# The Naked Truth about Cost Functions for Stereo Matching

Simon Hermann and Reinhard Klette

The *.enpeda..* Project, The University of Auckland
Auckland, New Zealand

**Abstract.** This paper reports about the performance of various cost functions as common or possible for stereo matching, within a 'purist's matching strategy', which does not integrate any further optimization approach such as a continuity term, coarse-to-fine, left/right consistency checks, or others. The performed experiments support a few general conclusions, such as about the relation between 1D- or 2D-window based cost functions, the use of zero-mean normalization, the census cost function, or the use of B-splines for subpixel accurate cost calculation.

## 1 Introduction

Stereo algorithms are currently becoming an integral part of vision-based driver assistance systems (DAS); see, for example, [4]. Stereo algorithms using a semi-global matching (SGM) strategy are of particular interests; the paper [6] initiated this interest in SGM algorithms. As in other stereo matching methods, a similarity measure (i.e., a cost function) determines the cost at every pixel for potentially relevant disparities. In SGM algorithms, these costs are then integrated by applying a semi-global accumulation scheme that incorporates smoothness terms and uses a dynamic programming technique for efficiency reasons. In belief propagation [3] stereo matching algorithms, a data term (i.e., a cost function) is combined with a continuity term.

A recent paper [8] evaluates the performance of matching costs on images with radiometric differences. This work is especially interesting for real-world stereo sequences as provided, for example, on EISATS – the *.enpeda..* image sequence analysis test site.[1] Differences in image brightness (say, between left and right camera at time $t$) are very common for stereo vision in DAS.

One of the core assumptions for the evaluation in [8] is that *"the performance of a matching cost can depend on the algorithm that uses the cost".* To assure that their results are not biased by using one particular stereo algorithm only, they tested most of the listed cost functions within representative algorithms for local, semi-global (SGM), or global matching strategies (one algorithm for each class).

Since cost calculation is the core component of a stereo matching algorithm, we have chosen for this paper a methodology that evaluates cost functions

[1] www.mi.auckland.ac.nz/EISATS

regardless (!) of the applied stereo algorithm. This may help to adjust other components of a stereo matching algorithm (e.g., number of search paths, the smoothness or continuity term) with respect to the preferred cost function.

In this sense this paper discusses the 'naked truth' about cost functions: we do not integrate those into any non-trivial stereo matching approach; there are no coarse-to-fine, smoothing, left/right consistency checks, or other means not directly specifying the cost function.

Section 2 specifies all cost functions considered in this paper. Typically we assume window-based cost functions which are usually defined by a square $k \times k$ neighborhood. Section 2 describes commonly known cost functions for such a 2D default window. Since our focus is not only on matching accuracy but also on computational time efficiency, we evaluate all those functions also over a 1D window $1 \times l$ along the image row (i.e., along the epipolar line). It turned out that a 1D $1 \times l$ window may (under some conditions) qualitatively outperform their 2D $k \times k$ equivalent, typically with $l > k$, but $l < k^2$, thus saving computation time and (!) improving accuracy.

Section 3 proposes approximating B-spline curves for cost computations. Although B-splines are, of course, very expensive in terms of computation time, we gain in return a continuous representation of the intensity signal. This provides potentials for the application of subpixel (or subsampling) techniques that can be of benefit for offline applications.

Section 4 discusses the evaluation methodology.[2] Experimental results are provided in Section 5. Section 6 concludes this paper.

Our motivation for the reported research is to adjust the components of an SGM algorithm for real-world stereo sequences (such as in vision-based driver assistance), allowing real-time processing (say, at $25 \ldots 30$ Hz).

## 2 Cost Functions

Cost functions (also called similarity measures) define the "core" of many stereo matching algorithms.

### 2.1 General Notation

In a (rectified) stereo image pair we consider a *base* and a *match image*. The base image is assumed to be the right image $R$. $L$ denotes the match image. We only consider gray-level (i.e., intensity) images in this paper with values between 0 and $G_{max}$.

Any local *cost function* $C$ defines a global mapping $\Gamma_C(R, L) = C$ that takes rectified stereo images $R$ and $L$ as input, and outputs a 3D *cost* matrix $C$ with

[2] We provide online a compiled command line application (costSGM.exe under win32) at www.mi.auckland.ac.nz/costSGM that allows one to compute a disparity map for a submitted rectified stereo image pair, selecting a cost function as evaluated in this paper. The applied stereo matching algorithm is the SGM algorithm as proposed in [6]. This application is documented by a brief online manual.

elements $C(i,j,d)$, representing the cost when matching a pixel at $(i,j)$ in $R$ with a pixel at $(i+d,j)$ in $L$, for any relevant disparity $d$.

We may simplify notations due to working with rectified images where epipolar lines are aligned to the $x$-axis, and we may consider a fixed image row $j$ in the base and match image at a time. Let $p_i$ denote a pixel location in $R$ at image column $i$. Let $R_i$ be the value at this location in the base image; $q_{i+d}$ denotes the pixel location $(i+d,j)$ in the match image $L$ with intensity $L_{i+d}$. $C(i,d)$ is the abbreviated notation for the cost.

We distinguish between *window-based* and *pixel-based* cost functions. Pixel-based cost functions only depend on values $R_i$ and $L_{i+d}$. Window-based cost functions take more intensity values into account, such as in $k \times k$ or $1 \times l$ neighborhoods of $p_i$ and $q_{i+d}$, for $k, l \geq 2$. Our experiments use odd integers $k$ and $l$.

