Automatika

Journal for Control, Measurement, Electronics, Computing and Communications

ISSN: 0005-1144 (Print) 1848-3380 (Online) Journal homepage: https://www.tandfonline.com/loi/taut20

# Depth sensing with disparity space images and three-dimensional recursive search

Miroslav Rožić & Tomislav Pribanić

**To cite this article:** Miroslav Rožić & Tomislav Pribanić (2018) Depth sensing with disparity space images and three-dimensional recursive search, Automatika, 59:2, 131-142, DOI: 10.1080/00051144.2018.1503137

**To link to this article:** https://doi.org/10.1080/00051144.2018.1503137



Published online: 14 Aug 2018.

AUTOMATIKA
2018, VOL. 59, NO. 2, 131–142
https://doi.org/10.1080/00051144.2018.1503137

REGULAR PAPER

OPEN ACCESS Check for updates

# Depth sensing with disparity space images and three-dimensional recursive search

Miroslav Rožić[a] and Tomislav Pribanić[b]

[a]Zagrebačka Banka d.d., Zagreb, Croatia; [b]Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia

**ABSTRACT**

We present a coarse-to-fine stereo matching optimization applicable to methods utilizing the Disparity Space Image (DSI) structure. With the Three-dimensional Recursive Search algorithm (3DRS), a coarse disparity seed is obtained first, with minimal computational effort. The coarse disparity seed is then used as a guidance to locally compute the DSI disparity space with a reduced number of disparity hypotheses, yielding significantly shorter execution times for the disparity computation. The method performance was measured on the well-known Dynamic Programming (DP) DSI-based method and the images from the Middlebury set. The DP method with the DSI optimization applied maintains or improves the overall level of disparity map accuracy while delivering a near sevenfold speed-up of execution in comparison to DP alone. We furthermore show that the optimized method's performance does not depend on the expected input disparity range, which is commonly restricted, or expected to be defined upfront, for DSI-based stereo matching methods.

**ARTICLE HISTORY**
Received 6 March 2018
Accepted 16 July 2018

**KEYWORDS**
Stereo vision; disparity computation; dynamic programming; DSI; 3DRS; census transform

## 1. Introduction

Computation of dense disparity maps is one of the most researched problems in computer vision, specially in the area of 3D modeling and reconstruction. The methods used are generally divided into active, employing a camera combined with structured light [1] or LIDAR, and passive methods, focusing on finding correspondences within two or more stereo images. A taxonomy of dense, two frame, passive stereo methods has been proposed [2] generally dividing passive methods into local, which generate disparity maps through local matching cost aggregation, or global, which aim to minimize a global energy function. Global methods, such as Graph Cuts [3] or Belief Propagation [4] generally outperform the local in terms of accuracy, but the execution time of global methods is significantly inferior, making them a poor choice for real-time applications required by recent applications such as autonomous vehicles. Therefore, usually faster but less accurate global methods such as Dynamic Programming or hybrid methods such as Semi-Global Matching [5] are a more common choice for real-time implementations.

Dynamic programming (DP), introduced for edge-based stereo estimation with one of the first methods by Ohta and Kanade [6], has been identified as a sufficiently fast global approach to solving the stereo matching problem, used in real-time solutions [7]. Our work focuses on a pixel-based stereo DP algorithm as described by Cox et al. [8] and further expanded by Bobick and Intille [9] with the introduction of the Disparity Space Image (DSI) concept and Ground Control Points (GCPs). Veksler [10] proposed the use of tree-based structures for matching as opposed to scan-lines. Other tree-based methods were proposed afterwards [11]. Real-time implementations have been explored with CPU [7] and GPU [12] hardware. Coarse-to-fine DP approaches were also explored [13], as well as guided approaches [14]. Methods have also been defined to address the scanline inconsistencies. Some of them include using a tree-like structure which spans multiple scanlines [10], aggregating the matching cost across scanlines [15], reusing calculated paths [7] or performing a second DP pass in the vertical direction.

While many research endeavors have tried to improve upon the accuracy of DP-based methods, these approaches have generally yielded an increase in computational complexity, which has been in turn addressed by high-performance hardware architectures [12]. Semi-global matching [5] (SGM), widely used in many real-time and real-world applications, makes extensive use of DP to compute the disparity for each pixel in the image by estimating the disparity for multiple paths intersecting for every pixel of the image. For each image pair a tensor of all possible disparities is computed, and for each path at each pixel a DSI is formed as a cross-section of the tensor in the path direction (horizontal, vertical, or diagonal). For methods

**CONTACT** Miroslav Rožić miroslav.rozic@gmail.com Zagrebačka Banka d.d., Samoborska cesta 145, Zagreb HR-10000, Croatia

such as SGM and many methods derived from it, optimizing the DP and DSI computation would yield significant improvements in execution time and memory footprint.

The goal of our research was therefore chosen to improve upon the performance of Dynamic Programming by reducing its computational load, in order to facilitate its use in real-time applications and within embedded systems. In this work, we will demonstrate a hybrid approach to DP which utilizes a coarse block-matching algorithm (3DRS) [16] as a pre-processing step to determine the range of disparities to be searched with DP. The 3DRS algorithm has been introduced by De Haan et al. as a motion estimation method for de-interlacing and frame rate up-conversion in high definition digital televisions. It has been further enhanced with hierarchical computation and penalized predictors [17]. It was shown by Hendriks and Marosi [18] that 3DRS can be applied to obtain disparity maps.

This work is organized as follows. Section 2 introduces the DP and 3DRS methods, as well as the combined hybrid method. Section 3 describes the implementation details and the obtained results. A discussion of the obtained results is given in Section 4. We present the conclusion of our work and future areas of investigation in Section 5.

## 2. Methods

### 2.1. Dynamic programming

Global stereo matching approaches aim to minimize a global energy function,

$$E(d) = E_{data}(d) + \lambda E_{smooth}(d) \tag{1}$$

by establishing the disparity function $d$ which minimizes $E$. In Equation (1), the data term $E_{data}$ accounts for the matching cost of the image pair, whereas the smoothness term $E_{smooth}$ accounts for the desired smoothness assumptions about the disparity image. $E_{data}$ can be defined as

$$E_{data}(d) = \sum_{(x,y)} C(x, y, d(x, y)), \tag{2}$$

using the disparity space formulation, where C is the matching cost disparity space image (DSI). The smoothness term is usually restricted to measurement of the disparities between neighboring pixels

$$E_{smooth}(d) = \sum_{(x,y)} \rho(d(x, y) - d(x + 1, y)) + \rho(d(x, y) - d(x, y + 1)), \tag{3}$$

and assigns a penalty to the discontinuities in the disparity map. The 2D optimization of Equation (1) has been shown to be NP-hard [2] for common classes of smoothness functions. However, the 1-dimensional case can be computed in polynomial time with DP, providing a global minimum for individual scanlines.

The algorithm operates under assumptions of *uniqueness* – that a single feature in the left image maps to a single feature in the right image, and *monotonicity* (also referred to as the *Ordering constraint* [8]), that the relative ordering of pixels on a scanline remains the same between the two views. Based on those assumptions, the algorithm computes the disparity by calculating the minimum matching cost path through the DSI (*x*,*disparity*) image for a pair of scanlines. The calculation of the shortest path is performed within the DSI as shown in Figure 1. Occlusions are explicitly handled by assigning a group of pixels from one image to a single pixel in another image, which corresponds to a discontinuity or a gap in the optimum path. The path which satisfies the ordering constraint traverses the DSI using three principal moves: a *match*, which moves in the $x$ direction keeping a constant disparity, a *vertical occlusion* corresponding to a right image occlusion and decreasing the disparity, and a *diagonal occlusion* which moves both in the $x$ direction and increases the disparity, corresponding to a left occlusion.

