Article

Semantic Edge Based Disparity Estimation Using Adaptive Dynamic Programming for Binocular Sensors

Dongchen Zhu [1,2,*], Jiamao Li [1], Xianshun Wang [1,2], Jingquan Peng [1,2,3], Wenjun Shi [1,2] and Xiaolin Zhang [1]

1 Bio-Vision System Laboratory, State Key Laboratory of Transducer Technology, Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China; jmli@mail.sim.ac.cn (J.L.); xswang@mail.sim.ac.cn (X.W.); pengjq@mail.sim.ac.cn (J.P.); wjs@mail.sim.ac.cn (W.S.); xlzhang@mail.sim.ac.cn (X.Z.)
2 University of Chinese Academy of Sciences, Beijing 100049, China
3 School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China
* Correspondence: dchzhu@mail.sim.ac.cn

Received: 14 February 2018; Accepted: 30 March 2018; Published: 3 April 2018

Abstract: Disparity calculation is crucial for binocular sensor ranging. The disparity estimation based on edges is an important branch in the research of sparse stereo matching and plays an important role in visual navigation. In this paper, we propose a robust sparse stereo matching method based on the semantic edges. Some simple matching costs are used first, and then a novel adaptive dynamic programming algorithm is proposed to obtain optimal solutions. This algorithm makes use of the disparity or semantic consistency constraint between the stereo images to adaptively search parameters, which can improve the robustness of our method. The proposed method is compared quantitatively and qualitatively with the traditional dynamic programming method, some dense stereo matching methods, and the advanced edge-based method respectively. Experiments show that our method can provide superior performance on the above comparison.

Keywords: sparse stereo matching; disparity estimation; semantic edge; dynamic programming; binocular sensor

1. Introduction

Depth acquisition is crucial for environment perception of intelligent devices, and it plays an important role in various applications, such as 3D reconstruction, Augmented Reality (AR), and vehicle navigation. Sensors that can be used to obtain depth include laser sensors, visual sensors, and radar sensors. Nowadays, laser and radar sensors have been widely used. With the progress of computer vision algorithms, visual sensors represented by binocular cameras show strong competitiveness, which are becoming an important research direction of automatic driving. Binocular sensors acquire depth by estimating the disparities, which aims to calculate the coordinate difference between corresponding points in the rectified stereo images. According to the principle of triangulation, the depth of each point is easy to measure with its disparity, since the disparity is inversely proportional to the distance from cameras.

Numerous algorithms have been widely studied to solve this problem over the past decades. According to the density of the generated disparity maps, these methods can be divided into two categories: dense stereo matching and sparse stereo matching. The dense matching methods strive to obtain each pixel's disparity. Although the traditional methods, such as Graph Cuts (GC) [1,2], Semi-Global Matching (SGM) [3], and the emerging approaches based on deep learning, like MC-CNN [4,5], have achieved high accuracy, there is an intractable problem of heavy computation in these dense methods. While the sparse methods based on points and lines are less studied than dense matching in recent years, they are not given up by the researchers who seek the balance between accuracy and efficiency because of the advantage of less computational complexity.

Point-based sparse methods have been effectively applied for the richly textured regions, but they fail in textureless areas [6]. The methods based on straight and curved lines possess universality, but most of the lines or edges extraction methods are based on low-level features with the lack of scene understanding. Calculating only these pixels' disparities cannot guarantee that the basic demands for visual navigation are satisfied, which means that these algorithms have difficulties with being applied in practice.

In this paper, we propose a new sparse disparity estimation method for outdoor navigation. The proposed approach focuses on the pixel matching of semantic edges because those contours not only segment out objects in the scene, but also give them different meanings. Estimating disparities of these semantic edges can meet the basic requirements of recognition, location, and obstacle avoidance in navigation completely. We also put forward an adaptive dynamic programming algorithm, which can automatically select the parameters to improve the robustness of our method.

We organize the remainder of this paper as follows. Section 2 introduces the related work about the stereo matching. The specific matching approach will be introduced in detail in Section 3. Some qualitative and quantitative experiments are demonstrated in Section 4. Section 5 makes a brief summary at the end of this paper.

2. Related Work

According to [7], the dense stereo matching can be generally divided into four steps: cost calculation, cost aggregation, disparity optimization, and optionally refinement. In fact, plenty of sparse methods also follow these four steps.

Absolute Difference (AD), the Sum of the Absolute Difference (SAD), Census transform (Census), and Normalization Cross Correlation (NCC) are all classic cost calculation methods [8]. Then [9] found that the combination of different costs can draw on each other's strength. They combined AD and Census to get a better result. This idea has been widely applied and we will also draw lessons from it. With the development of deep learning, the methods of calculating cost using neural networks, like [5], have also achieved excellent results in dataset tests. However, these methods are still limited in the ground truth of datasets for training. Our method still will choose the classic costs.

There are a variety of ways to aggregate costs. Mean filtering, median filtering, etc. can be regarded as aggregation with a window of fixed size. The more popular one is the adaptive aggregation method with irregular windows. Ke Zhang et al. [10] proposed the cross-based aggregation that determines the support window by comparing the similarity between the surrounding pixels and the anchor pixels, and suggested considering the left and right images together. However, this method is time-consuming. In this paper, we do not take the step of cost aggregation, but choose to increase the window size appropriately.

The disparity optimization has been widely studied, creating a variety of algorithms. Winner-Take-All (WTA) is the simplest one without considering global information. Dynamic programming (DP) has been widely used in dense and sparse matching, and is the method to be adopted in this paper. Belief propagation (BP) and Graph Cuts (GC) based on Markov random field also have excellent performance. In recent years, the most widely used is Semi-Global Matching (SGM). This algorithm employs multi-path one-dimensional optimizations to solve two-dimensional optimization problem by designing the constraints between each anchor pixel and its neighborhoods. This method is still very competitive since being proposed, and some results based on SGM will be compared with our method in Section 4.

In this paper, we do not modify the disparities obtained by optimization, so we will not introduce the refinement methods. Next, we will mainly discuss the related edge-based methods. They can also be divided into two kinds: based on straight lines and based on curved edges. Line-based approaches assume that lots of straight lines can be detected in the images. They have achieved results in stereo [11] and Structure From Motion (SFM) [12]. However, when there are fewer lines detectable in the scene, such as our outdoor navigation scenarios, these approaches are useless.

Curve-based approaches are improved with the development of edge extraction methods. In [13], the positions of edges are detected by differentiating and then linked to connected edges. Then, dynamic programming is used in intra- and inter-scanline search with edge-delimited intervals as elements to be matched. The advantage of [13] is that the inter-scanline constraint is taken into consideration, but the constraint is so strong that only the edges crossing multiple scanlines are preserved, without considering the horizontal edges. Moreover, the inter-scanline search results in a high cost of computation.

In [14], a sparse BP algorithm is designed to match the nodes of the mesh of left image, formed by edge and non-edge pixels, with the all pixels of the right image. Compared with the dense BP, the computational complexity of [14] is reduced. However, only the number of pixels to be matched is reduced, the matching search range of each pixel is not changed. The possible mismatches of dense BP algorithm cannot be avoided yet.

The methods proposed in [15–17] are all based on canny edges. In [15], some corresponding corner points are selected as seed points, and then the correspondences between edges around those seed points are determined. Seed points are helpful to narrow the range of edges to be matched, but the mismatches of seeds can lead to edge matching errors.

The methods of cost calculation and aggregation are the same in [16,17]. They choose the minimum of both asymmetric pixel-strips on either side of a candidate edge as the matching cost of the edge. In [16], a confidence-based refinement algorithm is applied after the initial WTA process, while a dynamic programming algorithm is implemented on each edge segment in [17]. They both take into consideration the consistency and smoothness of edges. However, the description of each edge pixel can be ambiguous, which is easy for causing a mismatch.

Subhayan Mukherjee et al. [18] uses K-means clustering to segment the left image, and refines the segment boundaries by morphological filtering. Then, SAD and WTA are used to match the edges of the left image with the full right image. Finally, a dense disparity map can be obtained by disparity propagation. The idea of using K-means segment boundaries is novel, but the value of k needs to be specified according to different images. In addition, the defects of WTA and sparse-dense matching mentioned above do not get addressed.

Dexmont Peña et al. [19] extended the Edge Drawing (ED) algorithm on stereo pairs, matched only a few anchor points and then propagated disparities along the edges. Using the proposed method, the number of computations can be greatly decreased. However, this method is highly dependent on the matching accuracy of anchor points. A tiny match error can be propagated to the entire edge.

With the development of deep learning, the semantic edge extraction method has been researched. Gedas Bertasius et al. [20] used VGG [21] to detect edges and then used semantic segmentation networks to classify edges. The newest [22] proposed an end-to-end network that detected semantic edge with multi-category labels. Our method will be based on Ref. [22]'s edges to estimate disparities, considering edge points as candidates to be matched and using an improved dynamic programming algorithm to obtain the global optimal solution for each scanline, which will be described in detail below.