Window-based cost functions $C$, as used in this paper are either 1D (i.e., defined on an $1 \times l$ window) or defined on a $k \times k$ window (without calling it 2D explicitly in this case).

We evaluate window-based cost functions either in original or zero-mean versions. In the *zero-mean version*, first the mean intensity of the window is computed and then subtracted from all the intensities in the window prior to applying the original version of the cost function. The zero-mean version of a cost function $C$ is identified by adding a prefix Z to its acronym. (Any acronym of an original version will not start with a Z.)

### 2.2 Pixel-Based Cost Functions

The *absolute difference* (AD) of base and match pixel is the simplest measure:

$$C_{AD}(i,d) = |R_i - L_{i+d}|$$

Another commonly used pixel-based cost function (BT) was presented in [1]. In a first step, intensities in $R$ and $L$ are interpolated using either a previous or a subsequent pixel along the epipolar line. For example,

$$R_{i-1/2} = \frac{1}{2}\,(\,R_i\;+\;R_{i-1}\,)$$

is an interpolated value at $p_i$, with respect to the previous pixel.

$$\mathcal{R}_i = \{R_{i-1/2}, R_i, R_{i+1/2}\}$$

is a set containing the intensity at $p_i$ in $R$ as well as the interpolated intensities with previous and subsequent pixels. Analogously, $\mathcal{L}_{i+d}$ is a set containing the intensity at $q_{i+d}$ in $L$ as well as the interpolated intensities with previous and subsequent pixels. The BT cost function is then defined as follows:

$$\begin{aligned} C_{BT}(i,d) &= \min\{a,b\} \text{ with} \\ a &= \max\{R_i - \max(\mathcal{L}_{i+d}),\ \min(\mathcal{L}_{i+d}) - R_i,\ 0\} \\ b &= \max\{L_{i+d} - \max(\mathcal{R}_i),\ \min(\mathcal{R}_i) - L_{i+d},\ 0\} \end{aligned}$$

### 2.3 Window-Based Cost Functions

The *sum of absolute differences* (SAD) cost function is a straightforward window extension of the AD cost function. For specifying the ZSAD cost function, let $W$ denote the set of all $l$ or $k^2$ pixel locations of the used window when centered at reference point $(0,0)$. Let $\overline{R}_i$ be the mean of all intensities at pixels in $W + p_i$ in $R$, and $\overline{L}_{i+d}$ be the mean of all intensities at pixels in $W + q_{i+d}$ in $L$. The ZSAD cost function is defined as follows:

$$C_{ZSAD}(i,d) = \sum_{(x,y)\in W+p_i} \left| [R_{xy} - \overline{R}_i] - [L_{x+d,y} - \overline{L}_{i+d}] \right|$$

For deriving the original $C_{SAD}$ from this formula, both means are set to be zero. A similar measure is the *sum of squared differences* (SSD). The ZSSD cost function is defined as follows:

$$C_{ZSSD}(i,d) = \sum_{(x,y)\in W+p_i} \left| [R_{xy} - \overline{R}_i]^2 - [L_{x+d,y} - \overline{L}_{i+d}]^2 \right|$$

For the original $C_{SSD}$, just set both means to zero.

Similarly, we specify the *normalized cross correlation* (NCC) function via the ZNCC version:

$$C_{ZNCC}(i,d) = 1.0 - \frac{\sum_{(x,y)\in W+p_i} \left[R_{xy} - \overline{R}_i\right] \left[L_{x+d,y} - \overline{L}_{i+d}\right]}{\sqrt{\sum_{(x,y)\in W+p_i} \left[R_{xy} - \overline{R}_i\right]^2 \sum_{(x,y)\in W+p_i} \left[L_{x+d,y} - \overline{L}_{i+d}\right]^2}}$$

$C_{NCC}$ follows for $\overline{R}_i = \overline{L}_{i+d} = 0$.

The *census* cost function is a very robust measure and we use it based on the following definition:

$$C_{Zcensus}(i,d) = \sum_{(x,y)\in W+p_i} \rho(x,y,d) \text{ with}$$

$$\rho(x,y,d) = \begin{cases} 0 & R_{xy} \triangle \overline{R}_i \text{ and } L_{x+d,y} \triangle \overline{L}_{i+d}, \text{ with } \triangle \text{ either } < \text{ or } > \\ 1 & \text{otherwise} \end{cases}$$

If we set $\overline{R}_i = R_i$ and $\overline{L}_{i+d} = L_{i+d}$ in this formula, then we have the original census cost function $C_{census}$.

## 3 B-Splines for Cost Calculation

For B-spline curves we refer to [9] for a practical introduction, and to [2] for a broader and more general coverage. We briefly recall B-spline curves here solely for their use for improving the performance of cost calculations.

