacts and refinement.

Semi-Global Matching (SGM) of Hirschmüller [8] uses pixel-by-pixel matching of mutual information combined with an approximately calculated global smoothness constraint, requiring about 1 s on typical images [8]. Global methods like graph cuts [9, 10] produce high quality disparity maps along with high processing time.

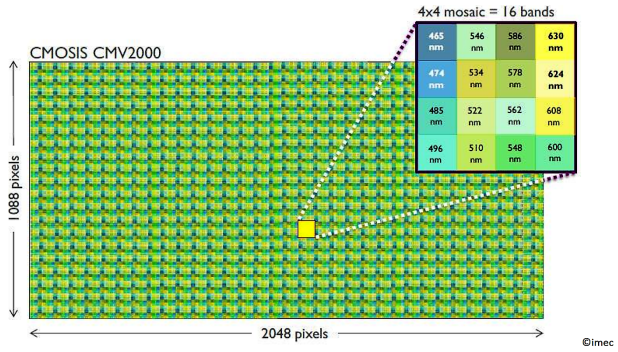

Fig. 1. Scheme of mosaic structure to combine 16 channels into one pixel (figure courtesy of Ximea).

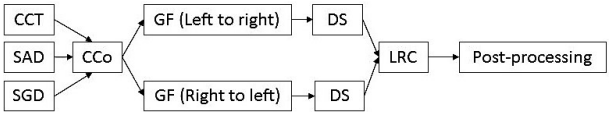

Fig. 2. Hyperspectral local stereo matching algorithm.

Hyperspectral stereo imaging is mainly used in the field of photometric stereo for the estimation of surface normals [11, 12], in remote sensing [13, 14] as well as for food analysis and control [15, 16]. In the field of robotics, Trierscheid et al. propose a method for the detection of victims with rescue robots using the characteristic spectra of human skin [17].

Mahieu and Lowney describe a CUDA implementation of SGM [18], evaluating their results with RGB datasets from the Middlebury Stereo Evaluation (MiEv) [19]. The runtime of the proposed CUDA implementation lies around 2.5 s for images with approximately 140,000 pixels on a NVIDIA GeForce 940M. Kowalczuk et al. implement stereo matching on CUDA using adaptive support-weight cost aggregation followed by an iterative refinement method [20]. One disparity image with 60 levels requires 120 ms on a NVIDIA GeForce GTX 580 for MiEv datasets with 640×480 pixels. Gallup et al. [21] achieve the calculation of disparity images with 40 Hz on 640×480 images with 50 disparity levels using SAD over a 7×7 matching window. Mei et al. propose cost matching utilizing SAD and CCT on CUDA with approximately 10 Hz, ranked top performer in the MiEv at the time of publication [7].

3. METHODOLOGY

3.1. Hyperspectral local stereo matching

We use Ximea xiSpec MQ022HG-IM-SM4X4-VIS cameras with 16 spectral bands from 465 - 630 nm, divided into 512×272 mosaic pixels as shown in Fig. 1. The 4×4 mosaic structure of the mosaic pixels is treated as one pixel containing 16 channels. Bilinear interpolation of the respective

Fig. 3. Evaluation images on the CPU: Vegetation with test objects (upper-left,(a)), mixed area (upper-right,(b)), pure urban zone (bottom,(c)).

intensities from neighboring mosaic pixels is applied and the image is processed as 512×272 image with 16 channels instead of three when using only RGB intensities. The hyperspectral 3D reconstruction rests on the three channel matching method proposed by Jiao et al. [2], being ranked 5/153 in the MiEv. Instead of using only RGB values, we evaluate the intensity values for each spectral band in the calculation of the disparity value. For the search of corresponding pixels, we compare the intensity values of equivalent bands matching mosaic pixels, for instance in the calculation of SAD. $I_i^L(p)$ denotes the intensity of channel i in pixel p of the left image, compared to the intensity $I_i^R(p-d)$ of pixel p and disparity d in the right image:

$$\text{SAD} = \frac{\sum_{i=1}^{16} ||I_i^L(p) - I_i^R(p-d)||}{16}.$$

Fig. 2 illustrates the process of calculating the disparity values from single cost functions to post-processing. CCo refers to the weighted sum of all cost values and DS to the selection of the disparity value with minimum cost. The disparity image can additionally be smoothed by a Median Filter (MF), posing a trade-off between high accuracy and smooth transitions of the disparity values with less invalid pixels. Each processing step is evaluated on images from unstructured outdoor environments as shown in Fig. 3, focusing on effective processing and accurate depth maps. The local method utilizing the combination of mosaic pixels into window structures favors parallel calculation of the single steps needed to obtain disparity images on a GPU.