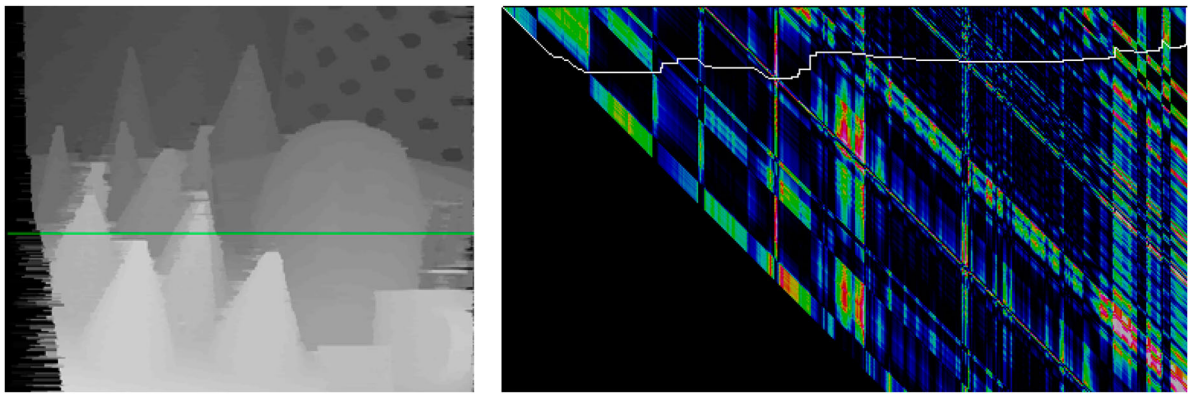


**Figure 1.** Computed disparity map and the corresponding DSI matching cost matrix structure. The computed path for the disparity scan-line highlighted in the left image is traced in white. The position $x$ is assigned to the horizontal axis, while the disparity $d$ is assigned to the vertical axis. The values in the structure represent the pairwise matching costs and are shown using a Jet color map.

The DSI is generated by Algorithm 1, in a common first forward pass of DP approaches which calculates the cost optimum. To obtain the actual disparities, the Algorithm 2 backtracks through the computed DSI along the optimum path, outputting a resultant pixel of the dense disparity map at each step.

**Algorithm 1** DP DSI computation

```
Require: DSI image with the size X_max, d_max + 1
Require: Match image with the size X_max, d_max + 1
  procedure ComputeDSI(I_l, I_r, y, DSI, Match)
    for i = 0 to X_max − 1 do
      d_high ← i < d_max ? i : d_max;
      DSI(i, 0) ← i * OccCost;
      DSI(i, d_high) ← i * OccCost;
    end for
    for i = 0 to X_max − 1 do
      d_high ← i < d_max ? i : d_max;
      for j = d_high to 1 do
        DiagCost ← DSI(i − 1, j − 1) + OccCost;
        VertCost ← DSI(i, j + 1) + OccCost;
        MatchCost ← DSI(i − 1, j) + C(I_l(i, y), I_r(i − j, y));
        MinCost ← min(DiagCost, VertCost, MatchCost);
        DSI(i, j) ← MinCost
        if MinCost = DiagCost then
          Match(i, j) ← DiagMove
        else if MinCost = VertCost then
          Match(i, j) ← VertMove
        else if MinCost = MatchCost then
          Match(i, j) ← MatchMove
        end if
      end for
    end for
  end procedure
```

**Algorithm 2** DP DSI backtracking

```
Require: Computed Match image with the size X_max, d_max + 1
  procedure Backtrack(D, y, Match)
    D_x ← D_max;
    x ← X_max − 1;
    while x > 0 and D_x > 0 do
      Move ← Match(x, D_x);
      if Move = DiagMove then
        Handle Left Occlusion
        x ← x − 1;
        D_x ← D_x − 1;
      else if Move = VertMove then
        Handle Right Occlusion
        D_x ← D_x + 1;
      else if Move = MatchMove then
        D(x, y) ← D_x;
        x ← x − 1;
      end if
    end while
  end procedure
```

An important factor in the execution of Algorithm 1 is the *OccCost* parameter, or the occlusion cost, which is the cost assigned to occluded pixels and greatly impacts the output results of the algorithm as it directly affects the decision whether the pixel is a match or occluded. The described DP method is repeated for each scanline to obtain the complete disparity map.

### 2.2. Three-dimensional recursive search (3DRS)

The 3DRS algorithm is a *block-matching algorithm*; the input source image is divided into blocks, for each of which the location of a best-matching block is searched in the reference image. A two-dimensional vector defining the distance between the original block position and the position of the matching block in the reference image is assigned. The map of vectors for all of the image blocks describes either the motion or disparity. For disparity estimations of rectified stereo pairs, the vertical component can be ignored.

The blocks are matched through a *matching cost function*, such as the *sum of absolute differences* (SAD). The proper matching block is determined by finding the extreme value of the matching cost function. To find the extreme an optimization method is usually applied, where the 3DRS algorithm reduces the number of required candidate blocks (or candidate vectors) by making two principal assumptions: first, that the objects or features in the image are larger than blocks, and second, that objects have inertia. The first assumption implies that the vectors of the neighboring blocks provide a good predictor for the block vector which is currently being estimated. The second assumption implies that a temporal predecessor – a previously estimated motion vector, is also a good candidate for the new vector. The application of these assumptions to the estimation process significantly reduces the number of candidates which have to be evaluated. To introduce variation, *update vectors* which are either generated randomly, from a distribution, or from a predefined set, are added to a subset of the predictors. The variation introduced by the update vectors ensures that the algorithm will progress towards a solution regardless of the initial conditions. The typical initial conditions are that the vector field contains only zero vectors. An example of the update vector set is

$$U_v = \left\{ \begin{bmatrix} \pm 2^k \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ \pm 2^k \end{bmatrix} \right\} \quad k = 0, 1, 2, 3, \ldots, k_{max} \tag{4}$$

Sequentially for every block in the image, 3DRS selects the output motion (or disparity) vector from a set of prediction vectors chosen from a spatio-temporal neighborhood, additionally applying updates in order to evaluate alternate solutions, as seen in Figure 2. For an individual block, the matching cost function is evaluated for each predictor. The predictor with the best matching cost is then assigned as the new vector for the current block, which in turn becomes a part of the candidate set for the next location, as the estimation progresses to the adjacent block.

The *temporal* aspect of 3DRS implies the reuse of a previous estimation. The initial state of the 3DRS vector field is that all vectors are set to zero, from where they converge towards a solution by aggregating update vectors during the estimation process. Between algorithm iterations on one or more pairs of images, the 3DRS vector field is not reset, enabling the reliable estimate calculated for the current pass to be used as a starting point in the next pass. The neighbourhood of vectors always contain the spatial candidates (estimated within the current iteration) and temporal candidates (estimated in the previous iteration).