### 3.1 B-Spline Curves

A parametric 2D B-spline curve of degree $n$ is defined by a sequence of control points $d_0, d_1, ..., d_{m-1}$ in $\mathbb{R}^2$ and a sequence of $m+n+1$ knots $u_0, u_1, ..., u_{m+n}$, with $u_i \leq u_{i+1}$; $[u_0, u_{m+n}) \in \mathbb{R}$ is the domain on which the curve is defined. To evaluate a point on the curve we have a mapping $\int : [u_0, u_{m+n}) \longrightarrow \mathbb{R}^2$, with $\int(t) = (x(t), y(t))$ and $t \in [u_0, u_{m+n})$. To evaluate $x(t)$ and $y(t)$, the following interpolation scheme is applied for the control points:

$$\int(t) := \sum_{i=0}^{m-1} \mathcal{N}_i^n(t) d_i$$

The basis functions $\mathcal{N}_i^n$ are recursively defined by the knot sequence as follows:

$$\mathcal{N}_i^0(t) = \begin{cases} 1 & t_i \leq t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{N}_i^k(t) = \frac{t - t_i}{t_{i+k} - t_i} \mathcal{N}_i^{k-1}(t) + \frac{t_{i+k+1} - t}{t_{i+k+1} - t_{i+1}} \mathcal{N}_{i+1}^{k-1}(t)$$

with $i = 0, ..., m+n+1$ and $k = 1, ..., n$. Obviously, the basis functions depend on the knot sequence, thus the choice of the knot vector affects the shape of the curve. For our cost calculations we choose an open uniform B-spline curve, where the knot sequence has multiplicities of $n+1$ at the beginning and end, and is equidistantly spaced between 0 and 1.

We set the degree to be $n = 3$. The knot sequence in case of, for example, seven control points would be $0, 0, 0, 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, 1, 1, 1$. The multiplicity of knots at the beginning and end ensures that the B-spline curves start at $d_0$ and end at $d_{m-1}$. Because we choose a cubic curve, we can also derive first and second order

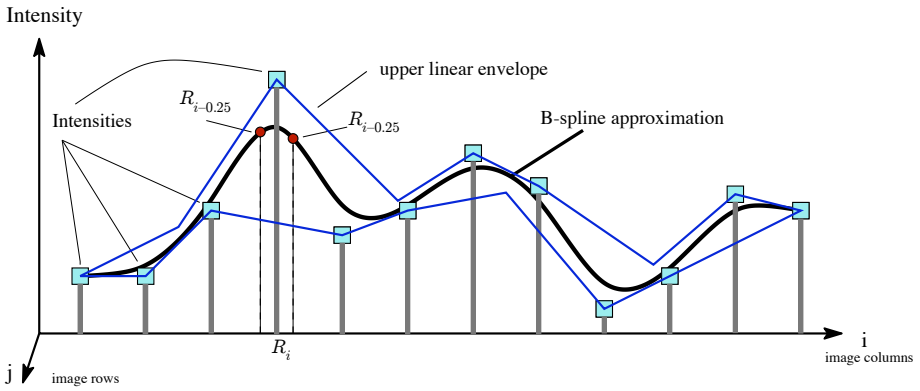


**Fig. 1.** A B-spline curve approximates a fixed image row with index $j$, using positions $i$ and corresponding intensities as a control polygon. For an intensity $R_i$, interpolates $R_{i-0.25}$ and $R_{i+0.25}$ are used for cost calculation. Note the smoothing effect at intensity amplitude $R_i$.

information at any curve point by formal differentiation of the basis functions. This might be of benefit if higher order information (such as curve normal orientation, or curvature) should also be considered in the similarity measure.

### 3.2 Approximation of a Single Image Row

B-splines are widely used for data approximation. We approximate a single image row by a 2D open uniform cubic B-spline curve. The row index $j$ is fixed, and does not need to be used. We identify the first coordinate of the control points with the column index $i$, and the second coordinate with the intensity $R_i$, for each pixel location in image row $j$. Thus, the control sequence has the form $d_i = (i, R_i)$, for $i = 0, ..., w-1$, where $w$ is the uniform width of $R$ and $L$.

We recall the following *convex hull property* of approximating B-splines: any point on the curve lies within the union of convex hulls, each defined by $n$ consecutive points $d_i, d_{i+1}, ..., d_{i+n-1}$ of the control sequence, with $i = 0, ..., m-n-1$ and $n$ being the degree of the curve. In other words, it is assured that the curve follows smoothly the sequence of discrete control points. For example, see Figure 1. The whole B-spline lies within the upper and lower linear envelope. Also note that, due to the approximating character of a B-spline, this creates a smoothing effect for the intensity signal. This effect occurs especially at outliers, such as $R_i$ in Figure 1.

### 3.3 Application to Cost Calculation

Consider the 1D SAD cost function which sums up intensity differences using an $1 \times l$ window, for pixel locations in some image row $j$ in $R$ and $L$. This is possible because we have direct access to intensity values $R_i$ or $L_{i+d}$ at grid points. However, after approximating such an image row by a B-spline curve, there is no explicit representation of the form $R_i = f(p_i)$, because B-splines are parametric curves, and multiple-valued functions in general.[3]

However, the used control polygon as specified follows the image row (i.e., the first component represents image column indices, and those are strictly monotonically increasing). It follows that the underlying curve behaves like a single-valued function. Thus we are able to apply a binary search scheme to read the intensity at an arbitrary column index within a predefined tolerance $\delta > 0$.