3. Proposed Method

In this section, our proposed Semantic Edge Matching (SEM) method is described in detail, which is illustrated in Figure 1. The input of our method is a calibrated and rectified stereo image pair, and the output is a disparity map, which is calculated sparsely for semantic edges in the scene. For this purpose, we first use the CASENet to detect the semantic edges of the left and right images, respectively. These extracted edge pixels are candidates to be matched in this paper. In order to find the correspondence between pixels in stereo images, we attribute the matching of each scan-line's pixels to the alignment of two sequences, and then this problem can be solved by a dynamic programming paradigm. Before solving the matching problem, we first need to measure the similarity between the matching candidates, and this is the calculation of the cost. In Section 3.2, we will introduce the cost computing methods used in experiments. Next, we propose an Adaptive Dynamic Programming (ADP) algorithm because the traditional dynamic programming method has the difficulty of selecting empirical parameters, which should not be the same for different scenes. ADP makes use of the potential consistency in stereo image pairs to select optimal parameters adaptively, and two kinds of consistencies can be applied in it. The results of our method will be analyzed qualitatively and quantitatively in Section 4. In the following subsections, we will elaborate on each step of our method.

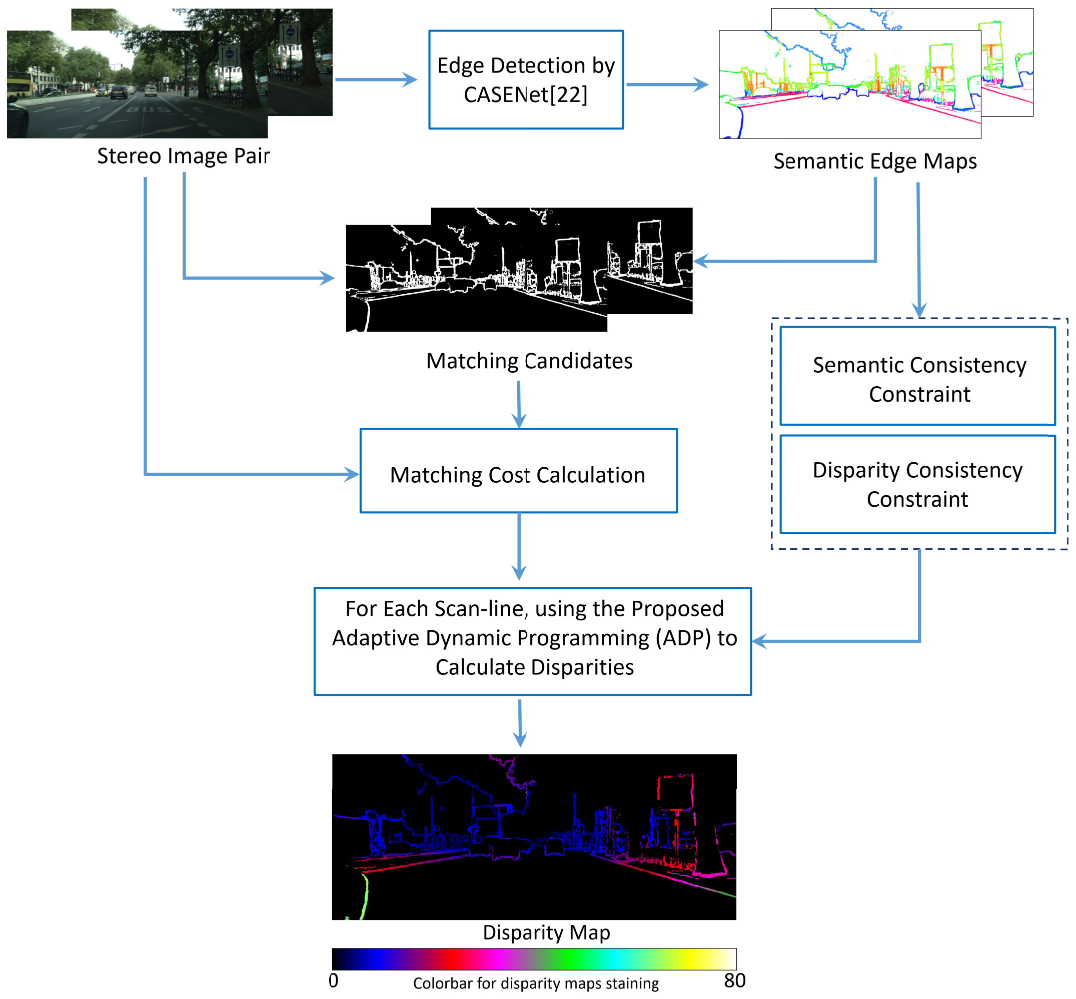
Stereo Image Pair
Edge Detection by CASENet[22]
Semantic Edge Maps
Matching Candidates
Semantic Consistency Constraint
Disparity Consistency Constraint
Matching Cost Calculation
For Each Scan-line, using the Proposed Adaptive Dynamic Programming (ADP) to Calculate Disparities
Disparity Map
0
Colorbar for disparity maps staining
80

Figure 1. Pipeline of our proposed method.

3.1. Semantic Edge Detection and Semantic Edge Matching Formulation

As a kind of important feature of images, edges are widely used in a series of computer vision problems. There are many existing methods of edge extraction, from the low-level method using simple convolutional filters such as Sobel [23] and Canny [24] to the high-level method with semantic neural networks such as High-for-Low approach (HFL) [20]. In this paper, our selected method is CASENet [22], which is a semantic edge detection network with category-aware. There are two main reasons why we choose this method:

1. CASENnet differs from the traditional methods in that it detects not the low-level edges, but the external contours of each object in the image. Since the lines inside of the object are not detected, there are fewer matching candidates in this paper, which can reduce the interference of some similar edges.
2. Unlike the traditional edge detection to solve a binary classification problem, in which a pixel belongs to edges or not, CASENet employs a new modeling approach that allows pixels in contours belonging to more than one category. In other words, CASENet detects object contours with semantic meaning. Therefore, only matching these pixels can satisfy the basic ranging requirement in navigation because the distance of each target in the scene can be calculated. At the same time, the multi-contour features of the edge pixels also make it possible for each pixel to measure the similarity in semantic space. This feature will be taken advantage of in Section 3.3.

An example of CASENnet's result is shown in Figure 3, and the meanings of different colors are the same as in [22]. There is still one thing with which we should be concerned. The width of each edge extracted by CASENet is several pixels rather than one pixel. With these edge pixels as the matching candidates, we formulate the semantic edge matching problem in this paper as:

Given two arrays $\{\boldsymbol{L}_1, \boldsymbol{L}_2, ..., \boldsymbol{L}_m\}$ and $\{\boldsymbol{R}_1, \boldsymbol{R}_2, ..., \boldsymbol{R}_n\}$ for each pair of scan-lines in left image $\boldsymbol{I}_L$ and right image $\boldsymbol{I}_R$, we aim to find the alignment $\boldsymbol{M}$ of minimum cost $C(\boldsymbol{M})$, in which:

(1) $\boldsymbol{L}_i = (x_{\boldsymbol{L}_i}, y_{\boldsymbol{L}_i})$ and $\boldsymbol{R}_j = (x_{\boldsymbol{R}_j}, y_{\boldsymbol{R}_j})$ are the coordinates of pixels to be matched in left and right images respectively, and there exists $y_{\boldsymbol{L}_i} = y_{\boldsymbol{R}_j}$.

(2) $\boldsymbol{M}$ is a set of pairs $\{\boldsymbol{L}_i, \boldsymbol{R}_j\}$ in order. More specifically, each item in left array $\{\boldsymbol{L}_1, \boldsymbol{L}_2, ..., \boldsymbol{L}_m\}$ has at most one corresponding item in right array $\{\boldsymbol{R}_1, \boldsymbol{R}_2, ..., \boldsymbol{R}_n\}$ and vice versa, and if there are two pairs $\{\boldsymbol{L}_{i_1}, \boldsymbol{R}_{j_1}\}$ and $\{\boldsymbol{L}_{i_2}, \boldsymbol{R}_{j_2}\}$, there must be $j_1 > j_2$ when $i_1 > i_2$.

(3) The cost of $\boldsymbol{M}$ is defined as:

$$C(\boldsymbol{M}) = \sum_{\{\boldsymbol{L}_i, \boldsymbol{R}_j\} \in \boldsymbol{M}} C_{\boldsymbol{L}_i, \boldsymbol{R}_j} + \sum_{\boldsymbol{L}_i or \boldsymbol{R}_j \text{unmatched}} C_{\text{gap}}, \tag{1}$$

where $C_{\boldsymbol{L}_i, \boldsymbol{R}_j}$ is the color cost between $\boldsymbol{L}_i$ and $\boldsymbol{R}_j$, which represents the similarity between the two pixels; C_{gap} is the unmatched cost if $\boldsymbol{L}_i$ or $\boldsymbol{R}_j$ has no matched item.

The formulated problem can be solved by dynamic programming paradigm. However, the traditional dynamic programming method suffers parameters selection difficulty. Therefore, we propose an improved method with an adaptive parameter to solve the above problem. Then, the disparity d of each pixel in left image can be calculated by

$$d_{\boldsymbol{L}} = x_{\boldsymbol{L}_i} - x_{\boldsymbol{R}_j}. \tag{2}$$

Different from other stereo matching methods, we do not set the value of maximum disparity in our method. However, in order to show the results more clearly, we set a maximum disparity for consistent staining. The color range is shown as Figure 1.