The block evaluation order is defined by the *scanning direction* as shown in Figure 3. The most common order is from top-left to bottom-right, with each line

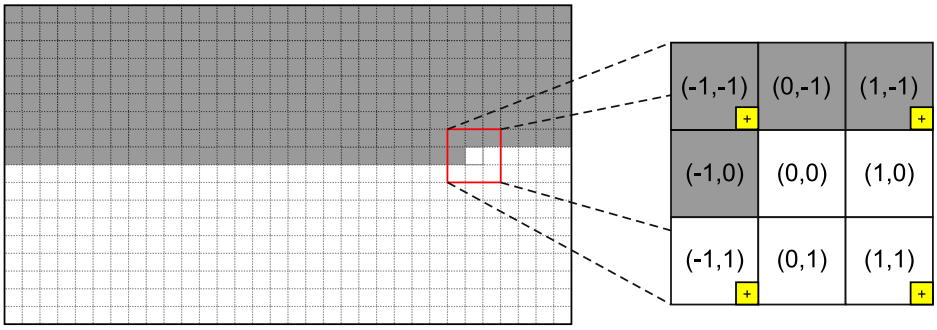



**Figure 2.** The spatio-temporal neighborhood used in the 3DRS estimation. The vector for the block at (0,0) is currently being estimated. Vectors for the blocks at (−1,−1), (0,−1), (1,−1) and (−1,0) have already been estimated for this image. The remainder of the blocks used are from a previous estimation, making them temporal predictors. The predictors coming from the blocks marked with a "+" are additionally modified with update vectors.

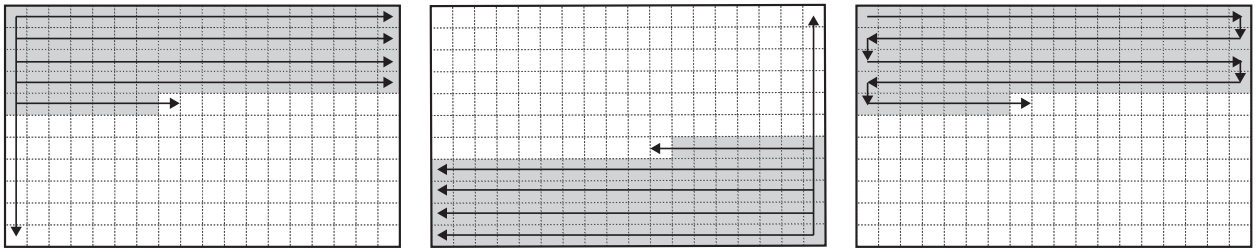



**Figure 3.** Various scanning orders for 3DRS. The shaded areas represent the progression of newly computed estimates. (a) Top to bottom, left to right. (b) Bottom to top, Right to left. (c) Meandering.

beginning on the left. This order defines which of the predictors are spatial, and which also have a temporal component, as seen in Figure 2. It also defines the direction in which the newly estimated, "good" values propagate within the image. An important aspect of 3DRS is the property of *convergence* [19], which is a measure of how fast the algorithm reaches the correct estimate. As estimated values propagate throughout the image, with the variation introduced by the update component, individual estimates may require several algorithm iterations, and updates, to reach the "correct" value. This is especially critical at the start of the estimation where all vectors start from a defined initial state. Selecting a particular update set can improve on this property, but can also introduce noise. Changing the scanning order from bottom-right to top-left on each new image, or alternating the scanning direction in a *boustrophedonic* or *meandering* way as shown in Figure 3(c), yields the propagation of good estimates in all four directions, helping the 3DRS to quickly converge on the final vector map. For the correct estimates to propagate even faster, multiple scans can be done for one image pair, alternating the direction and the meander of the scan on each pass.

The search range of the 3DRS algorithm is constrained in practice only by specific implementation details and available buffer memory. For a general software implementation, we can assume that the search range is constrained to the maximum allowed by the image horizontal resolution, which is the maximum disparity that can be detected. The iterative nature of the algorithm, the constant evaluation of new candidate vectors and propagation of previous good estimates throughout the image ensure that, given a sufficient number of iterations, any disparity which can be detected in the image pair shall be correctly estimated, as the algorithm will naturally converge towards a correct matching vector. The convergence speed can be further tuned by the number of iterations and the selection of the update set. Therefore, it can be assumed that 3DRS will converge towards a coarse disparity solution in deterministic time (fixed number of iterations) given any disparity range within the scene, without the need to test all of the possible disparities as with winner-take-all methods. This property of the algorithm allows us to efficiently pre-determine the search range for a subsequent fine estimation step, such as DP.

### 2.2.1. Formal 3DRS definition

The 3DRS algorithm analyzes two images: the source image $I_S$ and the reference image $I_R$ shown in Figure 4.

The source image $I_S$ is divided into a rectangular grid of $M \times N$ blocks of $n \times n$ pixels as shown in the figure. For each block $B_{ij}$, $i$ denotes the row index and $j$ denotes the column index of the block. A motion vector $\vec{d}_{ij}$ is assigned to each block.

The pixel position vector $\vec{p}_{ij}$ for each block in $I_S$ is defined as

$$\vec{p}_{ij} = n \cdot \begin{bmatrix} i \\ j \end{bmatrix} \tag{5}$$

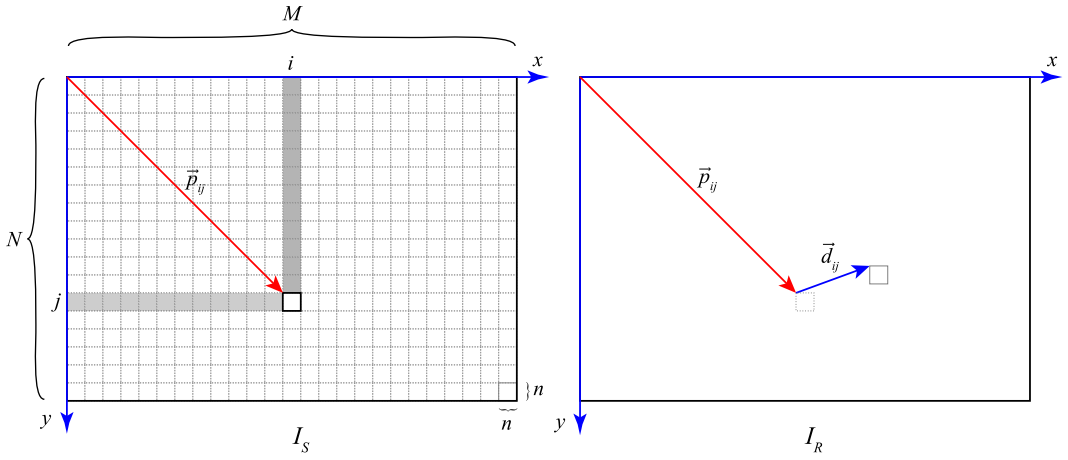



**Figure 4.** 3DRS image space and coordinate system.

This vector points from the origin to the upper-left pixel of each block.

The 3DRS algorithm consists of the following steps:

(1) For each block $B_{ij}$ in $I_S$, form a set of predictors:

$$P = \begin{cases} \vec{d}_{ij} \\ \vec{d}_{i+k\,j} \\ \vec{d}_{i\,j+l} \\ \vec{d}_{i+k\,j+l} + \vec{u}_v \end{cases} \qquad k = -1, 1;\ l = -1, 1 \quad (6)$$

In the set above, $\vec{u}_v$ is the update vector, and a new one is selected either randomly or from a predefined set, for each of the predictors it is applied to.