We utilize the described B-spline approximation for the calculation of cost functions 1D NCC, 1D ZNCC, 1D SAD and 1D ZSAD, as well as 1D SSD and 1D ZSSD, simply by replacing intensities $R_i$ by $R_{i-0.25}$ and $R_{i+0.25}$ using $\delta = 0.01$, (see Figure 1). In other words, we subsample the intensity signal and evaluate this way (within the same window) twice as many intensity values by using a step width of half a pixel. If using the intensity value $R_i$ at $i$ on the B-spline with a step width of 1, the B-spline would simply serve as a smoothing function. We used this sampling later in Section 5.6 to have a fair comparison in the evaluation of the subpixel accuracy approach.

[3] Note that the original grid points are encoded in the control polygon, and can easily be retrieved.

## 4 Evaluation of Cost Functions

We compute costs at every pixel location (except for a seven pixel offset from the border), and for any possible disparity.

### 4.1 The Purist's Matching Approach

Recall that the right input image acts as base image. For every pixel location $p_i$ we select that disparity (for the generated depth map) which has the overall minimum cost. If there is more than one minima, the result is ambiguous, and we consider the disparity at this pixel location to be invalid (i.e., undefined). Once a valid minimum is found, we fit a parabolic curve through costs of neighboring disparities to achieve subpixel accuracy; we then check this disparity against the ground truth; if this disparity does not deviate by more than 1.5 we call this a *perfect match*. Otherwise we call it a *mismatch*.

In our evaluation we computed the percentage of perfect matches and mismatches. Those are computed by dividing the number of perfect matches (mismatches) by the number of evaluated pixels, and this includes all pixels for which ground truth was available. Thus, if ambiguities occur, then those percentages do not sum up to 100%.

We also computed a *mean ambiguity* by dividing all minima by the number of evaluated pixels. Thus, a mean ambiguity of 1.0 means that there has been always exactly one minimum at each pixel location. A mean ambiguity of $>$ 1.0 means that there are pixel locations with multiple minima. The experiments allow one to conclude that most of the cost functions do have a very low mean ambiguity (when working with floating point numbers). The only exceptions are given by the pixel based cost functions as well as by the census cost function. (Thus we include results for mean ambiguity only in Tables 4 and 5.)

Note that we did not swap the roles of $R$ and $L$ of being base and match image in this order (to perform a right/left consistency check). This is because we consider such a check as being a part of a stereo computation, and this would alter the results of a pure cost function analysis.

### 4.2 Test Data

We evaluate stereo matching on rectified image pairs as provided on the Middlebury stereo website,[4] for which ground truth is available. We use the sequences *Tsukuba* (at a sixteenth of its size), *Teddy*, *Cones* (at a quarter of its size) as well as *Art*, *Books*, *Laundry*, *Dolls*, *Moebius*, and *Reindeer* (at a third of its size).

We decided to take those nine image pairs, provided with ground truth, to ensure diversity. Downsized versions are taken because many evaluations (such as in [8]) use them as well, and we have thus a comparable setup.

[4] vision.middlebury.edu/stereo/

In case of *Tsukuba* we did not accept any deviation for identifying a perfect match. We also did not aim at subpixel accuracy in this case because ground truth values are multiples of sixteen.

Note that allowing a deviation of 1.5 is just a logical consequence from rounding to the closest disparity value when multiplying by the factor used for downscaling the images, and allowing the common deviation of 1 at full resolution.

We applied our test methodology to all the image sets listed above, for all the specified cost functions, using window sizes $k \times k$ and $1 \times l$, for $k$ and $l$ being either 7 or 15.

## 5 Results

We report about those results first which appeared to us as being most interesting.

### 5.1 1D versus 2D; Does Size Matter?

Table 1 presents resulting mean percentages of perfect matches for studied cost functions, summarized over all (!) of the evaluated images pairs. The focus lies here on comparing a 1D against a 2D window, and also a larger 2D window versus a smaller 2D window. (For results on individual image pairs, see Tables 2 to 5 later on.)

**Table 1.** Mean of perfect matches for (top) SAD, (middle) SSD, and (bottom) NCC, comparing the performance of best performing 1D cost functions versus their corresponding 2D windows.

| *Window* | $l$=15 | | $k$=15 | | $k$=7 | |
|---|---|---|---|---|---|---|
| cost function | 1D SAD | 1D ZSAD | SAD | ZSAD | SAD | ZSAD |
| *mean* | 49.7% | 58.1% | 56.8% | 60.6% | 53.4% | 63.5% |

| *Window* | $l$=15 | | $k$=15 | | $k$=7 | |
|---|---|---|---|---|---|---|
| cost function | 1D SSD | 1D ZSSD | SSD | ZSSD | SSD | ZSSD |
| *mean* | 51.8% | 57.8% | 55.7% | 58.3% | 55.3% | 63.1% |

| *Window* | $l$=15 | | $k$=15 | | $k$=7 | |
|---|---|---|---|---|---|---|
| cost function | 1D NCC | 1D ZNCC | NCC | ZNCC | NCC | ZNCC |
| *mean* | 58.0% | 56.7% | 58.5% | 59.2% | 63.2% | 63.2% |