3.2. Color Cost Calculation

The first and most important step of stereo matching is to describe the potential property of each waiting-for-matched pixel effectively, and then scale the comparability between different pixels. That is to say, the matching cost is computed at first to represent the similarity of pixels. We will introduce three kinds of matching costs below, which will be used in our method.

The first one is SAD, which can be implemented quickly by using the mean filtering on AD. They both assume that the brightnesses of corresponding pixels are consistent. The mathematical expression of AD and SAD is as Equations (3) and (4), respectively:

$$C_{\text{AD}}(x, y, d) = |\boldsymbol{I}_L(x, y) - \boldsymbol{I}_R(x - d, y)|, \tag{3}$$

$$C_{\text{SAD}}(x,y,d) = \sum_{(x',y') \in \boldsymbol{N}_{(x,y)}} |\boldsymbol{I}_L(x',y') - \boldsymbol{I}_R(x'-d,y')|, \tag{4}$$

where $\boldsymbol{N}_{(x,y)}$ is the neighborhood of pixel (x,y), and it is usually a rectangular window with size $W_{SAD} \times H_{SAD}$. The W_{SAD} is the width of $\boldsymbol{N}_{(x,y)}$ and the H_{SAD} is the height.

Another cost is Census, which encodes each pixel into a binary descriptor and calculates the similarity between different pixels by Hamming distance. It encodes the brightness relation between the anchor pixel and its neighbors rather than the brightnesses themselves, so it is robust to the radiometric change. However, it has ambiguities in repetitive and similar structures. The mathematical expressions of it is as Equation (5):

$$C_{\text{Census}}(x,y,d) = \text{HammingDistance}(\rho(\boldsymbol{I}_L(x,y)), \rho(\boldsymbol{I}_R(x-d,y))), \tag{5}$$

in which, $\rho(\boldsymbol{I}(x,y)) = \otimes_{(x',y') \in \boldsymbol{N}_{(x,y)}} \begin{cases} 1 & \boldsymbol{I}(x',y') > \boldsymbol{I}(x,y) \\ 0 & otherwise \end{cases}$, $\otimes$ denotes concatenation and $\boldsymbol{N}_{(x,y)}$ is also the neighborhood of pixel (x,y) with a rectangular window size $W_{Census} \times H_{Census}$. An example of C_{Census} is shown as Figure 2.

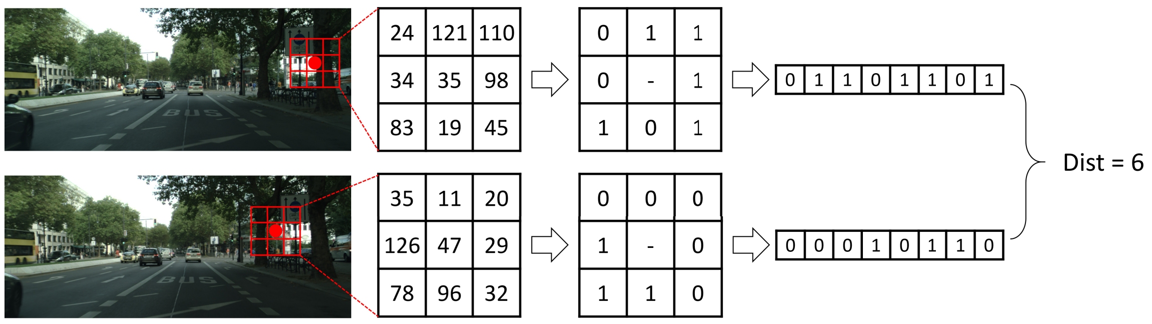

Figure 2. Example of Census cost. The similarity is measured by Hamming distance.

The last cost is the combination of $C_{\text{SAD}}(x,y,d)$ and $C_{\text{Census}}(x,y,d)$ shown as Equation (6), which is similar to the AD-Census Cost in [9]. However, we reduced the number of parameters to be determined, which improves the robustness of our method. Only one constant coefficient α should be given:

$$C_{\text{SADCen}}(x,y,d) = C_{\text{SAD}}(x,y,d) + \alpha \times C_{\text{Census}}(x,y,d). \tag{6}$$

In fact, the SAD cost and Census cost we used in this paper are normalized. With the above costs, the $\{C_{\boldsymbol{L}_i,\boldsymbol{R}_j}\}$ in Equation (1) are prepared. The results of our method using different kinds of costs will be compared in Section 4.

3.3. Adaptive Dynamic Programming

The minimization problem of Equation (1) can be solved by dynamic programming paradigm:

$$OPT(i,j) = \begin{cases} j \times C_{\text{gap}} & \text{if } i = 0, \\ \min \begin{cases} C_{\boldsymbol{L}_i,\boldsymbol{R}_j} + OPT(i-1,j-1) \\ C_{\text{gap}} + OPT(i-1,j) \\ C_{\text{gap}} + OPT(i,j-1) \end{cases} & \text{otherwise}, \\ i \times C_{\text{gap}} & \text{if } j = 0, \end{cases} \tag{7}$$

where $OPT(i,j)$ is the minimum cost of aligning arrays $\{\boldsymbol{L}_1, \boldsymbol{L}_2, ..., \boldsymbol{L}_i\}$ and $\{\boldsymbol{R}_1, \boldsymbol{R}_2, ..., \boldsymbol{R}_j\}$. The $C_{\boldsymbol{L}_i,\boldsymbol{R}_j}$ is calculated as Section 3.2. The C_{gap} is usually an empirical constant, which plays an important role but is hard to choose. When C_{gap} is too large, it will cause more mismatches; and when C_{gap} is too small, it will cause most pixels unmatched. A moderate value is very difficult to select in applications, as shown in Figure 3.

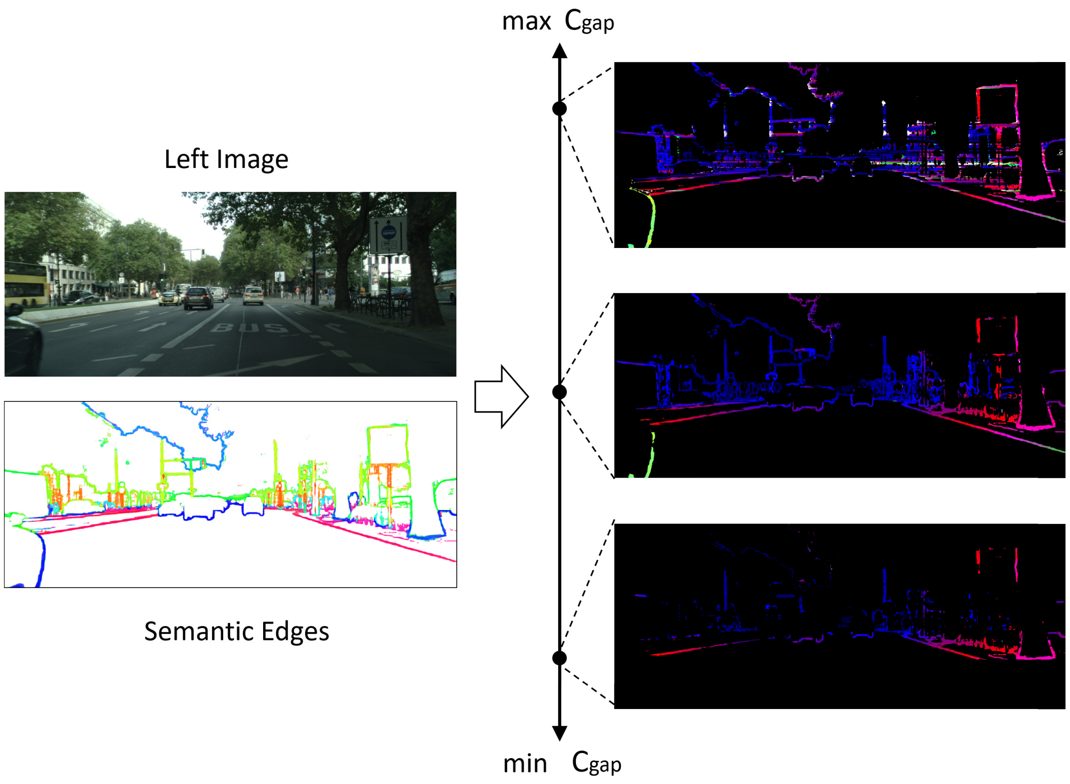

Figure 3. Influence of different C_{gap}s. Too small C_{gap} will cause lots of unmatched pixels, while too large C_{gap} will cause lots of mismatches.

In order to improve the robustness, we propose an improved dynamic programming algorithm, which makes use of the underlying consistency between stereo images to select the value of C_{gap} adaptively. We put forward two kinds of consistency constraints that can be used: disparity consistency constraint and semantic consistency constraint. We also try to utilize them at the same time. However, the experiments in Section 4 show that it is not necessary because the semantic consistency constraint in this case is much stronger than the disparity constraint.

3.3.1. Adaptive Dynamic Programming with Disparity Consistency Constraint

Once the correspondence between two pixels in stereo images is determined, Equation (8) must exist because of the consistency between left and right disparity maps. In stereo matching, it is called LRC (Left-Right Consistency) check and is usually used for mismatch detection:

$$d_L(x,y) = d_R(x - d_L(x,y), y). \tag{8}$$

Actually, this method can not detect all mismatches. Although the probability is low, there still exist consistent errors in left and right disparities. However, it is still the most popular detection method since it is easy to implement and is effective in practice. In this part, we count the number of pixel pairs that satisfy Equation (8) to obtain the value of C_{gap} dynamically, The details are shown in Algorithm 1.

