Triple-SGM: Stereo Processing using Semi-Global Matching with Cost Fusion

Jan Kallwies Torsten Engler Bianca Forkel Hans-Joachim Wuensche
Autonomous Systems Technology (TAS)
University of the Bundeswehr Munich
{jan.kallwies, torsten.engler, bianca.forkel, jw}@unibw.de

Abstract

In this work, we propose an extension of the Semi-Global Matching framework for three images from a triplet-stereo rig consisting of a horizontal and vertical camera pair. After calculating the matching costs separately for both image pairs, these are merged at cost level using cubic spline interpolation. For cost values near the left/bottom image boundaries, we propose an advanced weighting strategy. Subsequently, the fused matching can be used directly for the cost aggregation and disparity estimation.

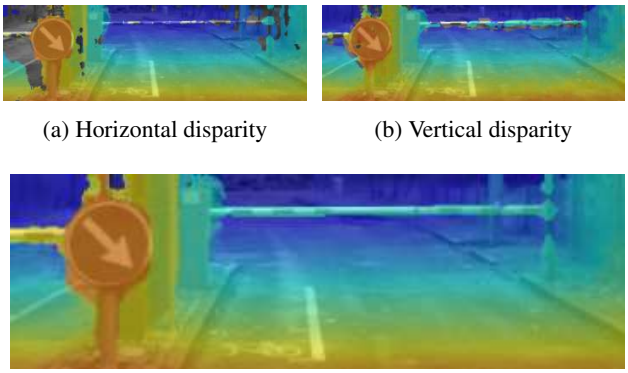
The benefits of the proposed fusion strategy are demonstrated by an evaluation based on synthetic and real-world data. To encourage further comparisons on triple stereo algorithms, the dataset used for evaluation is made publicly available.

1. Introduction

Today, 3D reconstruction and recognition is a vital task. Many technical systems depend on accurate and reliable 3D information. One possibility to provide this information is to use stereo vision systems. Compared to many other measurement principles, *e.g.* LiDAR or Radar, stereo systems have a high spatial resolution. Additionally, the color information for each point is directly available for further processing.

However, stereo vision systems have two major drawbacks: Firstly, the accuracy strongly depends on the distance to the object. Secondly, the aperture problem due to visually similar parts of the scene (*i.e.* periodic structures or objects with low texture) leads to wrong or missing measurements.

A solution is a larger baseline and the extension of the camera rig with a third camera. While horizontal stereo pairs are ideal to capture vertical structures, they often fail in capturing horizontal structures, as it can be seen in Fig. 1a. For that reason, the camera is placed above or below one of the existing cameras to create a new vertical stereo pair. Its baseline is aligned perpendicular to the baseline of the horizontal camera pair.



(a) Horizontal disparity (b) Vertical disparity

(c) Fused disparity (our result)

Figure 1: Disparity maps for one scene using a horizontal stereo pair, a vertical pair and our Triple-SGM. The disparity overlay colors range from red for high values to blue for low values. Note the missing disparities in (a) and (b).

As shown in Fig. 1b, the vertical pair is superior to the horizontal pair in the detection of horizontal structures while being poor at vertical structures. Fusing the results of those two stereo systems can combine the advantages of both, as it can be seen in Fig. 1c. It also improves matching of periodic structures, as it is more unlikely, that the periodicity is identical in both image directions.

The resulting disparities can be fused based on a confidence measure, *e.g.* the final matching cost [17]. However, we argue that fusion should be done as early in the algorithm as possible. With proper alignment of the cost maps, the cost can be fused directly, and the minimum search can be done on a single cost map. The benefit is an unambiguous result in the disparity. Moreover, regions which suffer from occlusion and have no valid cost entry in one stereo system may be completely visible in the other stereo rig.

Therefore we introduce an extension of the popular stereo framework Semi-Global Matching (SGM) [15] for the disparity estimation from three input images using cost fusion. Additionally, a trinocular rectification approach is described. We further provide a synthetic dataset for the evaluation of triple stereo algorithms. Our quantitative evaluation is based on this synthetic as well as real-world data.

2. Related work

One of the most popular and widely used stereo algorithms is Semi-Global Matching by Hirschmueller [15]. He introduced mutual information as a matching cost between pixels and implemented a cost aggregation across the entire image. Thereby, a global constraint for the disparity computation is enforced leading to much higher robustness and denseness of the disparity image. SGM achieved great results, and modified versions of it still appear in the upper ranks of various stereo vision challenges, *e.g*. the Middlebury challenge [21] or more recently, the Robust Vision Challenge [1].

To achieve real-time performance, SGM was ported to various hardware. Gehring *et al*. [8] for example use a FPGA. Many efforts have been made to implement SGM on graphics cards using OpenGL [7] or CUDA [10, 3, 19, 14]. Other work regarding SGM is done on its extension with Convolutional Neural Networks (CNN), *e.g*. [22, 5, 25]. These modern CNN methods deliver very good results, but are usually far too slow for real-time applications. In addition, they have to be explicitly trained for the case of application and thus need prior knowledge.

Another approach to improve the results of stereo matching is to use more than two cameras. Maitre *et al*. [18] use camera arrays to optimize the resulting depth structure. Additional cameras increase the processing load and make the algorithm difficult to apply in real-time scenarios. One of the most important factors for real-time applications of stereo algorithms is rectification. After rectification, corresponding image pixels are on the same row/column of the image, respectively. However, image rectification is only unambiguous for up to three cameras. More cameras can only be rectified if their optical centers lie on the same plane, leading to high production constraints of the camera array. Otherwise, the image rectification for every camera can only be computed in a least squares manner, thus introducing rectification errors.

