MTStereo 2.0: Accurate Stereo Depth Estimation via Max-Tree Matching

Rafaël Brandt[1(✉)], Nicola Strisciuglio[2], and Nicolai Petkov[1]

[1] Bernoulli Institute, University of Groningen, Groningen, The Netherlands
r.brandt@rug.nl

[2] Faculty of Electrical Engineering, Mathematics and Computer Science at University of Twente, Enschede, The Netherlands

Abstract. Efficient yet accurate extraction of depth from stereo image pairs is required by systems with low power resources, such as robotics and embedded systems. State-of-the-art stereo matching methods based on convolutional neural networks require intensive computations on GPUs and are difficult to deploy on embedded systems. In this paper, we propose MTStereo2.0, an improved version of the MTStereo stereo matching method, which includes a more robust context-driven cost function, better detection of incorrect matches and the computation of disparity at pixel level. MTStereo provides accurate sparse and semi-dense depth estimation and does not require intensive GPU computations. We tested it on several benchmark data sets, namely KITTI 2015, Driving, FlyingThings3D, Middlebury 2014, Monkaa and the TrimBot2020 garden data sets, and achieved competitive accuracy. The code is available at https://github.com/rbrandt1/MaxTreeS.

Keywords: Stereo matching · Max-Tree

1 Introduction

Estimation of scene depth from stereo image pairs is deployed as a building block in various high level computer vision applications [8] in which the three-dimensional structure of a scene is recovered by matching corresponding pixels.

The similarity of two pixels is quantitatively computed by a matching cost function, e.g. absolute image gradient or gray-level difference [27]. Substantial matching ambiguity can be caused by repetitive patterns and uniformly colored regions. Hence, costs are aggregated over neighbor pixels to strengthen the robustness of the matching. For instance, color similarity and proximity were used for weighted cost aggregation in [37] and [39], while the strength of image boundaries in [3]. Early methods performed exhaustive matching search [27], which requires many

This work was partially funded by the EU H2020 program, under the project TrimBot2020 (grant No. 688007).

N. Tsapatsoulis et al. (Eds.): CAIP 2021, LNCS 13052, pp. 110–119, 2021.
https://doi.org/10.1007/978-3-030-89128-2_11

computations. Later approaches reduced the disparity search-range by computing a coarse disparity map first and then refining it iteratively [10]. Image pyramids were also used to reduce disparity search range in [17,29]. A coarse disparity map is estimated considering the full disparity range. Then, increasingly higher-resolution disparity maps are constructed whereby the disparity search range is dictated by the coarser map. To increase efficiency and reduce ambiguity, matching (hierarchically structured) image regions instead of individual pixels was proposed [1,6,30].

Recent approaches use convolutional neural networks (CNNs) to compute aggregated matching costs. One of the first CNN-based methods deployed a siamese network [38], which works with small patch inputs. Increasing the receptive field while maintaining details in estimated disparity maps was investigated [4]. Pairs of siamese networks, each receiving as input a pair of patches at different scales were used and the matching cost was computed as the inner product between their responses. Although CNN-based methods are very accurate, they need power-consuming dedicated hardware (GPUs) to compute the many convolutions. This limits their usability on embedded or power-constrained systems. This applies, for instance, to battery-powered robots or drones, for which depth estimation has to trade-off between accuracy and computational efficiency. Furthermore, for robot navigation and obstacle avoidance, the very high accuracy and density of estimation achieved by CNNs are not strictly necessary.

We present a stereo matching method, named MTStereo 2.0, that balances efficiency with effectiveness, making it appropriate for devices where limited computational and energy resources are available. The MTStereo 1.0 algorithm, that we proposed in [2], exploits contrast information of objects in a hierarchical fashion to efficiently and effectively perform stereo matching. It constructs a hierarchical representation of image scan-lines using Max-Trees [24], and performs disparity estimation via tree matching cost computation that takes into account contextual image structural information. The MTStereo 2.0 algorithm that we propose improves on the 1.0 version as it *a)* deploys a more robust cost function, *b)* performs more thorough incorrect match detection, *c)* computes disparity maps with pixel-level rather than node-level precision. We carried out an extensive experimental benchmark on several data sets, namely the KITTI, Driving, FlyingThings3D, Middlebury, Monkaa and TrimBot2020 data sets.