The *MatchingCost* function aims to calculate each similarity between $\boldsymbol{L}_i, i \in \{1, 2, ..., m\}$ and $\boldsymbol{R}_j, j \in \{1, 2, ..., n\}$, so the numbers of items in $\{C_{\boldsymbol{L}_i,\boldsymbol{R}_j}\}$ and $\{C_{\boldsymbol{R}_j,\boldsymbol{L}_i}\}$ are $m \times n$, respectively. In effect, $\{C_{\boldsymbol{R}_j,\boldsymbol{L}_i}\}$ can be obtained from $\{C_{\boldsymbol{L}_i,\boldsymbol{R}_j}\}$ without recalculating. Generating the disparity map of left image is to try to find a matching relationship between $\{\boldsymbol{L}_1, \boldsymbol{L}_2, ..., \boldsymbol{L}_m\}$ and $\{\boldsymbol{R}_1, \boldsymbol{R}_2, ..., \boldsymbol{R}_n\}$, while generating the disparity map of the right image is to try to find a matching relationship between $\{\boldsymbol{R}_n, \boldsymbol{R}_{n-1}, ..., \boldsymbol{R}_1\}$ and $\{\boldsymbol{L}_m, \boldsymbol{L}_{m-1}, ..., \boldsymbol{L}_1\}$. With each image as the base, the coordinate system and the matching order are different. In the case of dynamic programming, different matching orders may result in different results. An example is given in Figure 4.

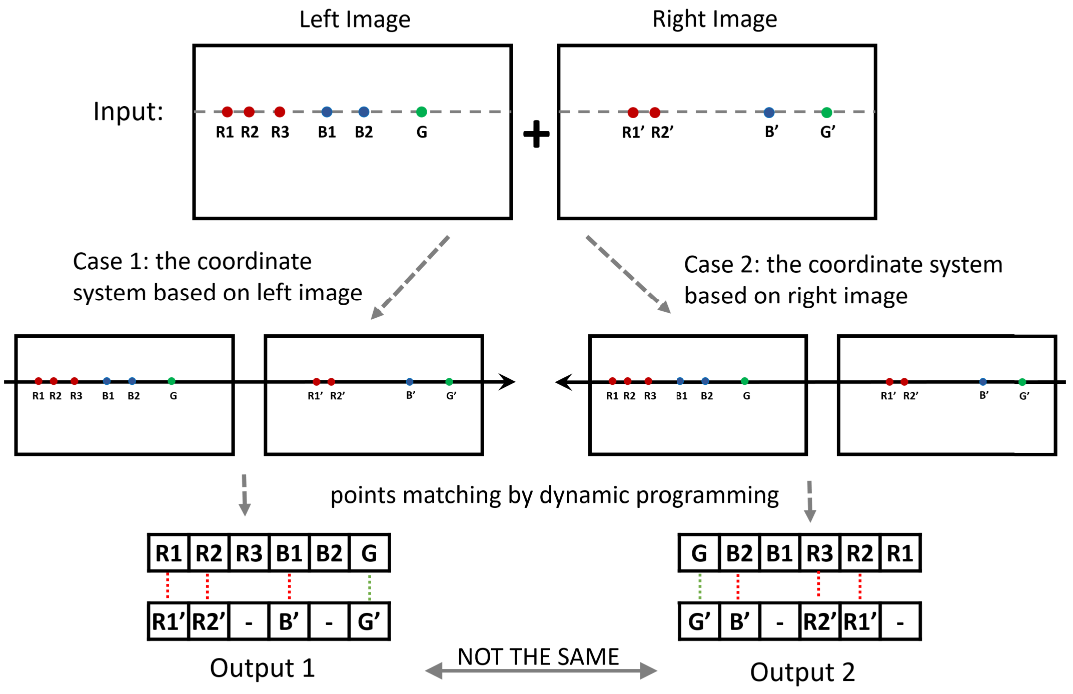

Figure 4. Example of different matching orders. For the same sequences to be matched, the results will be different if the coordinate systems are not the same. The red dotted lines in the results indicate inconsistent pairs on both sides.

Instead of fixing the value of C_{gap}, we provide a range of values for the search of each scan-line's optimal C_{gap}. When the Census cost is chosen, the bottom of the search range (λ_{min}) will be set as 0, the top of the search range (λ_{max}) will be set as $W_{Census} \times H_{Census}$, and the search interval (λ_{inter}) will be set as 1. This is due to the binary descriptor of Census. The maximum, the minimum, and the smallest interval of Census cost are 0, the length of descriptor, and 1, respectively. When the SAD cost is selected, the search interval (λ_{inter}) will be set as 0.01, and if $\{C_{\boldsymbol{L}_i,\boldsymbol{R}_j}\}$ is sorted in ascending order, the λ_{min} and λ_{max} will be set as $C_{\theta_{min}}$ and $C_{\theta_{max}}$ picked from the ordered cost $\{C_1, C_2, ..., C_{m \times n}\}$. The θ_{min} and θ_{max} is computed as Equations (9) and (10) based on empirical conjecture and experimental verification rather than set as the theoretical minimum and maximum, since the distribution of SAD cost is not as predictable as Census cost. Ideally, there should be at most $l_{min} = min\{m, n\}$ values in the cost that are distinct from the other values because there are at most l_{min} pairs of matches. However, in practice, the true match may occur at the second or third smallest cost. Therefore, we set λ_{min} as the $k_1 l_{min}$-th cost in $\{C_1, C_2, ..., C_{m \times n}\}$ and the extent size ($\lambda_{max} - \lambda_{min}$) as $k_2 l_{min}$. $k_1 \in \{1, 2, 3\}$ and $k_2 \in \{1, 2, 3\}$ in this paper:

$$\max \theta_{min} \ s.t. \ \theta_{min} \in \{3 \times l_{min}, 2 \times l_{min}, l_{min}\} \\ \theta_{min} < m \times n, l_{min} = min\{m, n\}, \tag{9}$$

$$\max \theta_{max} \; s.t. \; \theta_{max} - \theta_{min} \in \{3 \times l_{min}, 2 \times l_{min}, l_{min}\} \qquad (10)$$
$$\theta_{max} < m \times n, l_{min} = min\{m, n\}.$$

The *DPMatching* function computes the optimal value of Equation (7) and finds the solution. Since the dynamic programming algorithm is very classic, we will not give details in this paper. The *DisparityComputing* function converts the alignment $\boldsymbol{M_{LR}}$ to disparities $\{d^1_{\boldsymbol{L}}, d^2_{\boldsymbol{L}}, ..., d^p_{\boldsymbol{L}}\}$ with Equation (2), and the disparities $\{d^1_{\boldsymbol{R}}, d^2_{\boldsymbol{R}}, ..., d^q_{\boldsymbol{R}}\}$ can be obtained in a similar way. The *LRC* function eliminates those matches that do not satisfy Equation (8). The disparities $\{d^1_{\boldsymbol{L}}, d^2_{\boldsymbol{L}}, ..., d^k_{\boldsymbol{L}}\}$ that achieve the maximum consistency between left and right arrays are obtained by our algorithm, and the C_{gap} at this time is the optimal value. Hence, one can see that the C_{gap} is self-adapting for each scan-line in different images.

Algorithm 1: Adaptive Dynamic Programming with Disparity Consistency Constraint for Each Pair of Scan-Lines in Stereo Images.

```
1  Function DisparityADPMatching($I_L, I_R, \{L_1, L_2, ..., L_m\}, \{R_1, R_2, ..., R_n\}$)
       // computing the matching cost using equations in Section 3.2
       // $i \in \{1, 2, ..., m\}$, $j \in \{1, 2, ..., n\}$, $\{C_{L_i,R_j}\}$ and $\{C_{R_i,L_j}\}$ have $m \times n$ elements
          respectively
2      $\{C_{L_i,R_j}\}$ = MatchingCost($I_L, I_R, \{L_1, L_2, ..., L_m\}, \{R_1, R_2, ..., R_n\}$);
3      $\{C_{R_j,L_i}\}$ = MatchingCost($I_R, I_L, \{R_n, R_{n-1}, ..., R_1\}, \{L_m, L_{m-1}, ..., L_1\}$);
       // initializing the counting number
4      $N_{max} = 0$;
       // searching for the value of $C_{\text{gap}}$ dynamically
5      for ($C_{gap} = \lambda_{min}; C_{gap} < \lambda_{max}; C_{gap} = C_{gap} + \lambda_{inter}$) do
           // using dynamic programming sloving Equation (7) and returning the
              optimal solution $M$
           // $m$ and $n$ are the length of $\{L_1, ..., L_m\}$, $\{R_1, ..., R_n\}$ respectively
6          $M_{LR}$ = DPMatching($m, n, \{C_{L_i,R_j}\}, C_{gap}$);
7          $M_{RL}$ = DPMatching($m, n, \{C_{R_j,L_i}\}, C_{gap}$);
           // computing the disparities by Equation (2)
8          $\{d^1_L, d^2_L, ..., d^p_L\}$ = DisparityComputing($M_{LR}$);
9          $\{d^1_R, d^2_R, ..., d^q_R\}$ = DisparityComputing($M_{RL}$);
           // removing the pairs that do not satisfy Equation (8)
10         $\{d^1_L, d^2_L, ..., d^k_L\}$= LRC($\{d^1_L, d^2_L, ..., d^p_L\}, \{d^1_R, d^2_R, ..., d^q_R\}, \{L_1, L_2, ..., L_m\}$);
           // $k$ is the number of pairs that satisfy Equation (8)
11         if $N_{max} < k$ then
12             $N_{max} = k$;
13             $D_L = \{d^1_L, d^2_L, ..., d^k_L\}$;
14         end
15     end
17     return $D_L$;
18 end
```