A compromise between computational effort and the resulting depth quality can be achieved using a rack of three cameras. Furthermore, the rectification of three cameras is always well defined. Examples of trinocular image rectification can be found in [12] and [2]. Baik *et al*. offer an intuitive solution to the rectification problem but constrain the angle between the upper and the right image to $45°$ [2]. While this constraint simplifies the disparity computation, it is over-restrictive in our case.

In order to calculate the disparity in three camera images, different strategies can be applied. The most straightforward solution is the fusion of the resulting disparity images. Hirschmueller [15] suggested multi-baseline matching by separately computing the disparities for all image pairs and fusing them afterwards. He takes the mean value of the different disparities, weighted with the baseline of the corresponding stereo pair. In [17] the disparities are calculated separately and then fused based on a peakness confidence measure. However, late fusion is still susceptible to local optima in each of the calculations. Additionally, the runtime is increased, because each calculation step has to be executed twice.

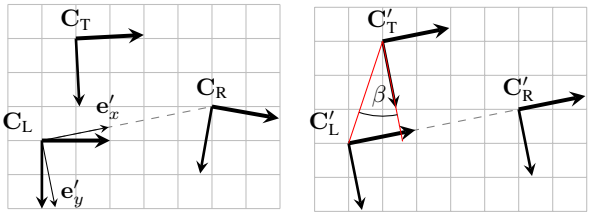
An alternative is a direct fusion on a cost basis. One of the first cost-based approaches was presented in [20]. The authors added the cost each disparity produced in both (horizontal and vertical) image pairs and searched for the globally optimal solution. The costs were based on the correlation of a window around the pixels without taking any global constraint into account. Heinrichs *et al*. proposed a cost fusion for image triplets based on SGM [13]. While the results are promising, the runtime is quite large (about $100\,\mathrm{s}$ per image triplet), and the algorithm is only applicable for camera triplets with two identical baseline lengths.

3. Trinocular rectification

Image rectification is the process of aligning the image planes of all cameras. We adapted the rectification process described in [2] for the use of arbitrary baselines. The goal of trinocular rectification is to align the epipolar lines with the horizontal and vertical axis of the image plane. This can be achieved by moving the epipoles to infinity what means in turn that the following constraints are satisfied:

1. The images planes of all three cameras are coplanar and parallel to the plane spanned by the three camera centers.
2. The x-axis of all cameras are parallel to the line between the left and right camera center.

In the following, we assume that the full calibration of all three cameras is known and that they can be modeled as pinhole cameras. We use the left camera as the common reference.



(a) Raw, unrectified cameras (b) Rectified cameras

Figure 2: Schematic illustration of the rectification process. The x-axis of each camera is shown as a bold arrow and the desired orientation of the rectified cameras is depicted by the basis vectors $\mathbf{e}'_x$ and $\mathbf{e}'_y$. The camera center points did not move by the rectification. Only the orientations of all three cameras have been modified.

Thus, the projection matrices are:

$$\mathbf{P}_\mathrm{L} = \mathbf{K}_\mathrm{L}\,[\mathbf{I}|\mathbf{0}] \tag{1}$$
$$\mathbf{P}_\mathrm{R} = \mathbf{K}_\mathrm{R}\,[\mathbf{R}_\mathrm{R}|-\mathbf{C}_\mathrm{R}] \tag{2}$$
$$\mathbf{P}_\mathrm{T} = \mathbf{K}_\mathrm{T}\,[\mathbf{R}_\mathrm{T}|-\mathbf{C}_\mathrm{T}]\,, \tag{3}$$

where $\mathbf{K}_\mathrm{L}$, $\mathbf{K}_\mathrm{R}$ and $\mathbf{K}_\mathrm{T}$ denote the camera matrices, $\mathbf{R}_\mathrm{R}$ and $\mathbf{R}_\mathrm{T}$ the rotation matrices of the right and the top camera relative to the left camera and $\mathbf{C}_\mathrm{R}$ and $\mathbf{C}_\mathrm{T}$ the camera center center points relative to the left camera.

In order to satisfy the constraints mentioned above, a common orientation of all three cameras is required. That orientation can be described by the three basis vectors $\mathbf{e}'_x$, $\mathbf{e}'_y$ and $\mathbf{e}'_z$, which can be determined as follows [2]:

1. The new x-axis is parallel to the horizontal baseline.

$$\mathbf{e}'_x = \frac{\mathbf{C}_\mathrm{R}}{||\mathbf{C}_\mathrm{R}||} \tag{4}$$

2. The new z-axis is perpendicular to the plane spanned by the three camera centers.

$$\mathbf{e}'_z = \frac{\mathbf{C}_\mathrm{R} \times -\mathbf{C}_\mathrm{T}}{||\mathbf{C}_\mathrm{R} \times -\mathbf{C}_\mathrm{T}||} \tag{5}$$

3. The new y-axis is determined by the new x- and z-axis.

$$\mathbf{e}'_y = \mathbf{e}'_z \times \mathbf{e}'_x \tag{6}$$

Thus, the common rotation matrix for all three rectified cameras is given by:

$$\mathbf{R}' = \begin{bmatrix}\mathbf{e}'_x & \mathbf{e}'_y & \mathbf{e}'_z\end{bmatrix}^\top. \tag{7}$$

Please note, that this matrix is completely determined by the positions of the camera centers and has no degree of freedom.