2 MTStereo 2.0

The matching procedure performed by our method is shown in Fig. 1. Matching the finest structures directly results in non-efficient yet precise stereo matching, while matching coarse structures results in disparity maps that lack precision but can be efficiently obtained. To tackle this trade-off and efficiently perform precise stereo matching, our method matches increasingly finer regions in an iterative manner, and only compares regions contained in earlier matched coarser ones.

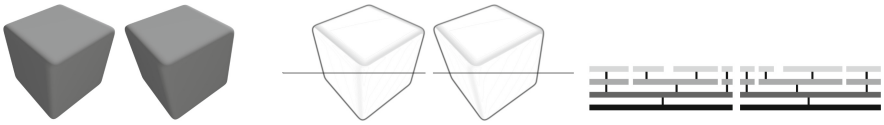

Fig. 1. A (left column) pair of stereo images and their (middle column) version resulted by edge-detection (darker regions contain more contrast). The Max-Tree representations of the horizontal scan-lines highlighted in the middle column are illustrated in the right column: MTstereo 2.0 computes the disparity for coarse structure first (nodes represented by darker bars) and increasingly matches only finer structures (with lighter color) contained in earlier matched coarser ones.

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

Fig. 2. Outputs after intermedate algorithm stages: (a) pre-processed image of Middlebury training data set, (b) coarse-to-fine matching (c) outlier removal, (d) reliable node extrapolation, (e) guided pixel matching, (f) outlier removal.

2.1 Steps Performed by MTStereo 2.0

MTStereo 2.0 is composed of the following steps: pre-processing, Max-Tree construction, cost volume computation, cost aggregation, consistency check, disparity map computation, confidence check and map refinement.

Pre-processing. We process the input rectified image pair images with a median filter for noise removal. Subsequently, we detect edges by a horizontal and a vertical Sobel operator, and average their absolute response images pixelwise. We perform contrast-stretch and color quantization on the inverted image, see an example in Fig. 2a. We compute a 1D Max-Tree for each scan-line of the pre-processed images. A parameter q controls the number of colors for color quantization and influences the size of the constructed trees. Shallower trees are less expensive to match, but represent less precisely the image structures.

Max-Tree Construction. The coarse-to-fine matching is facilitated by a hierarchical representation of both images in a rectified stereo pair: we compute 1D Max-Trees on the scan-lines of the images using the algorithm in [33]. We store regions with less contrast being contained in regions with more contrast [30]. The Max-Tree, proposed by [24], allows storing the hierarchy of connected components resulting from different thresholds and is efficiently constructed. A 1D connected component set contains the 1-valued pixels for which no 0-valued pixel exists in between any of the pixels in the binary image resulting from applying a threshold t to a 1D gray-scale image (i.e. a scan-line).

Cost Volume. We construct a cost volume by computing the cost of matching each pixel in the left image with those in the right image at all possible disparity levels. We compute the weighted-average of the absolute difference of the pixel gray-level, horizontal Sobel, and vertical Sobel values. We process each slide of the cost volume with a Gaussian blur operator, which smooths the estimation. A parameter ω_{cv} controls the size of the Gaussian blur kernel ($\omega_{cv} \times \omega_{cv}$). MTStereo 1.0 does not make use of a smoothed cost volume and does not consider gray-level difference in the cost computation.

Matching Cost Aggregation. We then use the cost volume and tree structures for matching of regions in image pairs: nodes that correspond to coarse image structures are matched first, followed by finer structures. Only the finest nodes and their descendants are matched, the width of which is greater than ω_α and less than ω_β (see [2] for details).