3.3.2. Adaptive Dynamic Programming with Semantic Consistency Constraint

In addition to the disparity consistency mentioned above, there should also exist semantic consistency between the left and right images, shown as Equation (11), since we use the semantic edge pixels as matching candidates. We name this algorithm as Semantic Left-Right Consistency (SLRC) check. Assuming the output of CASENet is E, $E(x, y) = \{0, 1\}^s$, 1 means belonging to a class, and 0 has the opposite meaning. The similarity between different vectors can be measured by Hamming Distance. If only the equality is determined, the vectors can also be converted to decimal digits to compare:

$$E_{\boldsymbol{L}}(x, y) = E_{\boldsymbol{R}}(x - d_{\boldsymbol{L}}(x, y), y). \tag{11}$$

By replacing the disparity consistency constraint with the semantic consistency constraint, Algorithm 2 can be obtained. The functions, which include *MatchingCost*, *DPMatching*, *DisparityComputing*, and the parameters, which include λ_{min}, λ_{max}, λ_{inter} are all the same as the ones in Algorithm 1. Similar to the *LRC* function, the *SLRC* function eliminates those matches that do not satisfy Equation (11). The greatest difference between Algorithms 1 and 2 lies on the right disparity map. Algorithm 2 just needs the left disparity map to determine the semantics of matching pixels. It makes Algorithm 2 faster than Algorithm 1.

Additionally, it is natural to merge the two algorithms using disparity and semantic constraints at the same time, and we did it in Section 4. However, the results are the same as using semantic constraints only in our experiments.

Algorithm 2: Adaptive Dynamic Programming with Semantic Consistency Constraint for Each Pair of Scan-Lines in Stereo Images.

```
1  Function SemanticADPMatching($I_L, I_R, E_L, E_R, \{L_1, L_2, ..., L_m\}, \{R_1, R_2, ..., R_n\}$)
2      $\{C_{L_i,R_j}\}$ = MatchingCost($I_L, I_R, \{L_1, L_2, ..., L_m\}, \{R_1, R_2, ..., R_n\}$);
3      $N_{max} = 0$;
4      for ($C_{gap} = \lambda_{min}; C_{gap} < \lambda_{max}; C_{gap} = C_{gap} + \lambda_{inter}$) do
5          $M_{LR}$ = DPMatching($m, n, \{C_{L_i,R_j}\}, C_{gap}$);
6          $\{d_L^1, d_L^2, ..., d_L^p\}$ = DisparityComputing($M_{LR}$);
           // removing the pairs that do not satisfy Equation (11)
7          $\{d_L^1, d_L^2, ..., d_L^k\}$ = SLRC($E_L, E_R, \{d_L^1, d_L^2, ..., d_L^p\}, \{L_1, L_2, ..., L_m\}$);
8          if $N_{max} < k$ then
9              $N_{max} = k$;
10             $D_L = \{d_L^1, d_L^2, ..., d_L^k\}$;
11         end
12     end
14     return $D_L$;
15 end
```

4. Experiments

In this section, we conduct experiments to study the performance of our method. We will introduce the dataset we used and the evaluation indicators. Then, we will discuss the performance of different window sizes, different costs, and our proposed method separately. We implemented our algorithms on the MATLAB platform with a laptop of Intel Core-i7 CPU (4 cores @2.20 GHz) and 16 GB RAM. No parallel technology is used to accelerate our algorithms.

4.1. Dataset and Evaluation Methods

In existing public datasets, there is no one that has plenty of semantic annotations for outdoors and dense enough ground truth of disparities at the same time. It has caused some difficulties for our quantitative evaluation. The candidate datasets considering by us include KITTI Stereo [25,26] and Cityscapes [27]. Both of them are captured by driving cars around the cities and contain the stereo images. KITTI provides the disparity ground truth by a Velodyne laser scanner and a GPS localization system [25]. However, the ground truth does not cover each whole image. The upper portion of the image has no ground truth, and the ground truth of the lower part is interlaced. The results of our method are sparse for the entire images. There are very small overlaps between our results and ground truth. In addition, KITTI has few semantic annotations provided by other researchers, like [28,29]. Most of these annotations are not in high quality. Cityscapes is a large-scale dataset with high quality annotations of 5000 frames and weakly annotated of 20,000 frames [27]. The output of CASENet on it is more reliable, which is the most important premise of our algorithm implementation. Although this dataset contains no disparity ground truth, we can make the ground truth using reliable feature points matching. As a consequence, we chose Cityscapes to evaluate our experiments. More specifically, we pick out the Berlin data of 544 frames from Cityscapes as our evaluation dataset.

We used a variety of local feature extraction methods for interest points detecting, extracting, and matching. SURF (Speed Up Robust Feature) [30], Harris [31], FAST (Features from Accelerated Segment Test) [32], and BRISK (Binary Robust Invariant Scalable Keypoints) [33] are all taken into account. Then, we applied GMS (Grid-based Motion Statistics) [34] to distinguish inliers from outliers quickly. We can get a set of ground truth, in which there are about 70,000 truth values for each image pairs in our dataset. Some examples are shown in Figure 5. With the ground truth made by us, some quantitative evaluation can be done for reference. We compute the following two measures in the remaining part of this paper:

$$Density = \frac{N_{\boldsymbol{D}}}{N_{\boldsymbol{I}}}, \tag{12}$$

where N_D is the number of the pixels with values in disparity maps, and N_I is the number of each whole image's pixels. In our experiments, N_I is equal to 825×2009:

$$Error = \frac{\sum_{(x,y)\in \boldsymbol{D}\cap \boldsymbol{GT}}(|d(x,y) - d_{\boldsymbol{GT}}(x,y)|) > \delta_d}{N_{\boldsymbol{D}\cap \boldsymbol{GT}}}, \tag{13}$$

where $N_{\boldsymbol{D}\cap \boldsymbol{GT}}$ is the number of the pixels that have values in disparity maps and ground truth maps simultaneously; δ_d is an error tolerance. We set the δ_d as 3, 2, and 1, respectively.

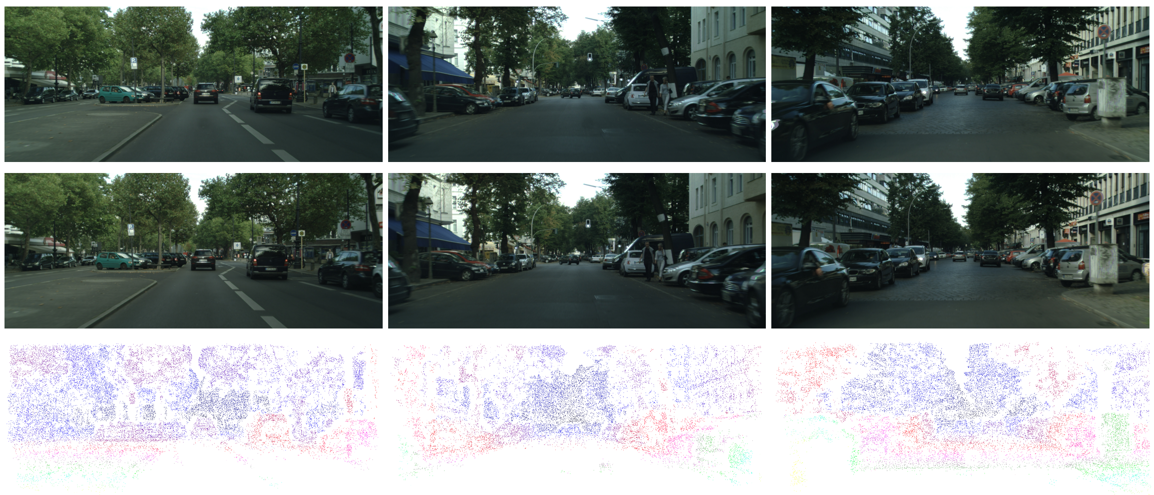

Figure 5. Examples of our ground truth. From top to bottom: the left images, the right images, and our ground truth.

4.2. Performance of Different Window Sizes

The costs used in this paper are simple local ones, and there is no cost aggregation to enhance the global information. In order to ensure that each candidate can be characterized well, it is crucial to choose the appropriate window size. If the window is too small, it is not enough to support each pixel; if it is too big, the calculation time will be increased. Therefore, we compare the effects of different window sizes through experiments.