The intrinsic parameters of the rectified cameras can be chosen freely. However, they have to be equal for all three cameras in order to satisfy the constraints for rectification:

$$\mathbf{K}' = \begin{bmatrix} f_u & s & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}. \tag{8}$$

Since real-world camera setups are never perfectly aligned, *i.e.* the angle between the right, left and top camera is not exactly 90°, the epipolar lines of the rectified vertical camera pair would not be aligned vertically (see angle β in Fig. 2b). In order to compensate that, a skew factor (also called shearing factor) has to be chosen appropriately:

$$s = \frac{\mathbf{C}'_{\mathrm{T}[x]}}{-\mathbf{C}'_{\mathrm{T}[y]}} = \tan\beta. \tag{9}$$

Here, the indices $[x]$ and $[y]$ depict the respective components of a vector, and $\mathbf{C}'_\mathrm{T}$ is the camera center of the top camera represented in the coordinate frame of the rectified left camera:

$$\mathbf{C}'_\mathrm{T} = \mathbf{R}' \cdot \mathbf{C}_\mathrm{T}. \tag{10}$$

The other intrinsic parameters, namely the focal lengths and the optical centers can be chosen freely and depend on the actual hardware setup.

Thus, the final rectified projection matrices are:

$$\mathbf{P}'_\mathrm{L} = \mathbf{K}'\mathbf{R}'\,[\mathbf{I}|\mathbf{0}] \tag{11}$$
$$\mathbf{P}'_\mathrm{R} = \mathbf{K}'\mathbf{R}'\,[\mathbf{R}_\mathrm{R}|-\mathbf{C}_\mathrm{R}] \tag{12}$$
$$\mathbf{P}'_\mathrm{T} = \mathbf{K}'\mathbf{R}'\,[\mathbf{R}_\mathrm{T}|-\mathbf{C}_\mathrm{T}]\,. \tag{13}$$

With the given rectified camera calibrations, the images can be transformed accordingly by applying the following homographies to the images (see *e.g.* [11]):

$$\mathbf{H}'_i = \mathbf{K}' \cdot \mathbf{R}' \cdot {\mathbf{K}_i}^{-1} \quad \text{with} \quad i \in \{\mathrm{L,R,T}\}\,. \tag{14}$$

4. Triple-SGM

An overview of our algorithm performing semi-global matching on three images is given in the flowchart of Fig. 3. It comprises the following steps (bold points indicate the difference to the original SGM approach):

1. Rectification of all **three** images.
2. **Rotation of left and top image by 90°.**
3. Census transform of all **four** images.
4. Calculation of horizontal (left ↔ right) and **vertical (left rotated ↔ top rotated) matching costs**.
5. **Interpolation and summation of matching costs**.
6. Cost accumulation along image paths.
7. Minimum search (winner takes it all).
8. Optional post-processing.

4.1. Semi-global matching

Before describing our approach to process three images, we introduce Semi-Global Matching (SGM) [15].
The energy function of the disparity image $\mathbf{D}$ is defined as

$$\begin{aligned} E(\mathbf{D}) = \sum_{\mathbf{p}} \Big(& C(\mathbf{p}, d_\mathbf{p}) \\ & + \sum_{\mathbf{q}\in N_\mathbf{p}} P_1 T[|d_\mathbf{p} - d_\mathbf{q}| = 1] \\ & + \sum_{\mathbf{q}\in N_\mathbf{p}} P_2 T[|d_\mathbf{p} - d_\mathbf{q}| > 1] \Big), \end{aligned} \tag{15}$$

where P_1 and P_2 are penalty factors, $\mathbf{p}$ and $\mathbf{q}$ are image points and $N_\mathbf{p}$ is a local region around an image point.

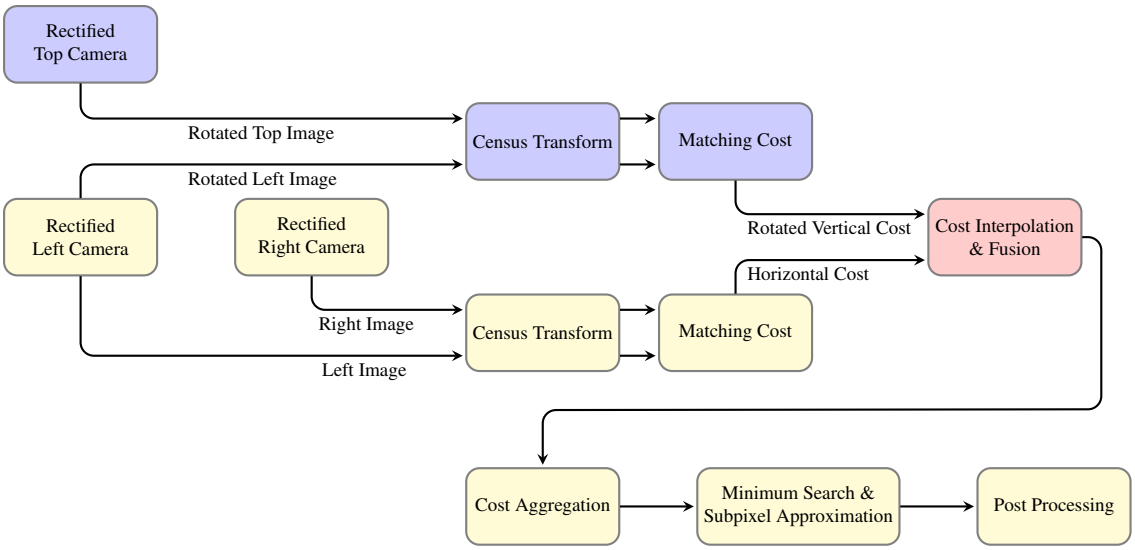


Figure 3: Flowchart of the presented triple SGM algorithm. The yellow boxes show the components of the standard SGM pipeline for a horizontal stereo pair. The blue boxes represent the additional components needed for vertical stereo. The red box forms the core of our contribution – the efficient fusion of the horizontal and vertical information at cost level. Thus, the red and blue boxes together represent the algorithmic contribution of this paper.

The first summand in Eq. (15) is the matching cost for a certain image position and disparity. The second one represents the penalty P_1 for disparity differences between adjacent pixels of 1, i.e. for slanted surfaces. The third summand introduces a penalty of P_2 for disparity discontinuities. Here, the function $T[\cdot]$ is the Kronecker delta returning 1 if the specified condition is met, and 0 otherwise.