We define the matching cost as a context cost and an intensity cost. The context cost is the average relative difference between the area of corresponding ancestors of a node pair. Given a node pair (n_1, n_2), we define that its ancestors (n_3, n_4), where n_3 is an ancestor of n_1 and n_4 is an ancestor of n_2, can be matched if in the Max-Trees the number of nodes between n_3 and n_1, and n_4 and n_2 is equal. We define the intensity cost as the average of the cost volume matching costs at aligned pixels part of the two nodes. Given a node pair, we compute the matching cost at the location of the left node's left (right) endpoint and disparity between the left (right) endpoints of both nodes, as well as at linearly interpolated disparity and pixel values in between the two endpoints. This definition of intensity cost is more robust than that of MTStereo 1.0. Parameter α controls the relative weight of the intensity cost (α) and context cost ($1 - \alpha$).

Let $P_y = (n_{L,y}, n_{R,y})$ denote a pair of nodes, respectively in the left and right image, both at row y. Matching cost is aggregated over their node neighborhood. We define the neighborhood of P_y as the nodes with the same coarseness level as P_y that have similar x-coordinates and are in scan-lines next to each other in the original image. Two nodes have the same coarseness level when the distance (i.e. the difference in tree level) between the leaf nodes with the greatest level out of the leaf nodes which are descendants of the nodes, and the nodes themselves are equal. Recursively, node pair P_{y+1} is part of the neighborhood of P_y, if both $n_{L,y+1}$ crosses the x-coordinate of the center of node $n_{L,y}$, $n_{R,y+1}$ crosses the x-coordinate of the center of node $n_{R,y}$, and both $n_{L,y+1}$, and $n_{R,y+1}$ have a y-coordinate which is one lower or higher than that of ($n_{L,y}$ and $n_{R,y}$). At most, the ω_γ nodes above and below a node pair are included in the neighborhood of P_y. In the coarse-to-fine matching procedure, we compute a disparity search range for each node in each iteration. Given a node pair that has likely been correctly matched in a previous iteration, only descendants of this node pair are matched in subsequent iterations. Nodes are considered likely correctly matched when they pass a left-right consistency check [9] and a confidence check (peak-ratio used in [36]). The confidence check, not used in MTStereo 1.0, provides more accurate disparity maps as it filters out ambiguous/incorrect matches.

Disparity Map. The coarse to fine matching procedure results in a list of matched node pairs. For each of the finest nodes, we compute the disparity at the left and right endpoints of matched nodes by linear interpolation, see Fig. 2b for an example disparity map. We detect and remove outliers, when in the $(2 \cdot 21) \times (2 \cdot 21)$ local neighborhood the number of pixels with a disparity difference that exceeds their x-offset surpasses the number of pixels with a disparity difference less than or equal to their x-offset (see Fig. 2c for a post-processed map).

We densify sparse disparity maps by computing the median disparity values of neighbor nodes across scan-lines. This procedure fills-in some missing disparity values and smooths out eventual remaining outliers. Furthermore, as in the tree disparity values are computed only for the left- and rightmost nodes, our densification process also includes the linear interpolation of the disparity values of the pixels within the node, thus obtain the disparity of an entire 1D region. Inside-node interpolation is only performed by the semi-dense variant. A disparity map after reliable node extrapolation is illustrated in Fig. 2d. Different from MTStereo 1.0, we perform pixel-level matching to recover surface shape. Disparity search range for pixels is set such that only disparities which differ at most a fraction ω_ω from the previously computed disparity value are considered. The matching cost of pixel pairs is derived from the constructed cost volume.

Confidence Check and Map Refinement. We perform a confidence check on the estimated pixel disparities, which evaluates the relative difference between the matching cost of the best match and that of the second-best match. If this difference is more than ω_Π the check is passed. Areas which contain more texture are more likely to pass the confidence check. We obtain the final disparity map by removing outliers. A disparity map after noise removal is shown in Fig. 2f.