We use SAD cost and our proposed ADP algorithm with SLRC in this subsection. The corresponding performance indicators are recorded by changing the window size. To facilitate the experiments, we set the height H_{SAD} and width W_{SAD} to the same value $wSize$. The results are collected in Figure 6. Before $wSize$ reaches 15, $Density$ is gradually increasing and $Error$s are decreasing with the increase of $wSize$. After that, $Density$ is still increasing, but $Error$s are increasing slightly. Therefore, we set the H_{SAD} and W_{SAD} as 15 in our next experiments. Actually, the H_{Census} and W_{Census} are set as 15 too.

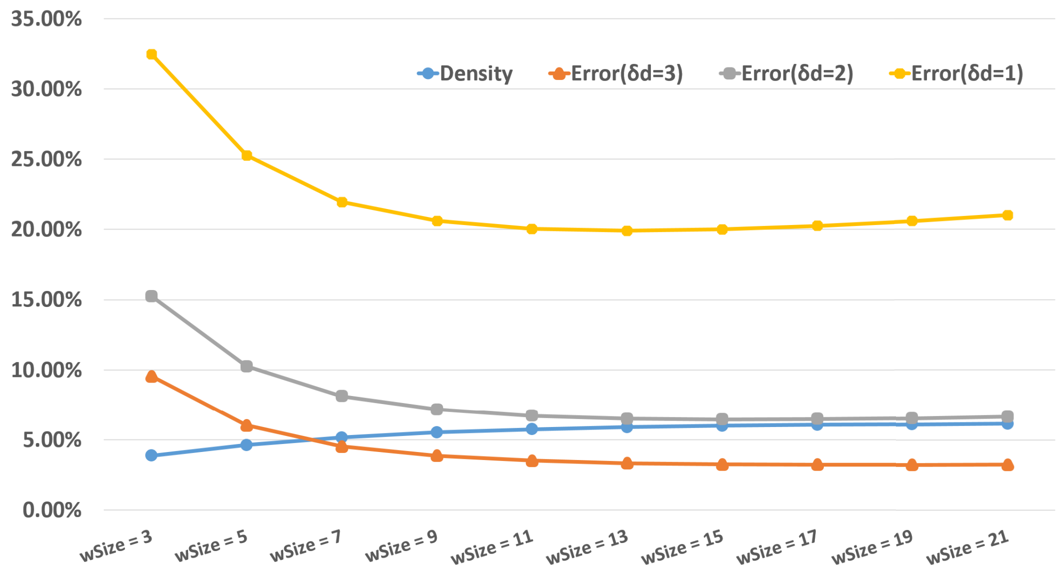

Figure 6. Effects of different window sizes.

4.3. *Performance of Adaptive Dynamic Programming*

As previously mentioned, to determine the value of C_{gap} is a key challenge in the dynamic programming algorithm. C_{gap}s that are too large or too small will result in unsatisfactory results (see in Figures 3 and 7). To prove the validity of our proposed algorithm, we assign several values to C_{gap} to compare with our adaptive algorithms. The cost we used in this subsection is still SAD.

Table 1 details the comparisons. When C_{gap} is equal to 0.0005, the $Density$ is very low. It means that 0.0005 is so small that many pixels cannot find correspondences. Although the $Error$s are low in this case, the results can not be used in practice. When C_{gap} is equal to 0.1 or 0.9, the $Density$ is high, but the $Error$s are high too. Our proposed adaptive programming algorithm, no matter which constraint is used, can guarantee a relatively high $Density$ and relatively low $Error$s. The algorithm with disparity constraint can obtain denser results, while the algorithm with semantic constraint can get higher accuracy. The results of the algorithms with semantic constraint and combined constraint are exactly the same. We will not consider the algorithm with combined constraints in the later experiments. From Table 1, we can also see that the iterative step of our proposed adaptive programming increases the runtime. As we inferred in the previous section, our proposed ADP algorithm with SLRC consumes less time than with LRC.

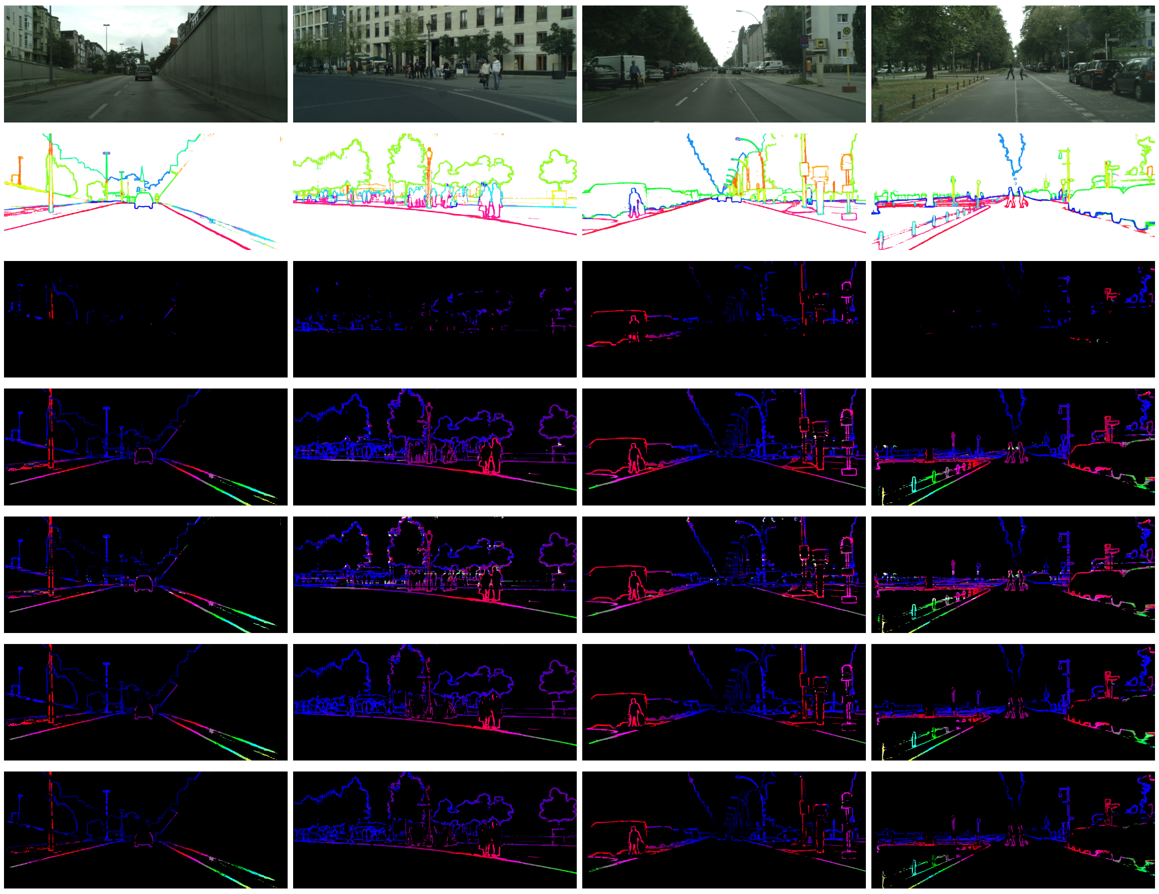

Figure 7. Examples of comparison between our method and the traditional dynamic programming algorithm. From top to bottom: the left images, the semantic edges, the disparity maps of the following algorithms separately: DP($C_{gap} = 0.0005$), DP($C_{gap} = 0.1$), DP($C_{gap} = 0.9$), ADP(LRC), ADP(SLRC).

Table 1. Comparison between our method and the traditional dynamic programming algorithm.

Algorithm	***Density***	***Error***$(\delta_d = 3)$	***Error***$(\delta_d = 2)$	***Error***$(\delta_d = 1)$	**Average Time (s)**
SAD + DP ($C_{gap} = 0.0005$)	2.29%	3.29%	**6.26%**	**19.18%**	27.89 (Matlab)
SAD + DP ($C_{gap} = 0.1$)	**10.14%**	14.86%	22.18%	41.35%	**23.62** (Matlab)
SAD + DP ($C_{gap} = 0.9$)	9.39%	37.61%	47.18%	64.25%	24.52 (Matlab)
SAD + ADP (LRC)	7.53%	3.35%	6.57%	20.12%	47.15 (Matlab)
SAD + ADP (SLRC)	6.01%	**3.26%**	6.47%	20.06%	30.40 (Matlab)
SAD + ADP (LRC + SLRC)	6.01%	**3.26%**	6.47%	20.06%	47.98 (Matlab)

4.4. Performance of Our Proposed Method

In this subsection, we will try the combinations of different costs and adaptive dynamic programming algorithms with different constraints, and then compare our results with other methods.

The six combinations of our method are shown in Table 2 and Figure 9. As you can see, the accuracy of using SAD cost is superior to that of using Census cost when the same ADP algorithm is employed. The combined cost has the best performance by absorbing the advantages of both. In the case of using the same kind of cost, the accuracy of ADP with SLRC is higher than that of ADP with LRC. The advantages of SLRC are especially prominent with Census cost. The average computation time of ADP with Census cost is much higher than others. One reason is the slow calculation of Census cost, and the other one is the wide range of C_{gap} when Census cost is used.

In addition to comparing with the traditional dynamic programming algorithm, we will make further efforts to compare our method with other ones in order to demonstrate the effectiveness of our method. So far as we know, there is no one only matching the extracted edges of CASENet as we did in this paper, which makes difficulties for our comparison. For this, we have developed two schemes as below.