(2) Evaluate the matching cost for each of the predictors, and assign to $\vec{d}_{ij}$ the predictor with the minimum matching cost.

$$\vec{d}_{ij} = arg \min_{\vec{d}_P \in P} (MC(\vec{p}_{ij}, \vec{p}_{ij} + \vec{d}_P)) \quad (7)$$

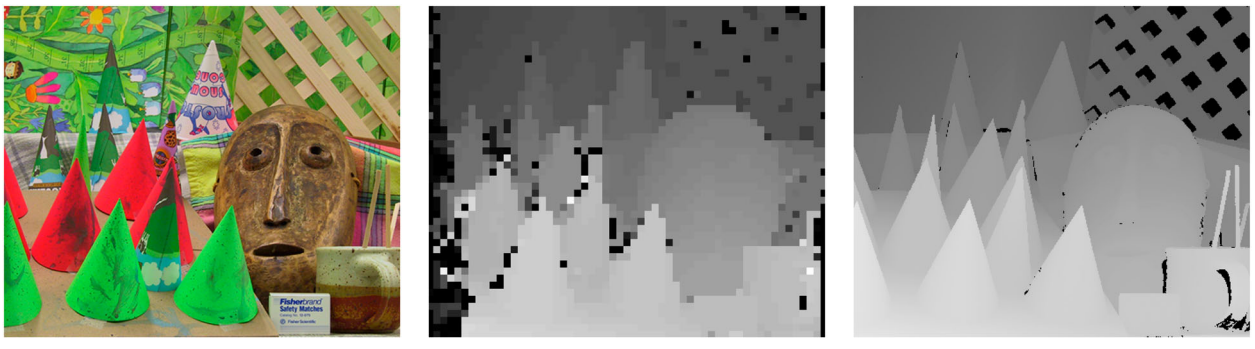
where $MC(\vec{s},\vec{r})$ is the aggregated matching cost for the compared blocks.

$$MC(\vec{s},\vec{r}) = \sum_{y=0}^{n-1} \sum_{x=0}^{n-1} C\left(I_S\left(\vec{s} + \begin{bmatrix} x \\ y \end{bmatrix}\right), I_R\left(\vec{r} + \begin{bmatrix} x \\ y \end{bmatrix}\right)\right) \quad (8)$$

(3) Repeat the previous two steps for every block in the image based on the selected scanning direction.

### 2.3. 3DRS-guided dynamic programming (3GDP)

The 3DRS motion estimator is extremely applicable in coarse-to-fine hierarchical approaches due to its short execution time and the ability to produce coarse disparity maps. The initial assumption of many stereo matching methods is that the stereo image pairs have been previously rectified to satisfy the *epipolar constraint* [2]. Under these conditions the matching can be performed between image scanlines. Applied to the 3DRS estimator, this constraint removes the need for the estimation of the $y$ component of the vector. Furthermore, as the disparity values, unlike motion vectors, are highly unlikely to be negative, the output of the disparity estimation can be clipped to address only the positive values. A 3DRS estimator with the described modifications is used as the initial step of our proposed method, named 3DRS-guided Dynamic Programming, or 3GDP. Figure 5 shows the output of the 3DRS estimator. The estimates are largely correct, with spurious



(a) (b) (c)

**Figure 5.** The result of the 3DRS disparity estimation. (a) Left image of the stereo pair. (b) The 3DRS disparity map. (c) Ground truth disparities.

outliers where the algorithm was not able to converge due to occlusions.

To compute the dense disparities from the coarse map we apply the DP algorithm. The computational cost of the DP method is directly proportional to the range of disparities being estimated. To reduce the computational cost, the coarse disparity map is used as a reference for piece-wise limitation of the disparity range within the DSI, which is divided into segments with width matching the width of the 3DRS blocks. Each segment has an assigned disparity range $(d_s^{\min}, d_s^{\max})$, which can be determined from the coarse disparity value plus or minus a range constant, an algorithm parameter. However, this does not account for the discontinuities in the optimum matching path. Use of varying estimated values obtained from the coarse disparity map, especially with the presence of unmatched outliers, may result in disconnected ranges, which would effectively prevent proper DSI traversal and backtracking in the DP step of the method. In order for the estimation to work, a fully end-to-end connected DSI structure must be ensured.

To achieve this, we re-apply the 3DRS assumption that the spatial neighborhood of the estimated block provides a good predictor for that block. By building the disparity range for a particular segment based on the values of neighboring segments with an additional clearance constant $R_{Off}$, we ensure that each of the segments is connected to its neighboring segments while retaining space for potential deviations from the coarse block disparity.

To generate a connected DSI, the $d_s^{\min}$ and $d_s^{\max}$ are computed for each segment as shown in Equation (9).

$$\begin{aligned} d_s^{\min}(x,y) &= \min(d_{3drs}(x+i, y+j)) - R_{Off} \\ d_s^{\max}(x,y) &= \max(d_{3drs}(x+i, y+j)) + R_{Off} \\ i &= -1, 0, 1 \\ j &= -1, 0, 1 \end{aligned} \tag{9}$$

The resultant segmented DSI is shown in Figure 6(b). Using the segmented DSI, the DP algorithm produces a dense disparity map. The comparison of the DP and 3GDP DSI structures demonstrates the reduction of required computational effort provided by the coarse 3DRS disparities.

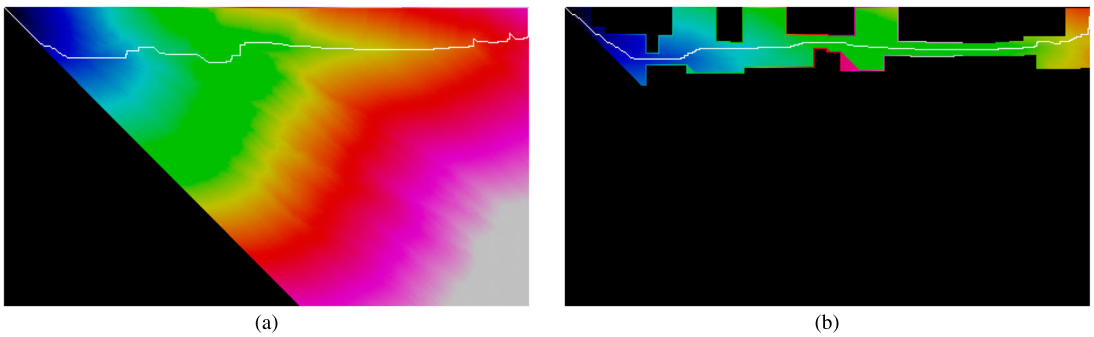
### 2.4. Matching cost

The proper selection of a matching cost function is vital for all passive stereo correspondence methods. In practice, the selected matching cost should provide the best possible matching accuracy under radiometric variations of input images, such as exposure differences, vignetting, varying lighting or noise [20]. In our hybrid approach, the matching cost quality is especially important for the coarse 3DRS step, as its output constrains the minimum cost path search area for the dense DP step. Evaluations [21] have shown that the Census [22] non-parametric matching cost provides the overall best performance. We have evaluated the SAD, ZSAD and Census matching cost with the 3DRS algorithm to determine which one has the best characteristic in our method.

## 3. Experimental results

The 3DRS, DP and the proposed hybrid 3GDP algorithm have been implemented within *StereoTest*, a visual evaluation environment we have developed for the purpose of evaluating stereo algorithms. The motivation for designing our own simulation environment was to reduce the dependency of the tested methods on external libraries as much as possible, with the ultimate goal of developing a real-time hardware implementation for the tested methods. All computation and image encoding is performed using integer and fixed-point arithmetic, with integer parameters. The environment does not implement a sub-pixel disparity refinement step after the optimization, as the goal of the evaluation was to compare the raw results of the disparity computation methods. The environment is coded in C# and operates under Microsoft's .Net Framework 4.5.2.

In our implementation we have tested the SAD, SSD, Zero-mean SAD, and Census cost functions. The comparison of the 3DRS results with different matching costs is shown in Figure 7, with a quantitative



(a) (b)

**Figure 6.** DSI structures and computed paths for DP and 3DRS-guided-DP algorithms. Shaded areas represent computed matching costs. The value of the cost is shown using the jet color map. (a) DP with the full disparity range. (b) 3DRS-guided DP.