3 Experiments

We evaluated MTStereo 2.0 on the following benchmarks: Middlebury 2014 [26], Kitti 2015 [19], Trimbot2020 Synthetic Garden [23] and Real Garden [25] (test sets used by [23]), Driving (cleanpass, fast, 35mm_focallength subset), Monkaa (cleanpass subset), and Flying Things 3D [18] (cleanpass, test set - stereo pairs excluded by [18] were excluded in our experiments as well). Marked results were taken from the online Kitti[1] and Middlebury[2] leaderboard. For Middlebury, accuracy results were weighted by the official weights, while average density results were not weighted. We used full-size images for our experiments. We ran our algorithm on an Intel® Core™ i7-2600K CPU @3.40GHz. We used 4 cores.

3.1 Evaluation

We computed standard metrics for the concerned benchmarks, allowing direct comparison with existing methods:

[1] http://www.cvlibs.net/datasets/kitti/eval_scene_flow.php?benchmark=stereo.
[2] https://vision.middlebury.edu/stereo/.

Table 1. Results on the Middlebury training and test set in *avgerr*(*Density*), compared with those of other methods, taken from the online leaderboard. Results of MTStereo 2.0 are bold. * Result was computed on downsized input image pairs.

Middlebury 2014 (train set)							
MotionStereo [31]	SGM [11]	**Sparse**	SNCC [7]	LPSM [34]	SDR [35]	Cens5 [13]	LS-ELAS [15]
1.72*(46%)	2.06(58%)	2.35(3%)	3.25*(62%)	4.22*(100%)	4.32*(100%)	4.32*(67%)	4.35(61%)
ELAS [10]	SLCCF	SED [22]	**Semi-Dense**	MANE [32]	SRM [21]	SGBM1 [12]	REAF [5]
4.94(73%)	5.27*(100%)	5.38(1%)	6.51(24%)	6.59*(67%)	6.92*(100%)	7.83*(78%)	8.49*(100%)
Middlebury 2014 (test set)							
SGM [11]	MotionStereo [31]	SNCC [7]	Cens5 [13]	SDR [35]	SLCCF	MANE [32]	LS-ELAS [15]
2.50(50%)	3.30*(38%)	3.96*(55%)	5.31*(61%)	6.16*(100%)	6.36*(100%)	8.70*(61%)	9.10(49%)
LPSM [34]	**Sparse**	SRM [21]	ELAS [10]	TSGO [20]	SED [22]	ELAS_RVC [10]	SGBM1 [12]
9.29*(100%)	9.36(3%)	10.40*(100%)	10.6(66%)	11.00(100%)	12.3(2%)	13.40*(100%)	14.20*(73%)

- **avgerr**: average absolute disparity error (in pixels) among all pixels of which both a disparity value was in the ground truth and estimated disparity map.
- **D-all-est**: the percentage of stereo disparity outliers (the disparity error is $\geq$3px and $\geq$5% of the true disparity) among all pixels of which both a disparity value was in the ground truth and estimated disparity map.
- **Density**: the rounded percentage of pixels with a disparity prediction with respect to the total number of pixels in the reference image.

We defined a single set of parameters that contributes to achieving robust performance across the different benchmark data sets. The number of color quantization levels q was set to 16 (8) for sparse (semi-dense) disparity maps. The weight α of the context cost relative to that of the gradient cost was set to 0.8. The minimum (or maximum) width of nodes to be matched ω_α (or ω_β) was set to 0 (or 1/2 of the image width). Matched node levels S was set to $\{1, 0\}$. The maximum neighborhood size ω_γ was set to 10. The size of the Gaussian kernel used to aggregate the cost volume ω_{cv} was 21. The minimum confidence ω_Π was set to 12 during coarse-to-fine matching. In guided pixel refinement, ω_Π was set to 12% (4%) for sparse (semi-dense) disparity maps. ω_ω was set to 15%.