The direct minimization of $E(\mathbf{D})$ is computationally extremely demanding. However, it can be approximated and solved efficiently by following 1D paths [15]. The cost of an arbitrary path along the direction $\mathbf{r}$ is given by

$$\begin{aligned} L_{\mathbf{r}}(\mathbf{p},d) = C(\mathbf{p},d) + \min\big(&L_{\mathbf{r}}(\mathbf{p}-\mathbf{r},d),\\ &L_{\mathbf{r}}(\mathbf{p}-\mathbf{r},d-1)+P_1,\\ &L_{\mathbf{r}}(\mathbf{p}-\mathbf{r},d+1)+P_1,\\ &\min_{i\neq d\pm 1} L_{\mathbf{r}}(\mathbf{p}-\mathbf{r},i)+P_2\big) - \min_k L_{\mathbf{r}}(\mathbf{p}-\mathbf{r},k). \end{aligned} \tag{16}$$

Adding the aggregated cost of all paths leads to an approximation of the original global energy function and can finally be used to compute the disparity at an image position $\mathbf{p}$ using the winner-takes-all-strategy:

$$\mathbf{D}(\mathbf{p}) = \arg\min_d \sum_{\mathbf{r}} L_{\mathbf{r}}(\mathbf{p},d). \tag{17}$$

4.2. Matching cost

The SGM algorithm requires an underlying cost function $C(\mathbf{p},d_{\mathbf{p}})$ to compare pixels based on their neighborhood. It is important that this cost function is robust against illumination and brightness changes. Possible choices for the cost function are mutual information [23], census transform or rank transform (both introduced in [24]). It has been shown, that census transform outperforms rank transform in terms of SGM matching accuracy [4] and that it offers good performance under illumination changes [9]. Additionally, it is straightforward to compute in parallel. This is why we use the census transform in this work. However, any other matching cost could be used as well.

Census transform The census signature is a bit string describing the surroundings of each pixel. Given a pixel at the image location i,j with the gray value $g_{(i,j)}$ the census string is defined as:

$$\begin{aligned} \mathcal{C}_{(i,j)} = \big[\quad \ldots \quad \mathcal{H}(g_{(i+l_\mathrm{v},j+l_\mathrm{h})} - g_{(i,j)}) \quad \ldots \quad \big] \\ \text{with } -0.5\,(w_\mathrm{v}-1) \le l_\mathrm{v} \le 0.5\,(w_\mathrm{v}-1) \\ \text{and } -0.5\,(w_\mathrm{h}-1) \le l_\mathrm{h} \le 0.5\,(w_\mathrm{h}-1)\,. \end{aligned} \tag{18}$$

Here, w_h and w_v are the odd horizontal and vertical sizes of the surrounding window, and the heavyside function $\mathcal{H}(\cdot)$ is given by

$$\mathcal{H}(x) := \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \ge 0\,. \end{cases} \tag{19}$$

Horizontal and vertical matching cost The matching cost between two images is based on the census strings calculated for each pixel. For each pixel in the left (resp. rotated left) image $g_{(i,j),\mathrm{L}}$ and each disparity $0 \le d < 128\,\mathrm{px}$

the matching cost is computed by counting the number of different bits in both census strings

$$C_{\mathrm{h}}(i,j,d_{\mathrm{h}}) = \Delta_{\mathrm{hamming}}\left(\mathcal{C}_{(i,j),\mathrm{L}}, \mathcal{C}_{(i,j+d_{\mathrm{h}}),\mathrm{R}}\right). \tag{20}$$

The result $C_{\mathrm{h}}(i,j,d_{\mathrm{h}})$ is a matching cost for each pixel and possible disparity. The same is done for the vertical pair:

$$C_{\mathrm{v}}(i,j,d_{\mathrm{v}}) = \Delta_{\mathrm{hamming}}\left(\mathcal{C}_{(i,j),\mathrm{L}}, \mathcal{C}_{(i+d_{\mathrm{v}},j),\mathrm{T}}\right). \tag{21}$$

4.3. Fusion of matching costs for three images

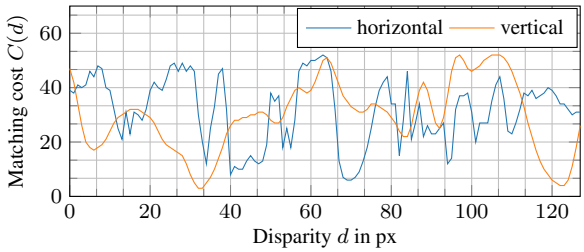
The most straightforward solution to the cost addition problem would be computing each step of SGM separately and adding the vertical and horizontal costs right before the winner takes it all step. However, this leads to an unnecessary increase in runtime (compare Section 5.4) because the smoothing step is hard to parallelize. Therefore, the cost addition is done before accumulating the paths of the SGM algorithm, avoiding duplicate accumulation:

$$\begin{aligned} C(i,j,d_{\mathrm{h}}) = {} & 0.5 \cdot C_{\mathrm{h}}\left(i,j,d_{\mathrm{h}}\right) + \\ & 0.5 \cdot C_{\mathrm{v}}\left(i,j,f_{\mathrm{b}} \cdot d_{\mathrm{h}}\right) \end{aligned} \tag{22}$$

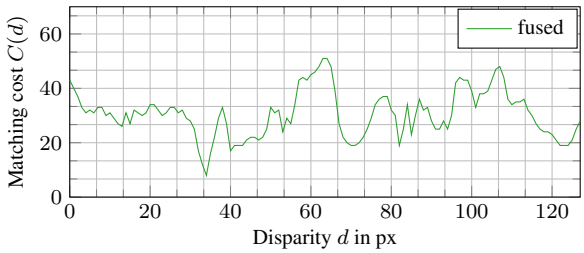
with the scaling factor f_{b} resulting from the different baselines of the horizontal and vertical camera pair b_{h} and b_{v}:

$$f_{\mathrm{b}} = \frac{b_{\mathrm{v}}}{b_{\mathrm{h}}}. \tag{23}$$

Figure 4 shows an example how the fusion of matching costs can resolve ambiguities.