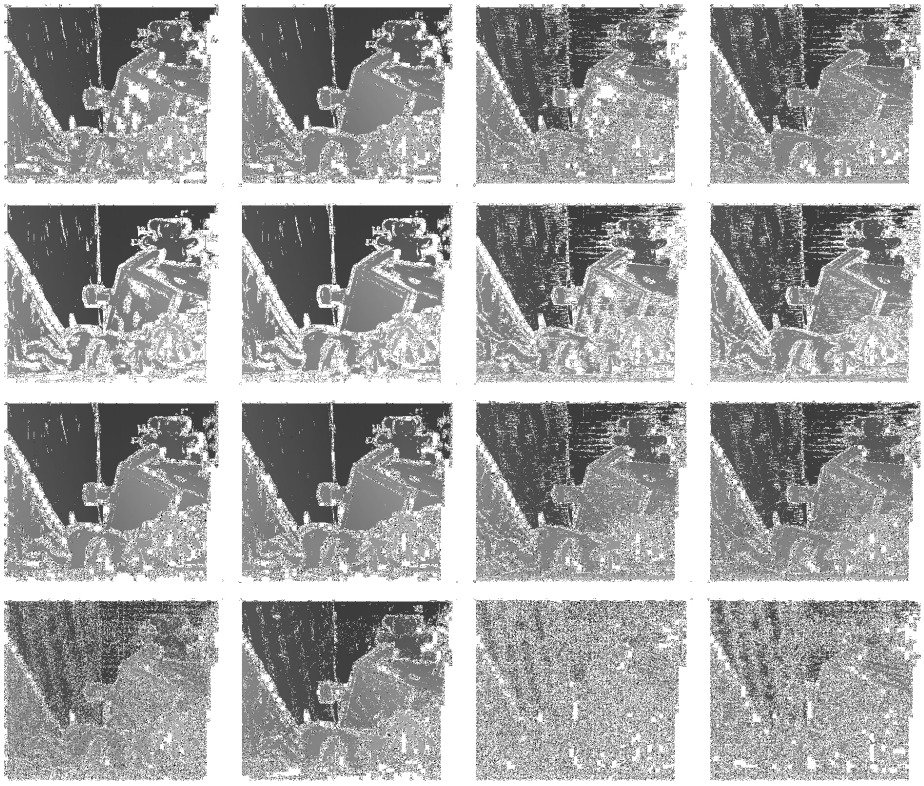

**Fig. 2.** Teddy image pair. From left to right. First Row: SAD, ZSAD, 1D SAD, 1D ZSAD. Second Row: SSD, ZSSD, 1D SSD, 1D ZSSD. Third Row: NCC, ZNCC, 1D NCC, 1D ZNCC. Fourth Row: Census, ZCensus, 1D Census, 1D ZCensus

Figure 2 shows results for a selection of cost functions for the *Teddy* image pair. Mismatches are labeled white.

We focus on the relation between accuracy and computational time. We summarize our evaluation as follows:

- SAD and SSD perform better in their zero-mean versions whereas ZNNC and NNC tend to lead to identical results. Thus, there appears to be no reason to run the more expensive zero-mean version for NCC.
- Choosing a window $k = 7$ performs as good as $k = 15$, and in some cases even better (e.g., NCC). Because of the complexity contribution $k^2$ to the run time, this is certainly an important observation.
- In case of NCC, the 1D version is as good as its 2D version if using $k = l = 15$. Thus, it appears that here is no reason to run the much more expensive 2D version for NCC.
- The 1D regular and zero-mean versions for SAD and SSD ($l = 15$) are about 3-4 % points below of the 2D ($k = 7$) versions, and about 10 % points below if

considering 1D original versus 2D zero mean. Here is a clear tradeoff between performance and time efficiency. (See the quality differences in Figure 2).
– 1D zero-mean ($l = 15$) qualitatively outperforms the original 2D version for $k = 7$. However, when looking at computational time, it probably does not make a big difference in terms of computational time!

**Table 2.** Mean results of different kinds of SAD cost functions for different image pairs, with window size equals 15: the given percentages denote perfect matches. For $SAD_{50}$ and $ZSAD_{50}$, see Section 5.4.

| ***Window*** $k$=15 / $l$=15 | SAD | $SAD_{50}$ | ZSAD | $ZSAD_{50}$ | 1D SAD | 1D ZSAD | B-spline$^{SAD}$ | ZB-spline$^{SAD}$ |
|---|---|---|---|---|---|---|---|---|
| TSUKUBA | 73.3% | 74.1% | 72.5% | 73.6% | 65.0% | 63.7% | 71.3% | 70.4% |
| CONES | 66.2% | 67.3% | 69.7% | 69.3% | 55.5% | 66.6% | 49.3% | 65.3% |
| TEDDY | 62.3% | 64.1% | 67.1% | 66.8% | 55.1% | 63.7% | 52.4% | 63.0% |
| ART | 41.4% | 43.3% | 47.0% | 47.7% | 35.4% | 47.8% | 33.2% | 46.4% |
| BOOKS | 58.3% | 60.5% | 61.3% | 61.4% | 43.8% | 56.4% | 41.4% | 56.6% |
| LAUNDRY | 43.4% | 47.2% | 51.1% | 51.1% | 37.6% | 46.8% | 38.1% | 48.4% |
| MOEBIUS | 60.7% | 62.3% | 64.0% | 63.9% | 55.4% | 63.0% | 52.5% | 62.6% |
| DOLLS | 55.2% | 55.7% | 58.6% | 58.2% | 52.3% | 59.7% | 50.6% | 58.6% |
| REINDEER | 50.5% | 51.8% | 54.3% | 53.9% | 47.6% | 55.0% | 47.0% | 55.5% |
| *mean* | 56.8% | 58.5% | 60.6% | 60.7% | 49.7% | 58.1% | 48.4% | 58.5% |