3.2. Optimization for real-time operation on GPUs

We use a NVIDIA Quadro M6000 with 12 GB DDR SDRAM, 317 GB/s throughput and 3072 CUDA kernels. Calculation optimizations as the replacement of the L^2-norm used in [2] with the L^1-norm exchanging square root calculation with division or the removal of the exponential function from the calculation of the CCT minimize processing time. We compare the cost values calculated on the CPU and optimized on

Fig. 4. Left: IOSB.amp.Q1, right: TULF.

the GPU on a reference image for each processing step at a fixed disparity value. Considering the high number of spectral channels and pairwise comparisons for correspondence formation, we focus on optimizing memory access.

Loading optimization is achieved by minimizing global memory access with repeated consolidation of overlapping windows in the local memory [22], formation of single instruction, multiple thread (SIMT) groups to share local memory allocations, and additional provisioning of overlapping SIMT group elements in local memory, using a tile structure including the margin regions of neighboring pixels of the SIMT window. The CPU implementation uses the OpenCV matrix structure row-column-channel. To prevent stride inefficiency, we import the channel information blockwise in the local memory of a SIMT structure, changing the matrix structure to channel-row-column on GPUs. We adjusted the dimensions of SIMT groups using the NVIDIA Visual Profiler to analyze registry allocation and memory requirements. A native CUDA context on the M6000 consists of 32 elements, hence we evaluate integer multiples of 32 and integer factors of the 512×272 resolution. With 16 channels, each pixel needs 16 bytes forming the basis for memory allocation. CUDA allows allocations up to 48 KiB. For effective latency hiding, a partitioning allowing eight contexts is necessary. As the M6000 has 96 KiB shared memory per streaming multiprocessor, we require less than 12 KiB per context.

4. EVALUATION

4.1. Technology demonstrators

The hyperspectral camera system is mounted on two autonomous platforms for driving in unstructured environment, shown in Fig. 4. The IOSB.amp.Q1 platform is equipped with one Velodyne HDL-64E 3D LiDAR sensor, the TULF [23] with two Velodyne HDL-32E. The baseline of the stereo systems amounts to 0.74 m on the IOSB.amp.Q1 and 1.18 m on the TULF. The platforms offer calibrated 3D LiDAR measurements with only translational shift to the vehicle frame. The LiDAR data is transformed into the vehicle frame as

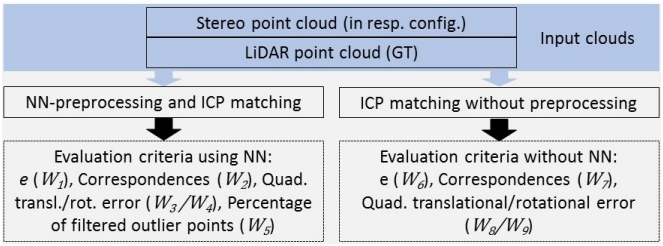

Fig. 5. Evaluation using NN-preprocessing and ICP matching, GT denotes ground truth and W_i the weighting for the evaluation metric.

Table 1. Parameter variation of all configurations, parameter variation of four most accurate results and parameters chosen for unstructured environments due to highest accuracy.

Parameter	Variation	Top 4	Chosen value
Weighting CCT	0-1	0.70	0.70
Weighting SAD	0-1	0/0.15	0.15
Weighting SGD_x	0-1	0.10	0.10
Weighting SGD_y	0-1	0/0.05	0.05
Perform GF	false/true	true	true
GF regularization	0.001/0.004	0.004	0.004
GF window size	2-25	7/10	7
Perform LRC	false/true	true	true
Perform CBIV	false/true	false	false
Perform MF	false/true	false/true	false
MF window size	3-7	5	–

ground truth for stereo matching as illustrated in Fig. 5.

The iterative closest point (ICP) method allows accurate registration of point clouds [24, 25]. Generalized-ICP [26] extends ICP to minimize errors between locally planar surfaces. We calibrate the stereo matching results to the LiDAR data using ICP-matching instead of GICP-matching as the point clouds generated from correlation-based stereo matching favor the minimization of the error between single corresponding points due to plate-shaped structures in larger distance from the cameras.