(a) Horizontal and vertical census matching costs for a single pixel.



(b) Fused census matching cost for a single pixel.

Figure 4: Though the horizontal and vertical matching costs alone would not lead to the correct disparity value, the fused cost function shows a unique and correctly located peak.

4.3.1 Cost interpolation

The evaluation of C_{v} in Eq. (22) is in general done at non-integer disparity values $f_{\mathrm{b}} \cdot d_{\mathrm{h}}$, whereas the actual costs are only available at integer steps (see Eq. (21)). The most straightforward approach would be to do nearest-neighbor interpolation. However, this leads to significant problems in the final results (see Fig. 5).

Therefore, we propose to use cubic Hermite splines (cSplines) for the interpolation:

$$\begin{aligned} C_{\mathrm{v}}\left(i,j,f_{\mathrm{b}} \cdot d_{\mathrm{h}}\right) = {} & (2t^3 - 3t^2 + 1)p_0 \\ & + (t^3 - 2t^2 + t)m_0 \\ & + (-2t^3 + 3t^2)p_1 \\ & + (t^3 - t^2)m_1 \end{aligned} \tag{24}$$

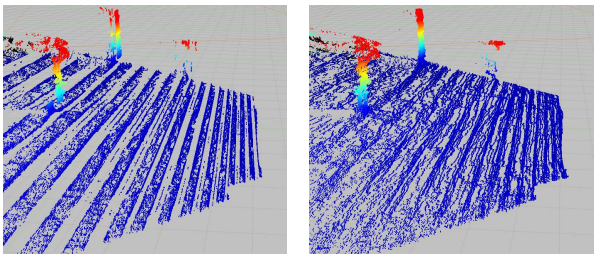
with

$$\begin{aligned} t &= f_{\mathrm{b}} \cdot d_{\mathrm{h}} - \lfloor f_{\mathrm{b}} \cdot d_{\mathrm{h}} \rfloor \\ p_0 &= C_{\mathrm{v}}\left(i,j,\lfloor f_{\mathrm{b}} \cdot d_{\mathrm{h}} \rfloor\right) \\ p_1 &= C_{\mathrm{v}}\left(i,j,\lfloor f_{\mathrm{b}} \cdot d_{\mathrm{h}} \rfloor + 1\right) \\ m_0 &= \frac{p_0 - p_{-1}}{2} + \frac{p_1 - p_0}{2} \\ m_1 &= \frac{p_1 - p_0}{2} + \frac{p_2 - p_1}{2} \end{aligned} \tag{25}$$

and

$$\begin{aligned} p_{-1} &= C_{\mathrm{v}}\left(i,j,\lfloor f_{\mathrm{b}} \cdot d_{\mathrm{h}} \rfloor - 1\right) \\ p_2 &= C_{\mathrm{v}}\left(i,j,\lfloor f_{\mathrm{b}} \cdot d_{\mathrm{h}} \rfloor + 2\right). \end{aligned} \tag{26}$$

Note that values of C_{v} outside the valid disparity range are padded.



(a) Resulting 3D point cloud without cSpline fit of the vertical cost vectors.

(b) Resulting 3D point cloud with cSpline fit of the vertical cost vectors.

Figure 5: Influence of the cSpline fit of the cost vectors. Without the interpolation, the necessary scaling of the baseline and thus of the disparity values leads to significant pixel-locking effects.

4.3.2 Handling of image boundary regions

In stereo algorithms, the disparity for parts of the left image cannot be calculated correctly. Assume an object close to the stereo rig with a high disparity d, located close to the left boundary of the left image. It would only be visible in the right image if it holds $j > d$. With a maximum disparity of $127\,\mathrm{px}$, the first $127\,\mathrm{px}$ of the left image cannot be calculated safely. Some algorithms like OpenCV Blockmatching crop this part of the disparity image, which is the save choice to do. Others interpolate and generate inaccurate results at best, and wrong results (if they miss an object entirely) at worst, *e.g.* [14]. The left-right consistency check, proposed by Hirschmüller [15] is able to remove most of the wrong results.

In the triplet case, more options are possible. The invalid area in one pair (horizontal or vertical) is often valid in the other. We propose to augment the costs in the invalid area with confidence measures a_v and $a_\mathrm{h} = 2 - a_\mathrm{v}$, in the range of $[0, 2]$ decreasing linearly to the edge of the image. The cost is weighted according to the confidences:

$$\begin{aligned} C(i,j,d_\mathrm{h}) = a_\mathrm{h} \cdot 0.5 \cdot C_\mathrm{h}(i,j,d_\mathrm{h}) + \\ a_\mathrm{v} \cdot 0.5 \cdot C_\mathrm{v}(i,j,f_\mathrm{b} \cdot d_\mathrm{h}) \end{aligned} \tag{27}$$

with

$$a_\mathrm{v} = \frac{1}{128} \begin{cases} i_\mathrm{rot} & \text{if } i_\mathrm{rot} < 128 \wedge j \geq 128 \\ 2 \cdot 128 - j & \text{if } i_\mathrm{rot} \geq 128 \wedge j < 128 \\ i_\mathrm{rot} - j + 128 & \text{if } i_\mathrm{rot} < 128 \wedge j < 128 \\ 128 & \text{otherwise} \end{cases} \tag{28}$$

and

$$i_\mathrm{rot} = h - i - 1. \tag{29}$$

5. Experimental results

The implementation we utilized to evaluate the effectiveness of our Triple-SGM algorithm as described in this section is based on libSGM, available at https://github.com/fixstars/libSGM. We used cost aggregation along four paths (aligned horizontally and vertically) with penalties of $P_1 = 10$ and $P_2 = 200$ (see Eq. (15)).