**Table 3.** Mean results of different kinds of SSD cost functions for different image pairs with window size equals 15. For $SAD_{50}$ and $ZSAD_{50}$, see Section 5.4.

| *Window* $k$=15 / $l$=15 | SSD | $SSD_{50}$ | ZSSD | $ZSSD_{50}$ | 1D SSD | 1D ZSSD | B-spline$^{SSD}$ | ZB-spline$^{SSD}$ |
|---|---|---|---|---|---|---|---|---|
| TSUKUBA | 73.5% | 73.6% | 73.5% | 73,7% | 65.7% | 65.4% | 70.0% | 70.4% |
| CONES | 64.9% | 65.0% | 66.3% | 66.0% | 58.6% | 65.2% | 53.4% | 64.2% |
| TEDDY | 61.0% | 61.8% | 64.2% | 63.9% | 57.4% | 63.5% | 54.0% | 62.7% |
| ART | 37.8% | 39.2% | 43.7% | 43.6% | 36.6% | 47.0% | 33.7% | 45.7% |
| BOOKS | 59.3% | 59.5% | 60.5% | 60.3% | 48.0% | 56.5% | 45.1% | 56.8% |
| LAUNDRY | 44.9% | 46.6% | 49.8% | 49.4% | 40.2% | 47.2% | 40.5% | 48.5% |
| MOEBIUS | 58.9% | 59.4% | 61.2% | 60.8% | 57.3% | 62.3% | 54.6% | 61.9% |
| DOLLS | 51.6% | 51.5% | 54.2% | 53.8% | 52.6% | 58.4% | 50.7% | 50.7% |
| REINDEER | 49.0% | 49.6% | 51.6% | 51.3% | 49.8% | 55.1% | 47.9% | 55.1% |
| *mean* | 55.7% | 56.2% | 58.3% | 58.1% | 51.8% | 57.8% | 50.0% | 57.3% |

### 5.2 Census Cost Function

The census cost function performs significantly better in its zero-mean version. Because the Hamming distance is used for the discrete cost calculation, the 1D version performs poor as expected, and with ambiguities. This is the reason that we excluded this cost function from being listed in the tables. However, an example of the poor performance of the 1D census function can be seen in

**Table 4.** Results of different kinds of NCC cost functions for different image pairs with window size equals 15: the given percentages denote perfect matches. Note that this table also covers the 2D census function as well as as AD and BT.

| *Window* $k$=15 / $l$=15 | NCC | ZNCC | 1D NCC | 1D ZNCC | B-spline$^{NCC}$ | ZB-spline$^{NCC}$ | Census | ZCensus | AD | BT |
|---|---|---|---|---|---|---|---|---|---|---|
| TSUKUBA | 73.9% | 73.9% | 65.8% | 64.5% | 70.6% | 69.5% | 54.5% | 63.9% | 15.9% | 3.4% |
| CONES | 66.2% | 67.4% | 65.0% | 64.8% | 64.0% | 63.9% | 60.6% | 66.7% | 4.9% | 0.7% |
| TEDDY | 64.2% | 65.5% | 63.3% | 62.8% | 62.6% | 62.4% | 52.7% | 62.5% | 5.3% | 0.7% |
| ART | 44.1% | 45.3% | 47.3% | 46.3% | 45.9% | 45.3% | 43.5% | 41.6% | 4.8% | 0.5% |
| BOOKS | 60.6% | 61.4% | 56.6% | 54.7% | 57.0% | 55.9% | 46.5% | 53.4% | 3.8% | 0.5% |
| LAUNDRY | 50.2% | 50.0% | 47.5% | 45.8% | 48.7% | 47.2% | 37.2% | 40.2% | 3.9% | 0.5% |
| MOEBIUS | 61.4% | 62.2% | 62.5% | 60.3% | 62.0% | 60.2% | 55.7% | 58.1% | 5.7% | 0.5% |
| DOLLS | 54.5% | 54.7% | 58.8% | 57.6% | 57.9% | 57.2% | 55.5% | 53.2% | 6.8% | 0.7% |
| REINDEER | 51.8% | 52.0% | 55.0% | 53.5% | 55.1% | 53.9% | 46.4% | 49.8% | 4.7% | 0.7% |
| *mean* | | | | | | | | | | |
| perfect matches | 58.5% | 59.2% | 58.0% | 56.7% | 58.2% | 57.3% | 49.8% | 54.4% | 6.2% | 0.9% |
| mean ambiguity | 1.00 | 1 | 1.01 | 1.02 | 1.00 | 1 | 1.13 | 1.42 | 2.65 | 9.29 |

Figure 2. When looking at results as in Figure 2, we concluded that the census function seems to be very robust in general, but SAD and NCC clearly indicate discontinuities (which may actually be usable features for object detection!).

### 5.3 Pixel-Based Cost Functions

Table 4 contains the evaluation of both pixel-based cost functions AD and BT. Both functions perform poorly when evaluated in our methodology. Obviously, the advantage of AD is that it is very fast (also compared to BT). Because BT needs some expensive interpolations and many comparisons it takes up more computation time than 1D SAD, which clearly performs better than BT. BT is also highly ambiguous. The mean ambiguity ranges from 6 minima per pixel up to 12 minima. The mean ambiguity of AD lies just between 2 to 3 minima per pixel. This does not necessarily mean that both perform poorly when used within a stereo matching strategy. But is just points out that these uncertainties should be considered in a stereo matching technique utilizing those.