3.2 Results and Comparison

In Table 1, we report the error achieved by our method on the Middlebury training and test data, as well as those of other methods that have low average execution times and do not run on a GPU. The lower (higher) the avgerr (Density) values, the better the performance. In Table 2, we report the results achieved by our method on the other considered benchmark data sets, as well as those of other methods that do not formulate the stereo matching as a learning problem (upper part) and those that deploy neural networks to tackle disparity estimation (lower part). Both the sparse and semi-dense versions of our method achieved better or comparable accuracy than those of many existing methods. The results on the Middlebury training data set show, for example, the avgerr in sparse disparity maps produced by MTStereo 2.0 is lower than that of all other listed methods except MotionStereo and SGM. Also, the avgerr on the

Table 2. Results in *avgerr*, except Kitti Test which is expressed in *D-all-est*. Average prediction *Density* is reported in brackets. Entries without density specification are 100% dense. "-SD" denotes Semi-Dense. $^{+}$: Result was taken from benchmark website. ^{d}: Result was computed on downsized input images. ^{g}: Generalization study result. *: Result was (possibly) computed on a different set of images than we used for evaluation. ^{t}: Result was (possibly) computed using a different metric computation approach.

	Middlebury		Kitti 2015		Real garden	Synth garden	Driving	Monkaa	Flying things
	Train	Test	Train	Test					
MTS2-Sparse	**2.35**$^{+}$(3)	**9.36**$^{+}$(3)	1.35(4)	7.83^{+}(4)	3.08(5)	5.22(6)	**5.36**(3)	**3.83**(4)	**1.7**(6)
MTS2-SD	6.51(24)	–	2.24(25)	–	3.72(20)	7.04(19)	7.38(15)	6.54(25)	2.61(30)
MTS1-Sparse	5.47^{+}(2)	15.5^{+}(2)	1.58(2)	8.92^{+}(3)	2.48(2)	7.32(2)	8.8(1)	7.22(3)	3.03(4)
MTS1-SD	17.47(57)	–	4.47(44)	–	3.78(18)	12.8(14)	16.2(38)	15.41(40)	6.93(58)
SGBM1 [12]	7.83^{+}(67)	16.3^{+}(63)	1.45(84)	–	2.61(89)	4.77(92)	16.06(70)	11.37(81)	5.14(86)
SGBM2 [12]	8.92^{+d}(83)	15.9^{+d}(77)	**1.27**(82)	**5.86**$^{+}$(90)	2.16(90)	**4.67**(90)	15.72(64)	10.5(78)	4.58(85)
ELAS_ROB [10]	10.5^{+d}	13.4^{+d}	1.49(99)	9.67^{+}	**2.06**	7.02	11.71	17.75	7.46
FPGA Stereo [14]	–	–	–	–	2.94 [23]	11.41 [23]	–	–	–
DispNet [18]	–	–	**0.68** [18]	4.34^{+}	**1.35** [23]	**6.28** [23]	**15.62*** [18]	**5.78** [18]	1.68 [18]
EdgeStereo [28]	**2.00**$^{+}$	**3.72**$^{+}$	2.07^{g} [28]	**2.08**$^{+}$	–	–	–	–	**0.74**^{*t} [28]
iResNet [16]	–	–	1.21^{g} [28]	2.44 [28]	–	–	–	–	0.95^{*t} [28]

Kitti training data set of MTStereo 2.0 sparse is lower than that of all other listed methods except DispNet, iResNet, and SGBM2. Furthermore, the avgerr of disparity maps produced by MTStereo 2.0 sparse is lower than all other non-learning based methods when evaluated on Driving, Monkaa, and Flying Things 3D. MTStereo 2.0 sparse produced more accurate disparity maps than some of the listed CNN based methods on the Kitti2015 training, Synthetic garden, Driving, and Monkaa data sets. The density of the disparity maps of MTStereo2.0 is lower than CNN-based methods. However, it can be regulated by manipulating the parameters that trade-off accuracy and density. These results indicate that MTStereo2.0 estimates very precisely the depth of regions for which a robust match in the max-trees is found, and can be used as an unsupervised matching strategy to serve as prior for more sophisticated methods.