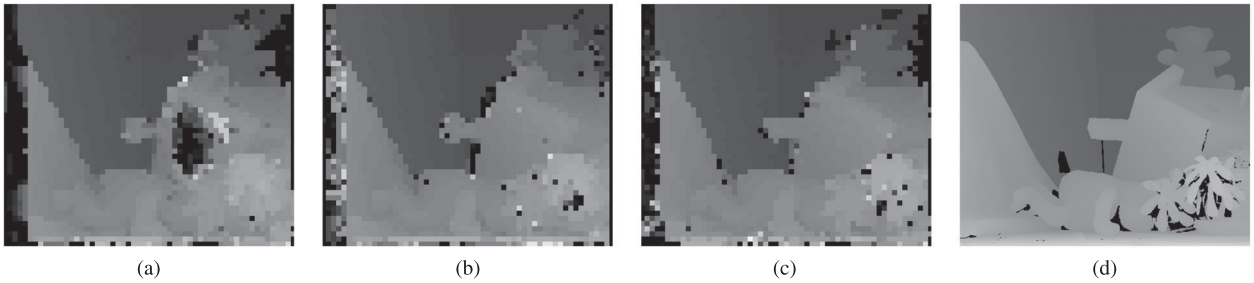



**Figure 7.** 3DRS coarse disparity maps for the "Teddy" test image based on different matching costs. (a) Sum of Absolute Differences (SAD). (b) Zero-mean SAD (ZSAD). (c) Census transform. (d) True disparities.

**Table 1.** Comparison of error rates for various matching costs using the 3DRS algorithm, considering all pixels. Lower is better.

| Set | SAD (%) | ZSAD (%) | Census (%) |
|---|---|---|---|
| Tsukuba | 12.61 | 14.40 | 21.44 |
| Venus | 8.23 | 7.37 | 6.85 |
| Teddy | 30.26 | 24.58 | 20.65 |
| Cones | 26.70 | 22.50 | 16.22 |
| Average | 19.45 | 17.21 | **16.29** |

Note: Bold values highlight the final result of the proposed algorithm.

analysis provided in Table 1, showing that the census cost achieves the lowest matching error rate. Based on the results, we have selected the Census as the matching cost to be used as the block matching cost in the 3DRS step, as well as the pixel-wise matching cost in the DP step.

The 3DRS implementation does not include advanced optimizations such as hierarchical processing or penalties [17], but supports multiple iterations to improve algorithm convergence on static images. SAD, ZSAD, and Census [23] cost functions can be utilized for block matching.

Our DP implementation follows the algorithm described in Section 2.1. As the computation of each scanline is individual, this introduces scanline inconsistencies visible as streaking artifacts. One of the noted approaches [7] is to reuse the costs and computed path from a previous pass with an applied weighting factor, thus constraining the new path to roughly follow the previously computed path, contributing to vertical smoothness. Our approach therefore employs a similar

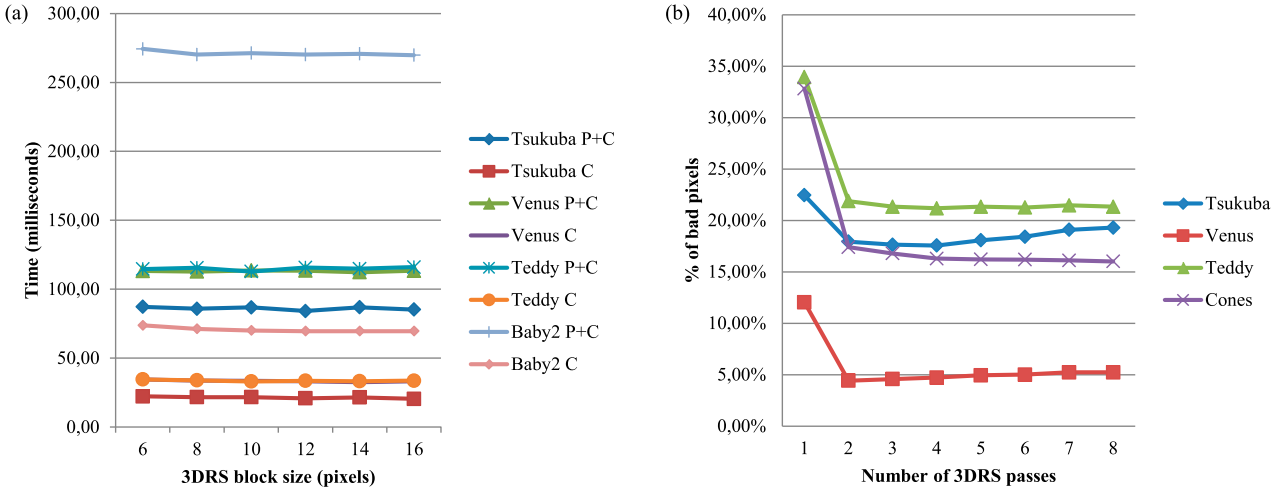



**Figure 8.** Execution speed and accuracy of implemented methods. (a) Execution times (in milliseconds) of 3DRS for the Middlebury set with varying 3DRS block sizes – data from Table 2. (b) Percentage of bad pixels in the 3DRS coarse disparity map shown in dependency to the number of 3DRS passes – data from Table 3.

**Table 2.** 3DRS execution time for the Middlebury set, measured in milliseconds, for varying 3DRS Block size.

| | | | Blocksize | | | | | |
|---|---|---|---|---|---|---|---|---|
| Set | Image size | Action | 6 | 8 | 10 | 12 | 14 | 16 |
| Tsukuba | 384×288 | Prep+Calc | 87.17 | 85.82 | 86.76 | 84.12 | 86.85 | 85.19 |
| | | Calc | 22.24 | 21.65 | 21.70 | 20.89 | 21.51 | 20.42 |
| Venus | 434×383 | Prep+Calc | 113.24 | 112.80 | 113.63 | 113.42 | 112.29 | 113.40 |
| | | Calc | 34.50 | 33.69 | 33.59 | 33.13 | 32.48 | 33.12 |
| Teddy | 450×375 | Prep+Calc | 114.65 | 115.56 | 112.81 | 115.66 | 114.83 | 116.02 |
| | | Calc | 34.55 | 33.86 | 33.03 | 33.60 | 33.25 | 33.68 |
| Baby2 | 620×555 | Prep+Calc | 274.35 | 270.22 | 271.25 | 270.18 | 270.73 | 269.75 |
| | | Calc | 73.75 | 71.19 | 70.02 | 69.53 | 69.56 | 69.57 |

**Table 3.** Accuracy of produced disparity maps (percentage of bad pixels, lower is better) vs. the number of 3DRS passes and elapsed time. The time is measured in milliseconds.

| | | Number of passes | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Set | Measure | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Tsukuba | Time | 89.33 | 102.08 | 134.33 | 158.77 | 174.91 | 203.98 | 226.12 | 250.37 |
| | Bad pixels | 22.46% | 17.94% | 17.64% | 17.56% | 18.07% | 18.41% | 19.09% | 19.30% |
| Venus | Time | 136.24 | 168.53 | 203.13 | 240.88 | 272.20 | 305.13 | 337.91 | 372.16 |
| | Bad pixels | 12.05% | 4.43% | 4.57% | 4.72% | 4.94% | 5.00% | 5.23% | 5.23% |
| Teddy | Time | 134.37 | 168.57 | 202.62 | 236.48 | 270.21 | 303.23 | 337.22 | 370.04 |
| | Bad Pixels | 33.95% | 21.88% | 21.33% | 21.18% | 21.33% | 21.26% | 21.47% | 21.34% |
| Cones | Time | 136.30 | 169.61 | 203.42 | 236.47 | 271.72 | 304.60 | 339.29 | 374.92 |
| | Bad Pixels | 32.82% | 17.40% | 16.79% | 16.29% | 16.22% | 16.19% | 16.12% | 16.02% |