### 5.4 Tuning of Cost Functions

We also apply a modification of the SAD and SSD cost functions. We only consider a certain percentage of the larger differences in the given window as possible contributors for the given sum. The reasoning is as follows: a match should have relatively small differences within the whole neighborhood; discarding a certain percentage of the smaller differences in each neighborhood boosts the impact of larger differences, and this may help to decide whether a pixel is a match or not. In our evaluation we choose to discard the lower 50% differences, leading to cost functions $\text{SAD}_{50}$ and $\text{SSD}_{50}$.

We decided for the *Cones* image pair for testing this approach. For this pair it seems that 50% is a good choice for all the used cost functions. However, a more representative study is needed to conclude properly for this type of tuning.

The tuning of SAD and SSD functions tends to improve results slightly for the original versions of SAD and SSD, but leads to slightly worse results for zero-mean versions. The big difficulty seems to choose the optimum percentage for given image pairs. Another approach might be to discard all differences that are lower than a certain threshold, and then just use the mean of SAD or SSD (to assure comparability between costs).

**Table 5.** Results of different kinds of NCC cost function for different image pairs with window size equals 7: the given percentages denote perfect matches. Also evaluated is the census 2D function.

| *Window k=7 / l=7* | NCC | ZNCC | 1D NCC | 1D ZNCC | B-spline$^{NCC}$ | ZB-spline$^{NCC}$ | Census | ZCensus |
|---|---|---|---|---|---|---|---|---|
| TSUKUBA | 68.7% | 67.8% | 55.9% | 50.9% | 63.4% | 58.8% | 39.2% | 48.0% |
| CONES | 73.1% | 74.7% | 58.9% | 49.6% | 60.9% | 51.9% | 50.1% | 61.3% |
| TEDDY | 69.9% | 71.2% | 47.4% | 38.5% | 50.2% | 41.4% | 38.6% | 50.3% |
| ART | 53.0% | 53.4% | 42.3% | 30.5% | 43.4% | 33.3% | 32.2% | 35.4% |
| BOOKS | 58.9% | 58.6% | 41.7% | 28.5% | 45.1% | 36.8% | 31.0% | 39.4% |
| LAUNDRY | 49.5% | 47.7% | 37.5% | 26.6% | 40.7% | 32.9% | 23.5% | 25.9% |
| MOEBIUS | 66.0% | 65.8% | 52.7% | 40.4% | 55.8% | 45.7% | 42.1% | 49.3% |
| DOLLS | 65.6% | 65.7% | 54.1% | 41.4% | 55.5% | 43.9% | 42.1% | 46.9% |
| REINDEER | 64.4% | 64.3% | 43.7% | 34.8% | 46.9% | 37.4% | 35.1% | 44.5% |
| *mean* | | | | | | | | |
| perfect matches | 63.2% | 63.2% | 48.2% | 37.9% | 51.3% | 42.5% | 37.1% | 44.6% |
| mean ambiguity | 1.00 | 1.00 | 1.04 | 1.46 | 1.00 | 1.00 | 1.55 | 2.10 |

### 5.5 Correlation-Based Measures

We briefly comment about the correlation-based measure; see Table 5. For window size $k = 15$ or $l = 15$, the 1D NCC performs almost as good as its 2D version (or even better (!); see *Art*, *Dolls* and *Reindeer*). There is also not a big difference between zero-mean and original versions in general. In most cases, the zero-mean version performs even slightly worse.

For window size 7, however, the NCC ($k = 7$) outperforms its 1D NCC ($l = 7$) equivalent. The 1D ZNCC also performs badly. With respect to computational time and accuracy, a 1D original version of NCC may be recommend if the $1 \times l$ window is of sufficiently large size, due to time benefits.

### 5.6 Subpixel Accuracy Using B-splines

The results for B-spline based functions, as proposed in Section 3.3, are slightly worse compared to those for the 1D SAD and SSD cost functions (both the original and the zero-mean version). For NCC, the subsampling of the

B-spline curve seems to improve the result slightly. However, another approach is suggested by the following concept of subpixel accuracy.

We compare again 1D SAD, SSD and NCC (note: the original versions) against their B-spline versions. This time we do not allow any deviation from the ground truth (i.e., we only accept the rounding to the nearest integer value in case of applying a parabolic curve fit). For the B-spline-based cost function, we do not use curve fitting but apply the following concept of subpixel accuracy.

Assume two different cost functions and the same minimum cost, and that both approaches apply a parabolic fit through this identical minimum cost value $d$ and its two neighboring cost values at $d-1$ and $d+1$, identifying a rational minimum $d_{min}$; still both cost functions may lead to different results (within the range of $\pm 1$ disparity). Because we can subsample the B-spline intensity signal with a step size of less than 1, we calculate costs for rational disparities $d \in \mathbb{Q}$ instead of for integer values only. The idea is that this improves the chance to identify a unique minimum.