The results of MTStereo 2.0 sparse and semi-dense are generally more accurate than those of their counterpart based on MTStereo 1.0. The cases where MTStereo 2.0 produces more accurate and dense disparity maps than MTStereo 1.0 (e.g. for the sparse variants when evaluated on the Middlebury, Kitti2015, Synth garden, and Driving data sets) suggest that MTStereo 2.0 is a more robust method than MTStereo 1.0.

In Fig. 3, we show example images from the Middlebury, Trimbot2020 Synthetic Garden, Monkaa, and Flying Things 3D data sets, with corresponding ground truth depth images, our semi-dense and sparse depth reconstruction. One can notice that our method can robustly extract depth information also in image regions with little texture, whose depth ambiguity can be successfully handled by MTStereo2.0. For example, the depth of the gray cubes in the Flying Things example is reasonably well estimated although the cube surfaces are rendered with little texture. When the parameter ω_{Π} is set to a low value, an assumption is made that regions with little texture are flat. The dis-ambiguous

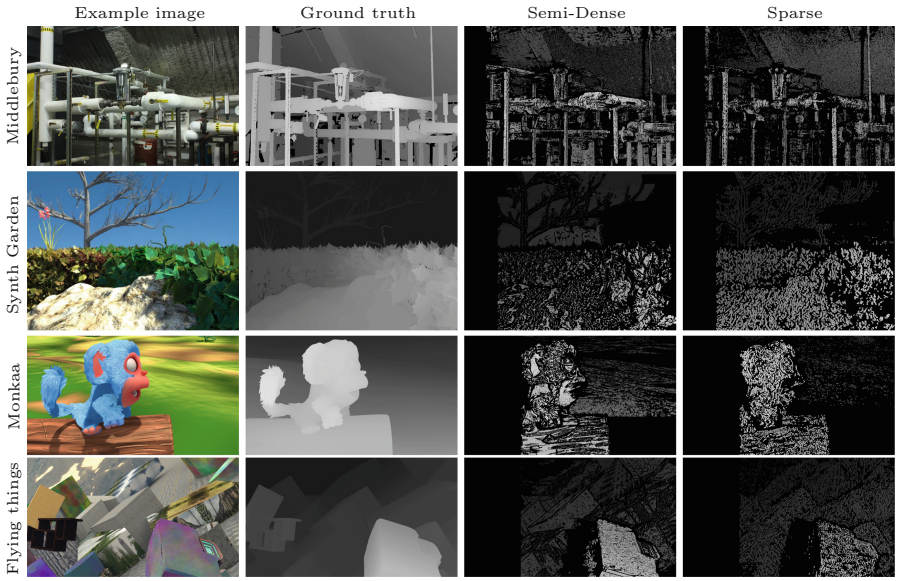

Fig. 3. Example images from the Middlebury, Trimbot2020 Synthetic Garden, Monkaa, and Flying Things 3D data sets, with corresponding ground truth images, our semi-dense and sparse reconstruction. Black disparity map pixels have no assignment. Morphological dilatation was applied to sparse outputs for visualization purposes.

disparity values at the left and right side edges of regions with little or no texture are then linearly interpolated within the region. When the assumption is not correct, such as between the tree branches in the MTStereo semi-dense disparity map of the Synth Garden example, the parameter ω_Π can be increased such that no disparity values are estimated in ambiguous regions. Edge blurring artifacts are frequently filtered out, as can be seen for the edges of the pipes in the first row or those of the tree in the second row of Fig. 3. The disparity maps produced by MTStereo2.0 tend to have very few outliers.

The methodological improvements of MTStereo2.0 with respect to version 1.0 increase the computation intensity of the method, which requires a longer time to process the images. The processing times of MTStereo 2.0 are generally longer than those of other methods listed in this paper. However, the efficiency of our implementation (publicly available) can likely be increased through code-level optimization, e.g. of our cost volume construction and coarse-to-fine matching procedures that take up to about 70% of the processing time.