Subpixel approximation is done using a local parabola fit of the accumulated costs. Furthermore, we use a median filter on the final disparity image as suggested in [16] with a window size of $5\,\mathrm{px}$. Only in case of purely horizontal or vertical stereo matching, a left-right consistency check [16] is performed.

We compare our Triple-SGM based on cost fusion to the disparity fusion proposed in [17] applied to standard SGM disparity images.

5.1. Experimental setup

We add a third camera above the left camera to an autonomous vehicle and thus create a vertical stereo pair with a baseline perpendicular to the existing horizontal stereo pair. To obtain good precision, the horizontal baseline is chosen relatively large with $80\,\mathrm{cm}$, while the vertical baseline is limited to $20\,\mathrm{cm}$ due to the mechanical setup. For comparability between the stereo pairs, the disparity errors of the vertical pair are scaled by factor 4 accordingly.

To evaluate the triplet stereo approach, we created a synthetic dataset using the autonomous driving simulator CARLA [6]. This way, we obtain accurate ground truth depth information for quantitative evaluation. Some example images of the dataset can be seen in Table 2. The complete synthetic dataset containing 2000 scenes with three camera views each, along with the corresponding ground truth disparity image, is freely available at https://www.mucar3.de/wacv2020-triple-sgm.

To assess Triple-SGM in real-world, we compare our triple stereo results with accumulated depth information from the LiDAR sensor Velodyne HDL-32E.

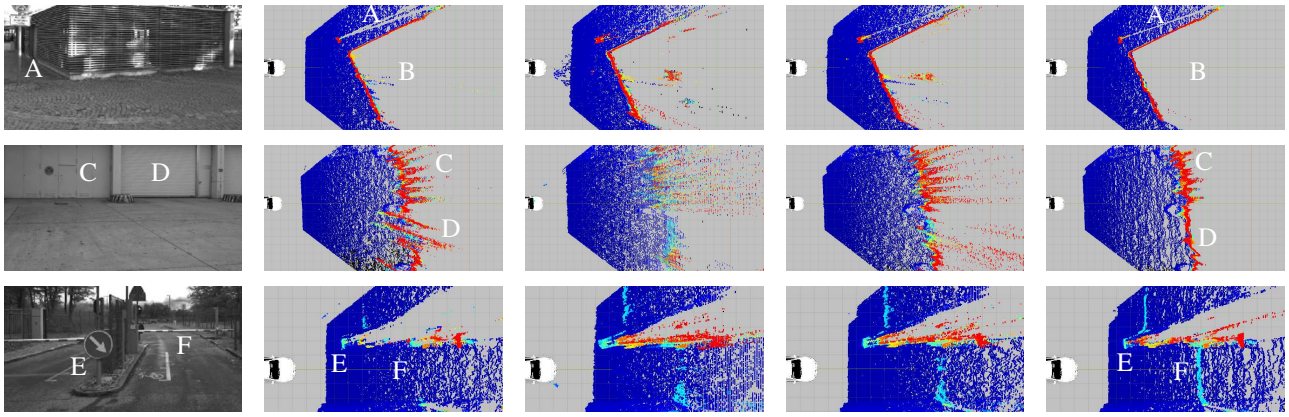
5.2. Qualitative evaluation

Experimental results obtained from real-world data can be seen in Fig. 6. The following observations can be made:

- The occlusion resulting from the large horizontal baseline is reduced by the combination with the vertical camera pair, *e.g.* (A).
- Artifacts are reduced significantly in case of cost fusion compared to classical stereo as well as disparity fusion, *e.g.* (B).
- Disparities on objects with low texture (C) and periodic structures (D) are significantly improved using cost fusion.
- Using cost fusion leads to better representation of object outlines, *e.g.* (E).
- The boom barrier (F) can not be detected using only a horizontal camera pair. Although being detected better using the vertical camera pair, the measurements are very inaccurate and still quite sparse, even if combined with the horizontal pair by disparity fusion. With the same input data, cost fusion detects the barrier completely and shows the most accurate measurements.

5.3. Quantitative evaluation

Evaluating our Triple-SGM on 2000 images of our synthetic dataset, we see a significant increase in pixels with correct disparity. Figure 7a shows, that a median of 88% of the image has a disparity error below $2\,\mathrm{px}$ compared to 82% using only a traditional horizontal stereo pair. The vertical pair performs worse due to its smaller baseline and the more frequent occurrence of vertical structures.



(a) Left camera image (b) Horizontal (c) Vertical (d) Disparity fusion [17] (e) Cost fusion (ours)

Figure 6: Qualitative results of real-world scenes shown in (a). The 3D point clouds shown in top-down view in (b) – (e) are derived from the different disparity images. The color of a point encodes its height with blue being near the ground and red being above a height of 2.2 m.

Max. disparity error	0.25 px	0.5 px	1 px	3 px
Horizontal	**51.6%**	70.8%	80.9%	85.2%
Vertical	27.0%	49.7%	73.3%	84.5%
Disparity fusion [17]	44.6%	69.4%	86.3%	91.7%
Cost fusion (ours)	46.0%	**71.1%**	**87.1%**	**92.0%**

Table 1: Percentage of correct pixels for different thresholds for the maximum disparity error.