**Table 4.** Accuracy of generated disparities, expressed as a percentage of incorrect pixels (lower is better); Non-occ – non-occluded regions only; All – all pixels; Disc – discontinuity regions only.

| | 3DRS | | | DP | | | 3GDP | | |
|---|---|---|---|---|---|---|---|---|---|
| Set | Non-occ | All | Disc | Non-occ | All | Disc | Non-occ | All | Disc |
| Tsukuba | 16.57% | 17.94% | 32.79% | 5.91% | 7.27% | 23.29% | 6.02% | 7.38% | 23.29% |
| Venus | 3.12% | 4.43% | 23.99% | 2.89% | 4.18% | 15.09% | 3.45% | 4.70% | 15.05% |
| Teddy | 14.22% | 21.88% | 33.51% | 8.77% | 16.61% | 20.09% | 8.98% | 17.32% | 21.11% |
| Cones | 9.16% | 17.90% | 25.57% | 5.22% | 13.84% | 13.58% | 4.84% | 13.98% | 13.79% |
| Average | | 18.42% | | | 11.40% | | | **11.66%** | |

Note: Bold values highlight the final result of the proposed algorithm.

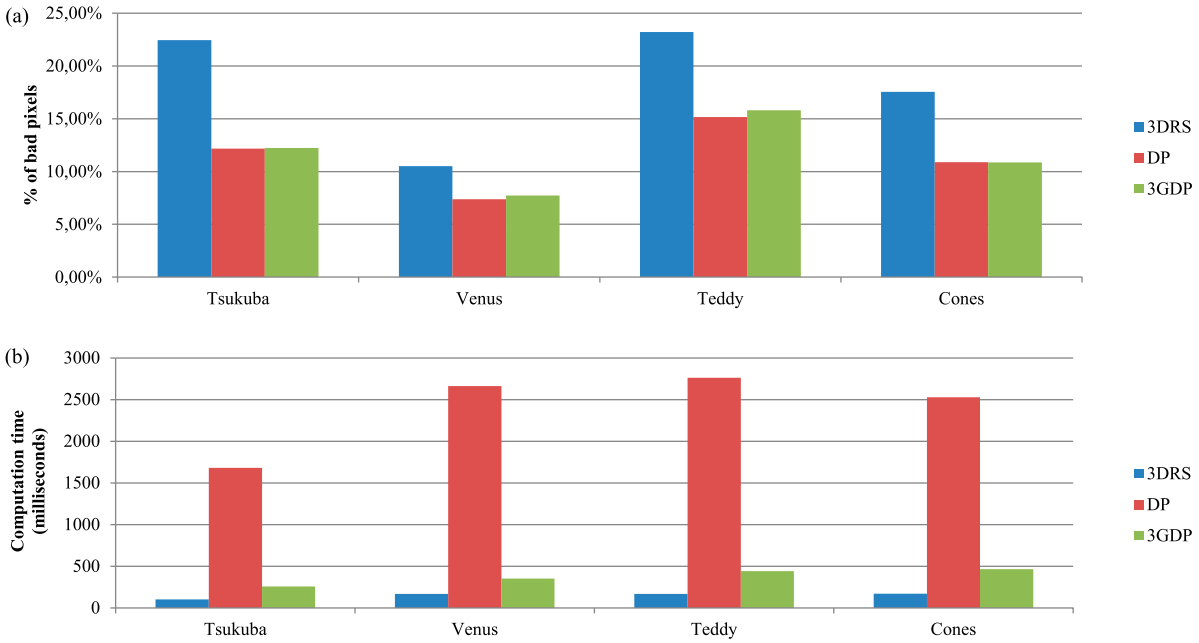



**Figure 9.** Comparison of accuracy for the computed disparities and method execution times for the images of the Middlebury set. Lower is better. (a) Accuracy for the computed disparities – data from Table 4. (b) Method execution times – data from Table 5.

vertical smoothing scheme, which adds a weighted cost from the previous calculated path to the cost of the current calculated path. We have found that satisfactory results are achieved with the weight parameter set to 180 (ranging from 0 to 255). The *OccCost* parameter was selected empirically based on the results. For the Census cost, it was observed that the value of *OccCost* = 8 performs well for all images. The reduced influence of the *OccCost* parameter on the final result is a result of employing the Census cost, which reduces the impact of radiometric differences.

For the $R_{off}$ parameter used to constrain the DSI path, using a $R_{off} = 5$ has shown good results.

**Table 5.** Comparison of execution times for the 3DRS, DP and 3GDP method. Times are given in milliseconds. The speed-up ratio is given as a dimensionless quantity.

| Set | $t_{3DRS}$ | $t_{DP}$ | $t_{3GDP}$ | $\frac{t_{DP}}{t_{3GDP}}$ |
|---|---|---|---|---|
| Tsukuba | 102.08 | 1680.5 | 258.09 | **6.51** |
| Venus | 168.53 | 2662.7 | 351.51 | **7.58** |
| Teddy | 168.57 | 2761.8 | 440.41 | **6.27** |
| Cones | 169.61 | 2527.3 | 465.58 | **5.43** |
| Average | 152.198 | 2408.1 | 378.9 | **6.45** |

Note: Bold values highlight the final result of the proposed algorithm.

There were no observable differences in quality for $R_{off} \geq 4$.

In our evaluation of the algorithm, we have measured two aspects of quality for the implemented

**Table 6.** Accuracy of generated disparities for the 3GDP method compared with other DP-based methods on the Middlebury stereo vision web site [24] (lower is better); Non-occ – non-occluded regions only; All – all pixels; Disc – discontinuity regions only.

| | DP [2] | | | SGM [5] | | | 3GDP | | |
|---|---|---|---|---|---|---|---|---|---|
| Set | Non-occ | All | Disc | Non-occ | All | Disc | Non-occ | All | Disc |
| Tsukuba | 4.12% | 5.04% | 12.0% | 3.26% | 3.96% | 12.8% | 6.02% | 7.38% | 23.29% |
| Venus | 10.10% | 11.00% | 21.00% | 1.00% | 1.57% | 11.30% | 3.45% | 4.70% | 15.05% |
| Teddy | 14.0% | 21.60% | 20.60% | 6.02% | 12.20% | 16.30% | 8.98% | 17.32% | 21.11% |
| Cones | 10.50% | 19.10% | 21.10% | 3.06% | 9.75% | 8.90% | 4.84% | 13.98% | 13.79% |
| Average | | 13.93% | | | 7.51% | | | **11.66%** | |

Note: Bold values highlight the final result of the proposed algorithm.