In the following experiments we used the image pairs of the quarter-sized *Teddy* and the *Cones*, testing for matches from 0 to $d_{max}$ by going in increments of $\frac{1}{4}$. For example, let us assume that we like to evaluate for $d_{max} = 10$, and the minimum is at $d_{min} = 6.75$. The original version may find a minimum at $d = 7$ and fits a parabolic curve through costs at $d = 6$, $d = 7$ and $d = 8$. The proposed method does not fit through those costs. Instead it tests for $d = 6.0$, 6.25, 6.5, 6.75, and so forth, and may (or may not) find the minimum at 6.75 as well. This approach is evaluated in Table 6. We consider a match as being valid only if it coincides with the ground truth. The 'enhanced' B-spline cost function, using the proposed subpixel approach, is denoted by subscript $subP$.

**Table 6.** B-spline cost function evaluated over windows with $l = 7$ and $l = 15$, compared with their corresponding original 1D versions; subpixel sampling either by common parabolic curve fit or by proposed B-spline based subsampling of intensities.

| | SAD / $SAD_{subP}$ | | SSD / $SSD_{subP}$ | | NCC / $NCC_{subP}$ | | ZSAD / $ZSAD_{subP}$ | | ZSSD / $ZSSD_{subP}$ | | ZNCC / $ZNCC_{subP}$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Window $l$=15 | | | | | | | | | | | | |
| TEDDY | 24.6% | 25.7% | 27.4% | 28.5% | 31.3% | 34.1% | 29.5% | 33.8% | 31.5% | 34.3% | 31.1% | 34.3% |
| CONES | 25.8% | 20.6% | 28.6% | 26.1% | 33.7% | 34.4% | 33.3% | 34.9% | 33.8% | 34.4% | 33.6% | 34.8% |
| | | | | | | | | | | | | |
| Window $l$=7 | | | | | | | | | | | | |
| TEDDY | 18.9% | 21.4% | 21.5% | 23.6% | 22.1% | 28.8% | 20.4% | 28.3% | 22.0% | 28.8% | 17.5% | 26.4% |
| CONES | 20.1% | 16.1% | 22.7% | 21.3% | 29.6% | 36.0% | 29.0% | 35.6% | 29.9% | 36.1% | 24.6% | 34.0% |

The accuracy seems to improve in almost every case, except for SAD and SSD in case of the *Cones* image pair. Especially NCC seems to benefit from a performance gain. Also, the performance gain of this approach seems to be larger when evaluated over the smaller window with $l = 7$. For a window with $l = 15$, the gain is not as significant. But values are on a higher level. This is another clue that 1D windows need to have a sufficient size to perform well. However, this gain comes with the cost of testing more disparities: if we test, for example, for $d_{max} = 10$, then we actually test for 40 disparities if using increments of $\frac{1}{4}$.

Thus, the proposed B-spline based subsampling of intensities improves results, but is not yet suitable for realtime applications.

## 6 Conclusions

Zero-mean versions outperform the original versions for SAD and SSD, but not for NCC. The zero-mean normalization seems to have even a negative impact in this case.

Computation time can be saved by choosing the 1D version of cost functions over a large window (e.g., $l = 15$) instead of using a 2D version over a smaller window (e.g., $k = 7$).

We introduced the use of B-spline curves for cost computation on subsampled image intensities. The subpixel accuracy approach using B-splines seems to improve results. Including first order information (like orientation of curve normals) into the cost calculation, or choosing different subsampling strategies, leaves some space for further investigations of the B-spline-based approach.

Pixel-based cost calculations (AD and BT) lead to high ambiguities. This should be considered when applying those in a stereo matching technique.

We introduced the idea of discarding a percentage of the lower differences for cost functions of type SAD and SSD. Further studies are needed here for general conclusions.

**Acknowledgement.** The authors thank Laura Zechel for her support in table and image formating. Without her help this paper would not have met the submission deadline.

## References

1. Birchfield, S., and Tomasi, C.: Depth discontinuities by pixel-to-pixel stereo. *Int. J. Computer Vision*, **35**:269–293 (1999)
2. Farin, G.: *Curves and Surfaces for CAGD. A Practical Guide.* Fifth edition, Morgan Kaufmann, San Francisco (2001)
3. Felzenszwalb, P.F., and Huttenlocher, D. P.: Efficient belief propagation for early vision. In Proc. *CVPR*, volume 1, pages 261–268 (2004)
4. Franke, U.: Progress in space-time machine vision. Talk at Freiburg University, http://www2.faw.uni-freiburg.de/kolloquium/ss08/franke.pdf (2008)
5. Hermann, S., Klette, R., and Destefanis, E.: Inclusion of a second-order prior into semi-global matching. In Proc. *Pacific Rim Symp. Image Video Technology (PSIVT)*, LNCS 5414, pages 633–644 (2009)
6. Hirschmüller, H.: Accurate and efficient stereo processing by semi-global matching and mutual information. In Proc. *IEEE Conf. Computer Vision Pattern Recognition (CVPR)*, volume 2, pages 807–814 (2005)
7. Hirschmüller, H.: Stereo vision in structured environments by consistent semi-global matching. In Proc. *IEEE Conf. Computer Vision Pattern Recognition (CVPR)*, volume 2, pages 2386–2393 (2006)

8. Hirschmüller, H., and Scharstein, D.: Evaluation of stereo matching costs on images with radiometric differences. *IEEE Trans. Pattern Analysis Machine Intelligence*, http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.221 (2008)
9. Rogers, D. F.: *An Introduction to Nurbs. With Historical Perspective.* Morgan Kaufmann, San Francisco (2001)