It can be seen in Fig. 7b that the horizontal stereo provides a few more very accurate disparity pixels; but as of allowing an error in the disparity of about 0.5 px, Triple-SGM outperforms traditional stereo. And, as Table 1 shows, our triple stereo based on cost fusion surpasses the disparity fusion of [17] in all cases. The fact that the different methods shown in Fig. 7b do not converge to a common rate of correct pixels can be explained by the different sizes of the occluded areas. While the classic stereo methods always suffer from occluded image areas, the combination of horizontal and vertical stereo can largely compensate that and thus shows an ultimately higher percentage of valid pixels.

A selection of the evaluation results for scenes from our synthetic dataset as well as real-world images is given in Table 2. The worse results for the real-world data can result from an inaccurate localization of the camera within the accumulated point cloud. It shows that the combination of horizontal and vertical stereo pairs is beneficial for the detection of objects represented by higher correct pixel rates. The accuracy (represented by the RMSE) of the vertical stereo rig is worse than that of the horizontal one due to the smaller baseline. This also results in slightly higher RMSE values for the triple case compared to horizontal stereo. A higher vertical baseline would mitigate this issue. Further, the results show that our proposed cost-based fusion algorithm is superior to disparity fusion [17].

5.4. Timing

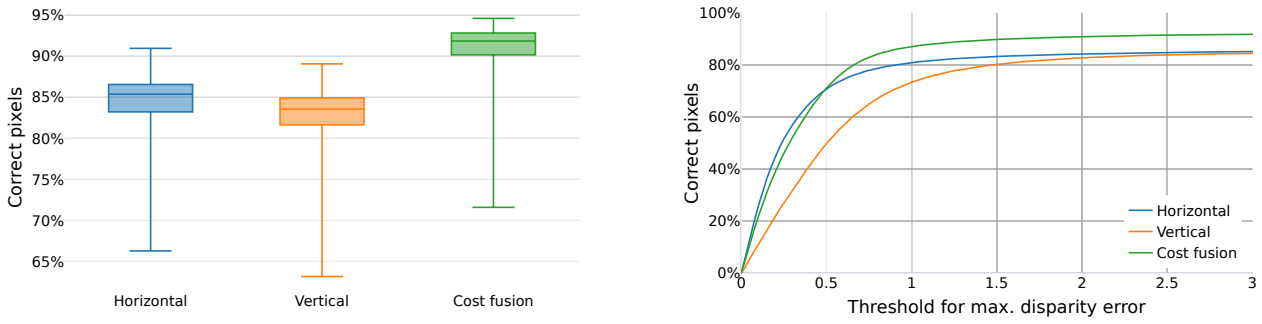
We tested the code on a Nvidia GeForce GTX 1050Ti with an input image size of 926 px × 276 px, averaged over 300 runs. Despite having to calculate the disparity of two stereo pairs instead of one, the new step of matching cost fusion takes only 7.2 ms. This results in a total runtime for our Triple-SGM algorithm of 25 ms compared to 17 ms for standard dual camera SGM. The most time-consuming part, the cost accumulation along the scanlines in SGM, remains unchanged with 14 ms. This is a major advantage over disparity fusion [17], where SGM has to be executed twice. Although each pixels' added cost can be calculated in parallel, there are overlapping memory accesses which slow the algorithm down. Therefore it is important to accumulate the costs before the actual scanning.

6. Conclusion

In this work, an extension of the SGM framework for the processing of image triplets is presented. The matching costs from a horizontal and a vertical stereo pair are fused by a simple weighted sum. This saves computation time because the cost aggregation of the original SGM algorithm has to be run only once and improves the overall results. We showed the effectiveness of the proposed fusion strategy using synthetic and real-world data. The synthetic dataset used for the evaluation is made publicly available in order to allow further comparison and development of triple stereo algorithms.

ACKNOWLEDGMENT

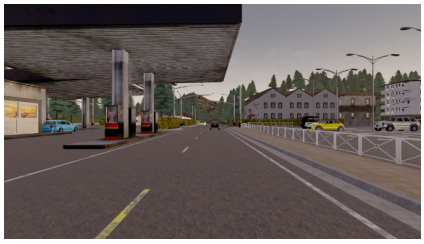
The authors gratefully acknowledge funding by the Federal Office of Bundeswehr Equipment, Information Technology and In-Service Support (BAAINBw).



(a) (b)

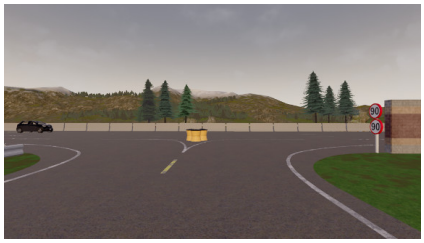
Figure 7: (a) Percentage of correct pixels when considering pixels with disparity errors under 2 px as correct. (b) Development of the correct pixels with varying threshold for the maximum disparity error.

Horizontal	71.2%	**0.51 px**
Vertical	73.0%	0.89 px
Disparity fusion [17]	81.3%	0.55 px
Cost fusion (ours)	**83.3%**	0.53 px



Horizontal	71.3%	**0.54 px**
Vertical	69.1%	0.74 px
Disparity fusion [17]	79.0%	0.54 px
Cost fusion (ours)	**83.7%**	**0.54 px**

Horizontal	70.0%	**0.47 px**
Vertical	72.5%	0.65 px
Disparity fusion [17]	82.4%	0.54 px
Cost fusion (ours)	**83.0%**	0.54 px



Horizontal	82.3%	**0.44 px**
Vertical	82.1%	0.66 px
Disparity fusion [17]	91.1%	0.47 px
Cost fusion (ours)	**91.7%**	0.45 px

Horizontal	83.2%	0.41 px
Vertical	85.1%	0.70 px
Disparity fusion [17]	91.5%	0.42 px
Cost fusion (ours)	**92.9%**	**0.40 px**