4.2. Hyperspectral local stereo matching

The results from hyperspectral stereo are compared to 3D LiDAR data. The datasets from the MiEv [19] contain only RGB data and only from structured indoor environments, making them unsuitable to evaluate performance in unstructured outdoor environments using 16 spectral bands. Hence we develop our own evaluation procedure. It is divided into matching the stereo point cloud to LiDAR data with ICP, optionally preprocessed by filtering outliers with Nearest Neighbor Search (NN) and secondly evaluating distance measurements to test objects. The evaluation metric for ICP and

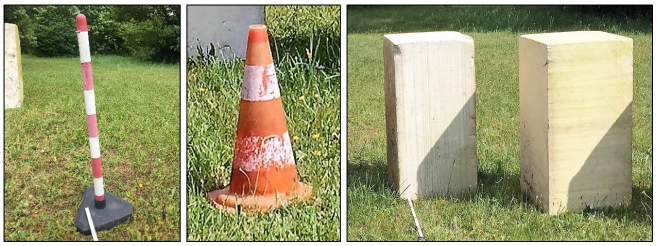

Fig. 6. Test objects.

Table 2. Timing results on the CPU (t_C) and GPU (t_G), max. disparity values 80 and 174, $n = \frac{t_C}{t_G}$.

Step	$t_{C,80}$ in ms	$t_{C,174}$ in ms	$t_{G,80}$ in ms	$t_{G,174}$ in ms	n_{80}	n_{174}
Init	3.68	2.88	2.05	1.84	1.80	1.57
CCT	502.79	486.29	4.76	5.05	105.56	96.22
SAD	228.34	404.16	1.04	1.34	218.76	301.95
SGD_x	155.60	272.86	2.80	3.31	55.62	82.55
SGD_y	144.50	272.88	3.39	3.74	42.64	72.85
CCo	43.66	94.65	4.66	8.06	9.36	11.74
GF	434.32	913.87	25.56	47.80	16.99	19.12
DS	25.89	49.00	1.37	1.39	20.28	35.15
GF	432.46	902.03	25.47	47.73	16.98	18.90
DS	26.55	48.46	1.37	1.43	19.33	33.95
LRC	0.74	0.67	0.85	0.74	–	–
Total	1998.53	3447.75	73.32	122.43	27.26	28.16

NN considers the criteria according to Fig. 5, among them the Euclidean fitness score e, calculated from the sum of squared errors between corresponding source $(x/y/z)_S$ and target points $(x/y/z)_T$, divided by the number of correspondences N:

$$e = \frac{\sum_{i=1}^{N}\left((x_{i,S}-x_{i,T})^2 + (y_{i,S}-y_{i,T})^2 + (z_{i,S}-z_{i,T})^2\right)}{N}.$$

We evaluate different parameter configurations with variations as given in Table 1 by ranking each configuration for each criterion i given in Fig. 5. All configurations are evaluated on the images presented in Fig. 3, which contain bushes, trees and buildings as well as lawn and street-like ground. We assign double importance to the results for images with vegetation ((a),(b) in Fig. 3), as we focus on unstructured outdoor environments. Awarding penalty points $p_{i,k}$ for the result of each criterion relative to other configurations, we combine the penalties applying the weighting factors W_i explained in Fig. 5 for each configuration k to the penalty p_k^j for configuration k on image j:

$$p_k^j = \sum_{i=1}^{9}\left(p_{i,k}^j W_i\right).$$

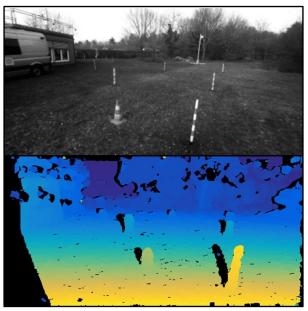

Fig. 7. GPU: evaluation image (top), disparity image (bottom).

Proc. step	Group dim.
CCT	16×8
SAD	128×1
SGD_x	16×16
SGD_y	16×16
CCo	8×64 - 512×1
DS	16×16
GF	256×1

Table 3. Group dimensions for M6000.

To combine the results for all evaluation images, the penalties are summarized:

$$p_k = 2p_k^{(a)} + 2p_k^{(b)} + p_k^{(c)}.$$

In the second evaluation step we weight the LiDAR-measured distance once and the hand-measured distance twice. Fig. 6 shows the test objects, positioned in 4.62 – 9.97 m from the origin of the vehicle frame. The mean squared error of the distance between the respective test object and the origin of the vehicle frame results to 0.0267 m^2 for the chosen value configuration in Table 1.

4.3. Optimization for real-time operation on GPUs

The runtime with the configuration given in Table 1 is measured on an Intel Xeon E5-2640 v3 system with eight cores, which can execute 16 threads in parallel. Before starting to execute the steps as illustrated in Fig. 2, the raw image has to be preprocessed (52 ms) and rectified (26 ms). The costs for all disparity values are equal between processing on the CPU and on the GPU. Fig. 7 shows the reference image for the comparison of the results from the CPU and the GPU as well as the final disparity image generated on the GPU. Table 2 describes the timing results. The group dimensions used on the M6000 are shown in Table 3.