4 Conclusions

We proposed a stereo matching method, MTStereo 2.0, for systems with limited computational and energy resources that require efficient and accurate depth estimation. It improves on its predecessor MTStereo 1.0 as it: *a)* deploys a more

robust cost function, *b)* performs more thorough detection of incorrect matches, *c)* computes disparity maps with pixel-level rather than node-level precision.

MTStereo 2.0 produces disparity maps which are generally more accurate than those produced by MTStereo 1.0, and does not require intensive GPU computations like methods based on CNNs. It can thus run on devices with low-power resources. Our method achieves competitive results on several benchmark data sets: Middlebury 2014, KITTI 2015, Driving, FlyingThings3D, Monkaa, and the TrimBot2020 garden data sets.

References

1. Arnold, R.D.: Automated stereo perception. Stanford Univ CA Dept of Computer Science, Technical report (1983)
2. Brandt, R., Strisciuglio, N., Petkov, N., Wilkinson, M.H.: Efficient binocular stereo correspondence matching with 1-D max-trees. Pattern Recogn. Lett. **135**, 402–408 (2020)
3. Chen, D., Ardabilian, M., Wang, X., Chen, L.: An improved non-local cost aggregation method for stereo matching based on color and boundary cue. In: IEEE ICME, pp. 1–6 (2013)
4. Chen, Z., Sun, X., Wang, L., Yu, Y., Huang, C.: A deep visual correspondence embedding model for stereo matching costs. In: IEEE ICCV, pp. 972–980 (2015)
5. Cigla, C.: Recursive edge-aware filters for stereo matching. In: CVPRW, pp. 27–34 (2015)
6. Cohen, L., Vinet, L., Sander, P.T., Gagalowicz, A.: Hierarchical region based stereo matching. In: IEEE CVPR, pp. 416–421 (1989)
7. Einecke, N., Eggert, J.: A two-stage correlation method for stereoscopic depth estimation. In: DICTA, pp. 227–234. IEEE (2010)
8. Engel, J., Stückler, J., Cremers, D.: Large-scale direct slam with stereo cameras. In: IEEE/RSJ IROS, pp. 1935–1942. IEEE (2015)
9. Fua, P.: A parallel stereo algorithm that produces dense depth maps and preserves image features. Mach. Vis. Appl. **6**(1), 35–49 (1993)
10. Geiger, A., Roser, M., Urtasun, R.: Efficient large-scale stereo matching. In: Kimmel, R., Klette, R., Sugimoto, A. (eds.) ACCV 2010. LNCS, vol. 6492, pp. 25–38. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19315-6_3
11. Hirschmuller, H.: Accurate and efficient stereo processing by semi-global matching and mutual information. In: CVPR, vol. 2, pp. 807–814 (2005)
12. Hirschmuller, H.: Stereo processing by semiglobal matching and mutual information. IEEE Trans. Pattern Anal. Mach. Intell. **30**(2), 328–341 (2008)
13. Hirschmüller, H., Innocent, P.R., Garibaldi, J.: Real-time correlation-based stereo vision with reduced border errors. IJCV **47**(1–3), 229–246 (2002)
14. Honegger, D., Sattler, T., Pollefeys, M.: Embedded real-time multi-baseline stereo. In: ICRA 2017, pp. 5245–5250. IEEE (2017)
15. Jellal, R.A., Lange, M., Wassermann, B., Schilling, A., Zell, A.: LS-ELAS: line segment based efficient large scale stereo matching. In: ICRA, pp. 146–152 (2017)
16. Liang, Z., et al.: Learning deep correspondence through prior and posterior feature constancy. arXiv preprint arXiv:1712.01039 (2017)
17. Luo, X., Bai, X., Li, S., Lu, H., Kamata, S.I.: Fast non-local stereo matching based on hierarchical disparity prediction. arXiv preprint arXiv:1509.08197 (2015)