Horizontal	70.9%	**0.38 px**
Vertical	71.4%	0.62 px
Disparity fusion [17]	81.1%	0.54 px
Cost fusion (ours)	**83.7%**	0.50 px

Horizontal	73.0%	0.99 px
Vertical	65.5%	1.00 px
Disparity fusion [17]	82.5%	**0.73 px**
Cost fusion (ours)	**84.2%**	0.83 px

Horizontal	66.5%	1.03 px
Vertical	36.8%	1.10 px
Disparity fusion [17]	62.4%	1.06 px
Cost fusion (ours)	**79.3%**	**1.01 px**

Horizontal	65.4%	**0.74 px**
Vertical	49.0%	1.13 px
Disparity fusion [17]	74.0%	0.84 px
Cost fusion (ours)	**77.3%**	**0.74 px**

Table 2: Quantitative results for example scenes from our synthetic dataset as well as real-world images. The first column under each image represents the percentage of pixels with a disparity error below 2 px. The second column gives the root-mean-square error (RMSE) of all those pixels.

References

[1] Robust Vision Challenge. http://www.robustvision.net/, 2018. 2

[2] Y. K. Baik, J. Choi, and K. M. Lee. An Efficient Trinocular Rectification Method for Stereo Vision. *Proceedings of Frontiers of Computer Vision (FCV)*, 2007. 2, 3

[3] C. Banz, H. Blume, and P. Pirsch. Real-Time Semi-Global Matching Disparity Estimation on the GPU. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 514–521, Nov. 2011. 2

[4] C. Banz, P. Pirsch, and H. Blume. Evaluation of Penalty Functions for Semi-Global Matching Cost Aggregation. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS*, pages 1–6, July 2012. 4

[5] Z. Chen, X. Sun, L. Wang, Y. Yu, and C. Huang. A Deep Visual Correspondence Embedding Model for Stereo Matching Costs. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, page 972–980, Dec. 2015. 2

[6] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, page 1–16, 2017. 6

[7] I. Ernst and H. Hirschmüller. Mutual Information Based Semi-Global Stereo Matching on the GPU. In *International Symposium on Visual Computing (ISVC)*, 2008. 2

[8] S. K. Gehrig, F. Eberli, and T. Meyer. A Real-Time Low-Power Stereo Vision Engine Using Semi-Global Matching. In M. Fritz, B. Schiele, and J. H. Piater, editors, *Computer Vision Systems*, pages 134–143. Springer Berlin Heidelberg, 2009. 2

[9] D. Hafner, O. Demetz, and J. Weickert. Why Is the Census Transform Good for Robust Optic Flow Computation? In A. Kuijper, K. Bredies, T. Pock, and H. Bischof, editors, *Scale Space and Variational Methods in Computer Vision*, pages 210–221. Springer Berlin Heidelberg, 2013. 4

[10] I. Haller and S. Nedevschi. GPU optimization of the SGM stereo algorithm. In *Proceedings of IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 197–202, Aug. 2010. 2

[11] A. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2006. 3

[12] M. Heinrichs and V. Rodehorst. Trinocular Rectification for Various Camera setups. In *Symposium of ISPRS Commission III-Photogrammetric Computer Vision PCV*, volume 6, page 43–48, 2006. 2

[13] M. Heinrichs, V. Rodehorst, and O. Hellwich. Efficient Semi-Global Matching for Trinocular Stereo. *Proceedings of Photogrammetric Image Analysis*, Jan. 2007. 2

[14] D. Hernandez-Juarez, A. Chacón, A. Espinosa, D. Vázquez, J. Moure, and A. López. Embedded Real-time Stereo Estimation via Semi-global Matching on the GPU. *Procedia Computer Science*, 80:143–153, 2016. International Conference on Computational Science (ICCS). 2, 6

[15] H. Hirschmueller. Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 1, 2, 3, 4, 6

[16] H. Hirschmüller. Semi-Global Matching: Motivation, Development and Applications, Sept. 2011. 6

[17] J. Kallwies and H.-J. Wuensche. Effective Combination of Vertical and Horizontal Stereo Vision. In *Proceedings of IEEE Winter Conference on Applications of Computer Vision (WACV)*, Lake Tahoe, NV/CA, USA, Mar. 2018. 1, 2, 6, 7, 8

[18] M. Maitre, Y. Shinagawa, and M. N. Do. Symmetric Multi-View Stereo Reconstruction from Planar Camera Arrays. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 2

[19] M. Michael, J. Salmen, J. Stallkamp, and M. Schlipsing. Real-time stereo vision: Optimizing Semi-Global Matching. In *Proceedings of IEEE Intelligent Vehicles Symposium (IV)*, pages 1197–1202, June 2013. 2

[20] J. Mulligan, V. Isler, and K. Daniilidis. Trinocular Stereo: a Real-Time Algorithm and its Evaluation. *International Journal of Computer Vision (IJCV)*, 47(1-3):51–61, 1 2002. 2

[21] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling. High-resolution Stereo Datasets with Subpixel-Accurate Ground Truth. In *Proceedings of German Conference on Pattern Recognition (GCPR)*, page 31–42. Springer, 2014. 2

[22] A. Seki and M. Pollefeys. SGM-Nets: Semi-Global Matching with Neural Networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 6640–6649, July 2017. 2

[23] P. Viola and W. M. Wells. Alignment by Maximization of Mutual Information. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, page 16–23, June 1995. 4

[24] R. Zabih and J. Woodfill. Non-parametric Local Transforms for Computing Visual Correspondence. In J.-O. Eklundh, editor, *Proceedings of European Conference on Computer Vision (ECCV)*, page 151–158. Springer Berlin Heidelberg, 1994. 4

[25] J. Zbontar and Y. LeCun. Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. *Journal of Machine Learning Research*, 17:1–32, 2016. 2