5. CONCLUSION

Our method allows the calculation of 8 to 13 disparity images per second including 3D reconstruction on the M6000 GPU, achieving a mean square error of 0.0267 m^2 in measuring distances up to 10 m. GPU implementation significantly speeds up the calculation and 3D depth information for unstructured outdoor environments can be generated in real-time, allowing proper close-range mapping of the environment.

6. ACKNOWLEDGEMENTS

The financial support from BAAINBw U6.2 and the WTD 41 (Bundeswehr) is gratefully acknowledged.

7. REFERENCES

[1] Thomas Emter, Christian Frese, Angelika Zube, and Janko Petereit, "Algorithm toolbox for autonomous mobile robotic systems," *ATZoffhighway worldwide*, vol. 10, no. 3, pp. 48–53, 2017.

[2] Jianbao Jiao, Ronggang Wang, Wenmin Wang, Shengfu Dong, Zhenyu Wang, and Wen Gao, "Local stereo matching with improved matching cost and disparity refinement," in *IEEE MultiMedia*, 2014, number 4.

[3] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, "SURF: Speeded up robust features," *Computer vision–ECCV 2006*, pp. 404–417, 2006.

[4] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski, "ORB: An efficient alternative to SIFT or SURF," in *IEEE international conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 2564–2571.

[5] Nick Pears, Yonghuai Liu, and Peter Bunting, *3D imaging, analysis and applications*, vol. 3, Springer, 2012.

[6] Kaiming He, Jian Sun, and Xiaoou Tang, "Guided image filtering," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[7] Xing Mei, Xun Sun, Mingcai Zhou, Shaohui Jiao, Haitao Wang, and Xiaopeng Zhang, "On building an accurate stereo matching system on graphics hardware," in *IEEE International Conference on Computer Vison Workshops (ICCV Workshops)*, 2011.

[8] Heiko Hirschmüller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[9] Yuri Boykov, Olga Veksler, and Ramin Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[10] Vladimir Kolmogorov and Ramin Zabih, "Computing visual correspondence with occlusions using graph cuts," in *IEEE International Conference on Computer Vision*, 2001, vol. 2, pp. 508–515.

[11] Keisuke Ozawa, Imari Sato, and Masahiro Yamaguchi, "Hyperspectral photometric stereo for a single capture," *J. Opt. Soc. Am. A*, vol. 34, no. 3, Mar 2017.

[12] Giljoo Nam and Min H Kim, "Multispectral photometric stereo for acquiring high-fidelity surface normals," *IEEE computer graphics and applications*, vol. 34, no. 6, pp. 57–68, 2014.

[13] Nahum Gat and CA Torrance, "Real-time multi-and hyper-spectral imaging for remote sensing and machine vision: an overview," in *Proc. ASAE Annual International Mtg*, 1998.

[14] Thomas Lillesand, Ralph W Kiefer, and Jonathan Chipman, *Remote sensing and image interpretation*, John Wiley & Sons, 2014.

[15] Da-Wen Sun, *Hyperspectral imaging for food quality analysis and control*, Elsevier, 2010.

[16] Gamal ElMasry, Ning Wang, Adel ElSayed, and Michael Ngadi, "Hyperspectral imaging for nondestructive determination of some quality attributes for strawberry," *Journal of Food Engineering*, vol. 81, 2007.

[17] M. Trierscheid, J. Pellenz, D. Paulus, and D. Balthasar, "Hyperspectral imaging or victim detection with rescue robots," in *IEEE International Workshop on Safety, Security and Rescue Robotics*, Oct 2008.

[18] Robert Mahieu and Michael Lowney, "Real-time semi-global matching using CUDA implementation," 2016.

[19] Daniel Scharstein and Richard Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[20] Jedrzej Kowalczuk, Eric T Psota, and Lance C Perez, "Real-time stereo matching on CUDA using an iterative refinement method for adaptive support-weight correspondences," *IEEE transactions on circuits and systems for video technology*, vol. 23, 2013.

[21] David Gallup, Jan-Michael Frahm, and Joe Stam, "Real-time local stereo using CUDA," in *NVIDIA Research Summit*, 2009.

[22] John Nickolls and William J Dally, "The GPU computing era," *IEEE micro*, vol. 30, no. 2, 2010.

[23] "ELROB, Robotics for military applications," 2016.

[24] Paul J. Besl and Neil D. McKay, "A method for registration of 3-D shapes," in *IEEE Transactions on pattern analysis and machine intelligence*, 1992.

[25] Zhengyou Zhang, "Iterative point matching for registration of free-form curves and surfaces," in *International Journal of Computer Vision*, 1994, number 13.

[26] A. V. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proceedings of Robotics: Science and Systems*, 2009.