18. Mayer, N., et al.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: CVPR 2016, pp. 4040–4048 (2016)
19. Menze, M., Heipke, C., Geiger, A.: Joint 3D estimation of vehicles and scene flow. In: ISPRS Workshop on Image Sequence Analysis (ISA) (2015)
20. Mozerov, M.G., van de Weijer, J.: Accurate stereo matching by two-step energy minimization. IEEE Trans. Image Process. **24**(3), 1153–1163 (2015)
21. Okae, J., Du, J., Hu, Y.: Robust statistical approach to stereo disparity maps denoising and refinement. Control Theory Technol. **18**(4), 348–361 (2020). https://doi.org/10.1007/s11768-020-00014-y
22. Peña, D., Sutherland, A.: Disparity estimation by simultaneous edge drawing. In: Chen, C.-S., Lu, J., Ma, K.-K. (eds.) ACCV 2016. LNCS, vol. 10117, pp. 124–135. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54427-4_10
23. Pu, C., Song, R., Tylecek, R., Li, N., Fisher, R.: SDF-MAN: semi-supervised disparity fusion with multi-scale adversarial networks. Remote Sens. **11**(5), 487 (2019)
24. Salembier, P., Oliveras, A., Garrido, L.: Antiextensive connected operators for image and sequence processing. IEEE Trans. Image Process. **7**(4), 555–570 (1998)
25. Sattler, T., Tylecek, R., Brox, T., Pollefeys, M., Fisher, R.B.: 3D reconstruction meets semantics-reconstruction challenge 2017. In: ICCV Workshop (2017)
26. Scharstein, D., et al.: High-resolution stereo datasets with subpixel-accurate ground truth. In: Jiang, X., Hornegger, J., Koch, R. (eds.) GCPR 2014. LNCS, vol. 8753, pp. 31–42. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11752-2_3
27. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. Comput. Vis. **47**(1–3), 7–42 (2002)
28. Song, X., Zhao, X., Fang, L., Hu, H.: Edgestereo: an effective multi-task learning network for stereo matching and edge detection. arXiv:1903.01700 (2019)
29. Sun, C.: A fast stereo matching method. In: DICTA, pp. 95–100. Citeseer (1997)
30. Todorovic, S., Ahuja, N.: Region-based hierarchical image matching. Int. J. Comput. Vision **78**(1), 47–66 (2008)
31. Valentin, J., et al.: Depth from motion for smartphone AR. In: SIGGRAPH Asia, p. 193. ACM (2018)
32. Vázquez-Delgado, H.D., et al.: Real-time multi-window stereo matching algorithm with fuzzy logic. IET Comput. Vision **15**(3), 208–223 (2021)
33. Wilkinson, M.H.: A fast component-tree algorithm for high dynamic-range images and second generation connectivity. In: IEEE ICIP, pp. 1021–1024 (2011)
34. Xu, C., Wu, C., Qu, D., Xu, F., Sun, H., Song, J.: Accurate and efficient stereo matching by log-angle and pyramid-tree. IEEE Trans. Circuits Syst. Video Technol. (2020). https://doi.org/10.1109/TCSVT.2020.3044891
35. Yan, T., Gan, Y., Xia, Z., Zhao, Q.: Segment-based disparity refinement with occlusion handling for stereo matching. IEEE TIP **28**(8), 3885–3897 (2019)
36. Yang, Q., Ji, P., Li, D., Yao, S., Zhang, M.: Fast stereo matching using adaptive guided filtering. Image Vis. Comput. **32**(3), 202–211 (2014)
37. Yoon, K.J., Kweon, I.S.: Adaptive support-weight approach for correspondence search. IEEE Trans. Pattern Anal. Mach. Intell. **28**(4), 650–656 (2006)
38. Zbontar, J., LeCun, Y., et al.: Stereo matching by training a convolutional neural network to compare image patches. J. Mach. Learn. Res. **17**(1–32), 2 (2016)
39. Zhang, K., Lu, J., Lafruit, G.: Cross-based local stereo matching using orthogonal integral images. IEEE Trans. Circuits Syst. Video Technol. **19**(7), 1073–1079 (2009)