The first one is to compare with other dense matching methods, like [16] did. We extract edge parts from the whole disparity maps and only evaluate them. The extracted disparities are denser than our results and can be compared with ours in both qualitative and quantitative evaluation. We contrast our method with SGM [3] using the codes provided in [5]. Different costs are used, and there are two options for SGM: with refinement or without. The refinement strategy also refers interpolation steps in [5]. We also compare our method with MeshStereo [35], ELAS (Efficient LArge-scale Stereo) [36], SPSStereo [37] and MC-CNN-fast [5] with the codes provided by their authors. Results are shown in Table 2 and Figure 8. The best results of our method are superior to others'. Although the running time of our method is at a disadvantage, it can be improved with code optimization and parallel computation.

The other one is to compare with SED (Simultaneous Edge Drawing) [19], another sparse matching method based on edges. The *Density* of SED is 2.78% only. The intersection of SED's results and our ground truth is very small, so we can not make quantitative evaluation directly. Inspired by the evaluation method of the KITTI stereo dataset [25,26], the initial disparity maps of SED are interpolated according to the interpolation method provided by KITTI. The interpolated dense disparity maps are then evaluated according to the comparison method mentioned above. The results of contrast are shown in Table 3. The qualitative results are shown in Figure 9 for reference, and you can see many obvious mistakes before interpolation. The contours of our method are more clear, and the errors are sporadic.

Table 2. Comparison between our method and other methods.

Algorithm	*Density*	$Error(\delta_d = 3)$	$Error(\delta_d = 2)$	$Error(\delta_d = 1)$	**Average Time (s)**
SAD + SGM	11.24%	2.90%	5.75%	17.87%	20.42 (GPU)
SAD + SGM + Refinement	11.25%	3.02%	5.89%	18.06%	22.65 (GPU)
Census + SGM	11.23%	3.81%	7.77%	22.85%	20.42 (GPU)
Census + SGM + Refinement	11.24%	3.56%	7.16%	21.97%	22.65 (GPU)
MeshStereo [35]	11.69%	6.88%	11.02%	26.35%	19.81 (C++)
ELAS [36]	10.35%	3.22%	7.42%	29.46%	**0.97** (C++)
SPSStereo [37]	**11.72%**	3.17%	7.44%	28.04%	9.11 (C++)
MC-CNN-fast [5]	**11.72%**	2.79%	6.11%	22.05%	11.61 (GPU)
SAD + ADP (LRC)	7.53%	3.35%	6.57%	20.12%	47.15 (Matlab)
SAD + ADP (SLRC)	6.01%	3.26%	6.47%	20.06%	30.40 (Matlab)
Census + ADP (LRC)	10.33%	12.60%	17.85%	34.22%	1198 (Matlab)
Census + ADP (SLRC)	5.85%	5.47%	8.89%	20.82%	831.2 (Matlab)
SAD + 0.1Census + ADP (LRC)	8.37%	2.88%	5.58%	17.73%	127.9 (Matlab)
SAD + 0.1Census + ADP (SLRC)	6.61%	**2.75%**	**5.44%**	**17.59%**	47.05 (Matlab)

Table 3. Comparison between our method and SED [19].

Algorithm	*Density*	$Error(\delta_d = 3)$	$Error(\delta_d = 2)$	$Error(\delta_d = 1)$	**Average Time (s)**
SED [19]	2.78%	-	-	-	**2.01 (C++)**
SED after interpolation	11.67%	13.86%	20.96%	39.16%	-
SAD + 0.1Census + ADP (LRC)	8.37%	2.88%	5.58%	17.73%	127.9 (Matlab)
SAD + 0.1Census + ADP (SLRC)	6.61%	**2.75%**	**5.44%**	**17.59%**	47.05 (Matlab)

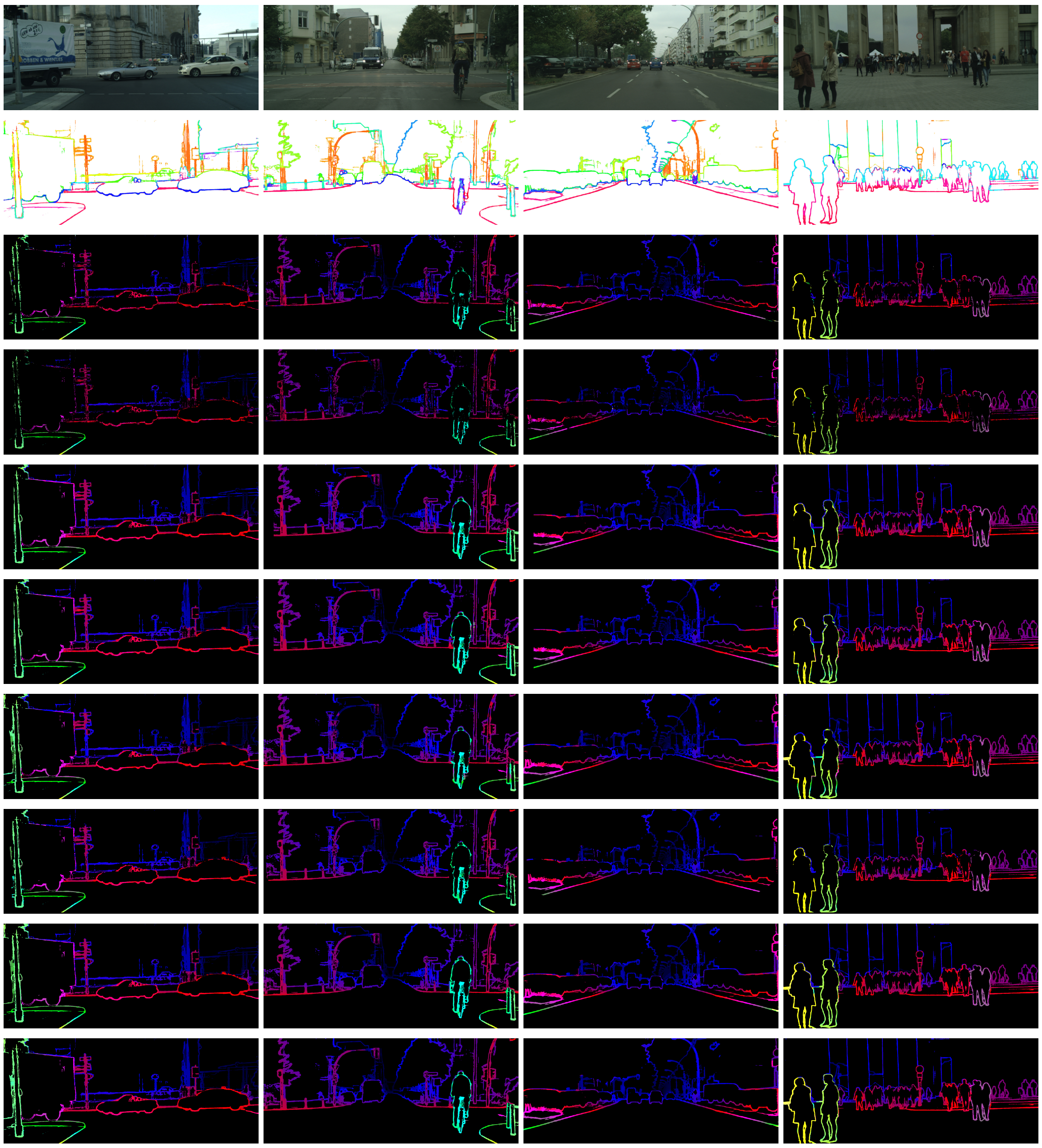

Figure 8. Examples of comparison between our method and other methods. From top to bottom: the left images, the semantic edges, the disparity maps of the following methods separately: SAD + 0.1Census + ADP (LRC), SAD + 0.1Census + ADP (SLRC), SAD + SGM + Refinement, Census + SGM + Refinement, MeshStereo [35], ELAS [36], SPSStereo [37], MC-CNN-fast [5].