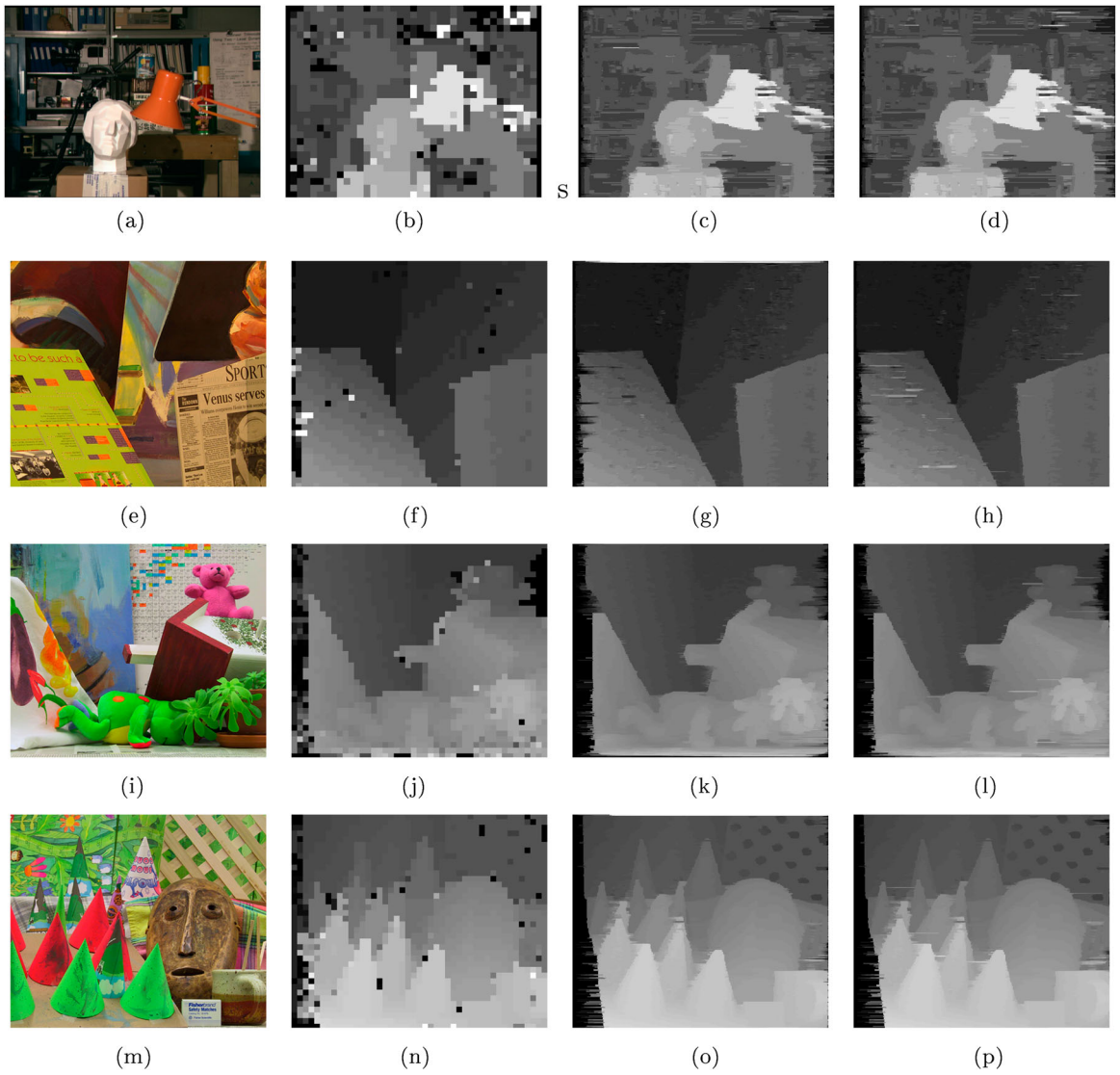



**Figure 10.** Simulation results – Left reference image and resultant disparities for three methods. (a) Tsukuba – Left image. (b) Tsukuba – 3DRS result. (c) Tsukuba – DP result. (d)Tsukuba – 3GDP result. (e) Venus – Left image. (f) Venus – 3DRS result. (g) Venus – DP result. (h) Venus – 3GDP result. (i) Teddy – Left image. (j) Teddy – 3DRS result. (k) Teddy – DP result. (l) Teddy – 3GDP result. (m) Cones – Left image. (n) Cones – 3DRS result. (o) Cones – DP result. (p) Cones – 3GDP result.

methods. The first is the accuracy of the output results indicated by the percentage of incorrect pixels

$$B = \frac{1}{N} \sum_{(x,y)} (|d_C(x,y) - d_T(x,y)| > \delta_D) \qquad (10)$$

where $d_C$ is the estimated disparity map and $d_T$ is the ground truth disparity map [2]. The threshold $\delta_D$ is 1. The second aspect is the algorithm run-time performance, indicated by the computation time. The method was tested on reference images from the Middlebury set [24]: "Tsukuba", "Venus", "Teddy" and "Cones".

With the behavior of DP mostly well researched in previous work, we focused our measurement on the properties of the 3DRS phase in order to extract the parameters which would yield the highest quality guidance for the DP step.

The relationship between the 3DRS block size and the computation time per image is shown in Table 2 and Figure 8(a). The *Prep+Calc* time involves both the

preparation of resources and input images (memory allocation, Census transform of inputs), while the *Calc* time measures only the time required to perform a single 3DRS pass. The measurement was performed for a single 3DRS pass, using the Census cost.

Another important property of the 3DRS algorithm which affects the guidance of the DP step is convergence. We assume that using a low-confidence guidance map, which exhibits a greater estimation error, will adversely affect the final results of the 3GDP method. Therefore, we measure the accuracy of the 3DRS-produced coarse disparity map depending on the number of 3DRS passes. The measurement was performed with a blocksize of 10, using the Census cost. The results are provided in Table 3 and Figure 8(b). The percentage of incorrect pixels was calculated using Equation 10, with $\delta_D = 1$, using the full image area with defined disparities (including the occluded areas).

Based on these results, we compare the final outputs of 3DRS, DP, and 3GDP in terms of execution time and disparity map accuracy for both full image and non-occluded areas. Both 3DRS and DP passes employ the Census matching cost. The 3DRS method performs two passes in all cases (standalone and 3GDP). The measured accuracy of individual methods and the combined 3GDP method are provided in Table 4 and Figure 9(a).

Additionally, in Table 6 we compare the accuracy of our 3GDP method with the accuracies from the reference DP method implemented in [2], and the accuracy of the widely used SGM method [5].

The execution time of methods is shown and compared in Table 5 and Figure 9(b). The resultant disparity maps are shown in Figure 10.

## 4. Discussion

The 3DRS estimation has been shown to operate with constant time regardless of the selected block size, depending solely on the input image resolution. As the 3DRS block size rises, the number of matching costs to calculate per vector increases, but the overall number of the vectors in the image also decreases as less blocks cover the image. The results suggest that the actual number of pixels to be calculated for matching cost, which is the dominant component in the computation complexity of 3DRS, does not change significantly with varying block sizes and is near-constant. The implications of this fact are that the 3DRS estimation, used to obtain a coarse disparity map, incurs only a constant, small, resolution-dependent penalty, and that the coarseness of the produced disparity map, defined by the block size can be freely tweaked without impacting performance. In our work we have selected the block size of 10, which maps well to the features in all tested images.

The convergence results show that a single 3DRS pass produces a disparity guidance with visibly lesser confidence than after multiple passes, however, after only two passes the confidence improves to the point where subsequent passes do not improve the confidence further. These results match the expectations based on 3DRS theory, as the estimation starts from a zero-initialized state and requires several update cycles to reach a good estimate, which means that a certain good estimate will be reached mid-image. As the vertical direction alternates between passes, the second pass will start from known good estimates and propagate them to the less good estimates from the first pass. After two meandering passes, we can assume that all vectors have a solid degree of confidence. Subsequent passes can improve the result further (although this is apparently image-dependent), but add constant time penalties. For this reasons, we have selected a two-pass 3DRS pre-estimation in our 3GDP method.

It is therefore evident from the results that the disparity estimation obtained from 3DRS does not change significantly after only two passes. Moreover, the disparity estimation enables the final accurate disparity computation via DP, independently from the input disparity range which is different for the tested pairs of images. In other words, unlike other methods, the proposed 3GDP method does not require an explicit definition of the overall range of disparities within which the algorithm has to search for the optimum solution. Instead, the range is locally constrained with the 3DRS coarse result.

The pure DP method provides the overall best quality in our evaluation, by a small margin. However, the execution time required ranges between 1.7 and 2.7 seconds in our implementation. The 3GDP method produces very close results (off by on average 0.3% of the absolute score), but the execution time of 3GDP is 6.45 times faster on average. Therefore, with the hybrid 3GDP approach we have obtained a near seven-fold speedup while maintaining the overall level of accuracy. The overall average disparity error of our method is 11.66%, comparable with the results obtained by other scan-line DP-based approaches. As compared in Table 6, our method obtains higher accuracy than the basic DP method as benchmarked in [2], however, in DSI-based methods, it is still outperformed by SGM [5]. The execution speed, on average, exceeds the speed obtained by [25].

Although the proposed solution does not always improve on the accuracy of other DSI-based algorithms, this can be attributed to several factors, such as the omission of the Disparity refinement step (as defined by the taxonomy of Scharstein and Szeliski [2]). However, we see the near-sevenfold acceleration and memory footprint reduction for DSI methods as the main contribution of our method to the state of the art, as it shows that the existing DSI based methods

can be significantly accelerated without considerably decreasing their accuracy.

## 5. Conclusion and future work

We have presented a novel approach to stereo matching by combining a standard Dynamic Programming estimation with the 3DRS block-based motion estimator. The resultant method exhibits a nearly seven-fold increase in performance from the original DP method in our implementation, while retaining the same level of quality. The proposed method is also insensitive to the range of input disparities and is not limited by calculation in a particular disparity range. Overall, the use of 3DRS as a method of reducing and defining the DSI space has potential applications to other DSI-based methods, or other methods which estimate within fixed range boundaries. Also, the natural ability of 3DRS to estimate 2D vectors might provide a guidance with inputs which do not exhibit perfect epipolar rectification, such as the latest Middlebury test sets [26]. In our future work, we will further explore these methods, aiming to improve the accuracy of the final result as well as further improve the speed, with the ultimate goal of defining an architecture for a real-time embedded hardware implementation.

## Disclosure statement

No potential conflict of interest was reported by the authors.

## References


[1] Salvi J, Fernandez S, Pribanic T, et al. A state of the art in structured light patterns for surface profilometry. Pattern Recognit. 2010;43(8):2666–2680.

[2] Scharstein D, Szeliski R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int J Comput Vision. 2002;47(1–3):7–42.

[3] Kolmogorov V, Zabih R. Computing visual correspondence with occlusions via graph cuts. International Conference on Computer Vision. Vancouver, BC: IEEE Computer Society; 2001. p. 508–515.

[4] Sun J, Zheng NN, Shum HY. Stereo matching using belief propagation. IEEE Trans Pattern Anal Mach Intell. 2003;25(7):787–800.

[5] Hirschmuller H. Accurate and efficient stereo processing by semi-global matching and mutual information. Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - -Volume 2 – Volume 02; Washington, DC, USA. IEEE Computer Society; 2005. p. 807–814; CVPR '05.

[6] Ohta Y, Kanade T. Stereo by intra- and inter-Scanline search using dynamic programming. IEEE Trans Pattern Anal Mach Intell. 1985;7(2):139–154.

[7] Forstmann S, Kanou Y, Ohya J, et al. Real-time stereo by using dynamic programming. Computer Vision and Pattern Recognition Workshop, 2004. CVPRW '04. Conference on. Washington DC: IEEE Computer Society; 2004 Jun. p. 29–29.

[8] Cox IJ, Hingorani SL, Rao SB, et al. A maximum likelihood stereo algorithm. Comput Vis Image Underst. 1996;63:542–567.

[9] Bobick AF, Intille SS. Large occlusion stereo. Int J Comput Vis. 1999;33(3):181–200.

[10] Veksler O. Stereo correspondence by dynamic programming on a tree. Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on; Vol. 2. San Diego, CA: IEEE Computer Society; 2005 Jun. p. 384–390.

[11] Bleyer M, Gelautz M. Simple but effective tree structures for dynamic programming-based stereo matching. International Conference on Computer Vision Theory and Applications (VISAPP); 2008. p. 415–422. Vortrag: International Conference on Computer Vision Theory and Applications (VISAPP 2008), Funchal, Madeira - Portugal; 2008 Jan 22–25.

[12] Congote J, Barandiaran J, Barandiaran I, et al. Realtime dense stereo matching with dynamic programming in CUDA. San Sebastian, Spain. Eurographics Association; 2009. p. 231–234.

[13] Cai J. Fast stereo matching: coarser to finer with selective updating. Image and Vision Computing New Zealand 2007; Hamilton, New Zealand. Image and Vision Computing New Zealand; 2007. p. 266–270.

[14] Chang X, Zhou Z, Wang L, et al. Real-time accurate stereo matching using modified two-pass aggregation and winner-take-all guided dynamic programming. 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2011 International Conference on. Hangzhou: IEEE Computer Society; 2011 May. p. 73–79.

[15] Wang L, Liao M, Gong M, et al. High-quality real-time stereo using adaptive cost aggregation and dynamic programming. 3rd International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT 2006), 2006 June 14–16, Chapel Hill, North Carolina, USA; 2006. p. 798–805.

[16] de Haan G, Biezen P, Huijgen H, et al. True-motion estimation with 3-d recursive search block matching. IEEE Trans circuits Sys Video Tech. 1993;3(5):368–379, 388.

[17] Heinrich A, Bartels C, van der Vleuten R, et al. Optimization of hierarchical 3drs motion estimators for picture rate conversion. IEEE J select top sign proc. 2011;5(2):262–274.

[18] Hendriks EA, Marosi G. Recursive disparity estimation algorithm for real time stereoscopic video applications. ICIP (2); 1996. p. 891–894.

[19] Braspenning RA, de Haan G. True-motion estimation using feature correspondences. Vol. 5308; 2004. p. 396–407.

[20] Hirschmuller H, Scharstein D. Evaluation of cost functions for stereo matching. Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on. Minneapolis, MN: IEEE Computer Society; 2007 Jun. p. 1–8.

[21] Hirschmuller H, Scharstein D. Evaluation of stereo matching costs on images with radiometric differences. IEEE Trans Pattern Anal Mach Inte. 2009;31(9):1582–1599.

[22] Zabih R, Woodfill J. Non-parametric local transforms for computing visual correspondence. Proceedings of the Third European Conference on Computer Vision (Vol. II); Secaucus, NJ, USA. Springer-Verlag New York, Inc.; 1994. p. 151–158; ECCV '94.

[23] Hirschmuller H, Scharstein D. Evaluation of stereo matching costs on images with radiometric differences. IEEE Trans Pattern Anal Mach Intell. 2009;31(9):1582–1599.
[24] Scharstein D, Szeliski R. Middlebury stereo vision page. 2014. [cited 2014 Jan 31]: Available from: http://vision.middlebury.edu/stereo
[25] Salmen J, Schlipsing M, Edelbrunner J, et al. Real-time stereo vision: Making more out of dynamic programming. Computer Analysis of Images and Patterns, 13th International Conference, CAIP 2009, Münster, Germany, 2009 Sep 2–4. Proceedings. p. 1096–1103.
[26] Scharstein D, Hirschm'uller H, Kitajima Y, et al. High-resolution stereo datasets with subpixel-accurate ground truth. In: Jiang X, Hornegger J, Koch R, editors. GCPR; (Lecture Notes in Computer Science; Vol. 8753). Springer; 2014. p. 31–42.