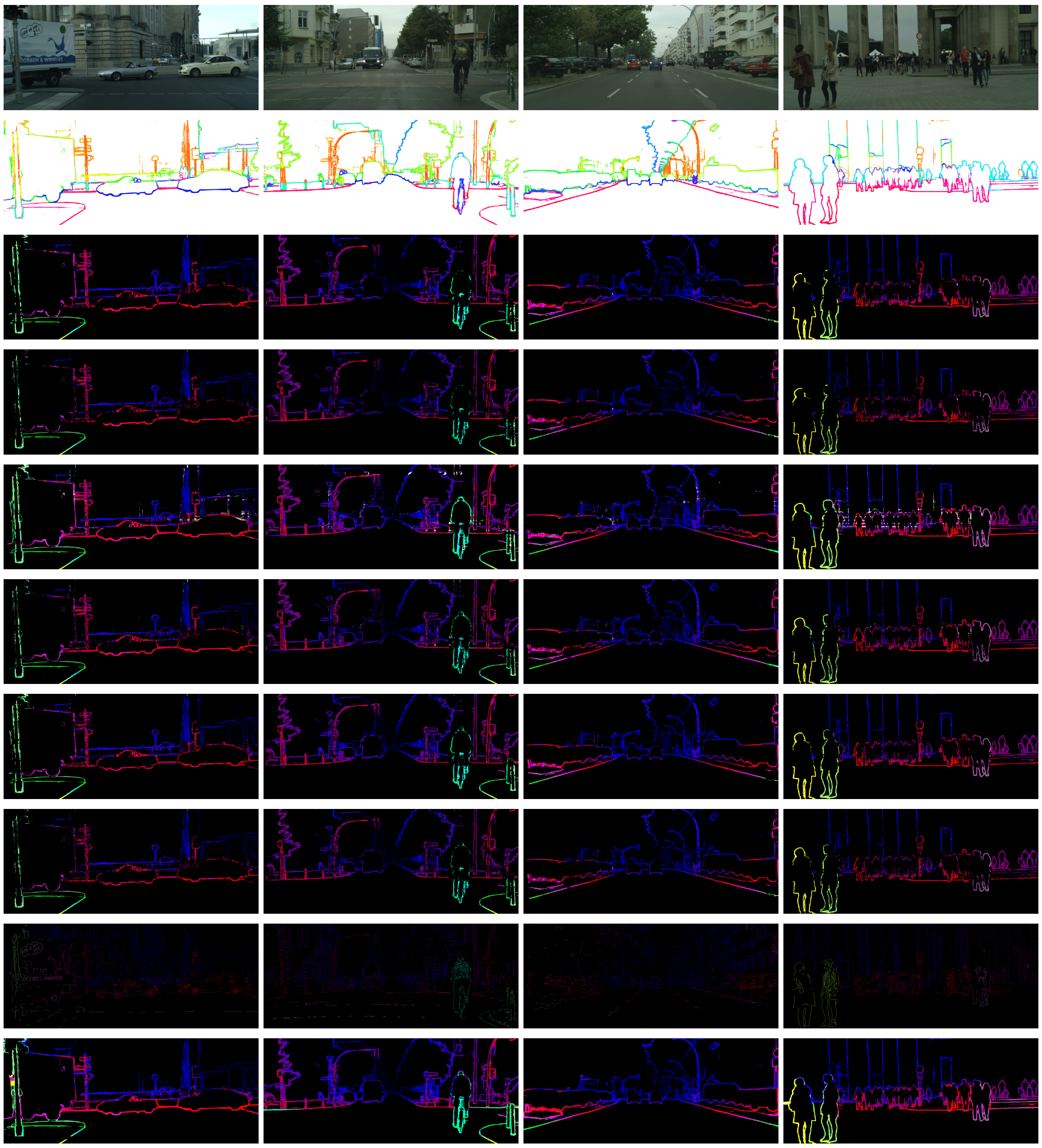

Figure 9. Examples of comparison between our method and SED [19]. From top to bottom: the left images, the semantic edges, the disparity maps of the following methods separately: SAD + ADP (LRC), SAD + ADP (SLRC), Census + ADP (LRC), Census + ADP (SLRC), SAD + 0.1Census + ADP (LRC), SAD + 0.1Census + ADP (SLRC), SED [19], SED after interpolation.

5. Conclusions

In this paper, we present a semantic edge based stereo matching method using an improved adaptive dynamic programming algorithm. The robustness of our method benefits from the reduction of empirical parameters. The single SAD or Census cost can have a considerable effect with our ADP algorithm. When the two costs are combined, we reduce the number of parameters that need to be determined to one. The combined cost can obtain the best results according to our experiments. In ADP, the empirical parameter that is difficult to select in a traditional algorithm is replaced by a range of parameters. The bounds of the range are decided based on the characteristics of the cost by ADP automatically, and the optimal value in the range is obtained based on the disparity or semantic consistency constraint. Different from many other methods, we do not need to set the maximum disparity to improve our performance. The maximum disparity is difficult to estimate in practical applications. Experiments demonstrate that our method offers high accuracy performance and can be put down as one of the top edge-based methods. Furthermore, this study can be extended by considering the constraints between adjacent scan-lines. In our future work, we would like to improve the accuracy of dense stereo matching method with semantic segmentations.

Acknowledgments: This work was supported by the Shanghai Science and Technology Committee, China (Grant No. 17511105803 and Grant No. 16JC1420503), and the National Natural Science Foundation of China (Grant No. 61671014).

Author Contributions: Dongchen Zhu contributed to the theory research, the experiments conception and design; Dongchen Zhu, Xianshun Wang and Jingquan Peng performed the experiments; Dongchen Zhu and Wenjun Shi analyzed the data; Dongchen Zhu and Jiamao Li wrote the paper; Xiaolin Zhang and Jiamao Li contributed to scientific advising and proofreading.

Conflicts of Interest:

Abbreviations

The following abbreviations are used in this manuscript:

AD	Absolute Difference
ADP	Adaptive Dynamic Programming
AR	Augmented Reality
BRISK	Binary Robust Invariant Scalable Keypoints
BP	Belief Propagation
Census	Census Transform
DP	Dynamic Programming
ED	Edge Drawing
ELAS	Efficient LArge-scale Stereo
FAST	Features from Accelerated Segment Test
GC	Graph Cuts
GMS	Grid-Based Motion Statistics
HFL	High-for-Low approach
LRC	Left-Right Consistency
NCC	Normalization Cross Correlation
SAD	the Sum of the Absolute Difference
SED	Simultaneous Edge Drawing
SEM	Semantic Edge Matching
SFM	Structure From Motion
SGM	Semi-Global Matching
SLRC	Semantic Left-Right Consistency
SURF	Speed Up Robust Feature
WTA	Winner-Take-All

References

1. Kolmogorov, V.; Zabih, R. Computing visual correspondence with occlusions using graph cuts. In Proceedings of the Eighth IEEE International Conference on Computer Vision, ICCV 2001, Vancouver, BC, Canada, 7–14 July 2001; Volume 2, pp. 508–515.
2. Boykov, Y.; Veksler, O.; Zabih, R. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.* **2001**, *23*, 1222–1239.
3. Hirschmuller, H. Accurate and efficient stereo processing by semi-global matching and mutual information. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; Volume 2, pp. 807–814.

4. Žbontar, J.; LeCun, Y. Computing the stereo matching cost with a convolutional neural network. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15), Boston, MA, USA, 7–12 June 2015; pp. 1592–1599.
5. Zbontar, J.; LeCun, Y. Stereo matching by training a convolutional neural network to compare image patches. *J. Mach. Learn. Res.* **2016**, *17*, 1–32.
6. Fabbri, R.; Kimia, B. 3D curve sketch: Flexible curve-based stereo reconstruction and calibration. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 1538–1545.
7. Scharstein, D.; Szeliski, R.; Zabih, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001), Kauai, HI, USA, 9–10 December 2001; pp. 131–140.
8. Hirschmuller, H.; Scharstein, D. Evaluation of Stereo Matching Costs on Images with Radiometric Differences. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 1582–1599.
9. Mei, X.; Sun, X.; Zhou, M.; Jiao, S.; Wang, H.; Zhang, X. On building an accurate stereo matching system on graphics hardware. In Proceedings of the 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), Barcelona, Spain, 6–13 November 2011; pp. 467–474.
10. Zhang, K.; Lu, J.; Lafruit, G. Cross-Based Local Stereo Matching Using Orthogonal Integral Images. *IEEE Trans. Circuits Syst. Video Technol.* **2009**, *19*, 1073–1079.
11. Al-Shahri, M.; Yilmaz, A. Line Matching in Wide-Baseline Stereo: A Top-Down Approach. *IEEE Trans. Image Process.* **2014**, *23*, 4199–4210.
12. Hofer, M.; Donoser, M.; Bischof, H. Semi-Global 3D Line Modeling for Incremental Structure-from-Motion. In Proceedings of the British Machine Vision Conference, Nottingham, UK, 1–5 September 2014.
13. Ohta, Y.; Kanade, T. Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming. *IEEE Trans. Pattern Anal. Mach. Intell.* **1985**, *PAMI-7*, 139–154.
14. Sarkis, M.; Diepold, K. Sparse stereo matching using belief propagation. In Proceedings of the 2008 15th IEEE International Conference on Image Processing, San Diego, CA, USA, 12–15 October 2008; pp. 1780–1783.
15. Wei, W.; Ngan, K.N. Disparity estimation with edge-based matching and interpolation. In Proceedings of the 2005 International Symposium on Intelligent Signal Processing and Communication Systems, Hong Kong, China, 13–16 December 2005; pp. 153–156.
16. Witt, J.; Weltin, U. Robust Real-Time Stereo Edge Matching by Confidence-Based Refinement. In *Intelligent Robotics and Applications*; Su, C.Y., Rakheja, S., Liu, H., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 512–522.
17. Witt, J.; Weltin, U. Sparse stereo by edge-based search using dynamic programming. In Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), Tsukuba, Japan, 11–15 November 2012; pp. 3631–3635.
18. Mukherjee, S.; Guddeti, R.M.R. A hybrid algorithm for disparity calculation from sparse disparity estimates based on stereo vision. In Proceedings of the 2014 International Conference on Signal Processing and Communications (SPCOM), Bangalore, India, 22–25 July 2014; pp. 1–6.
19. Peña, D.; Sutherland, A. Disparity Estimation by Simultaneous Edge Drawing. In Proceedings of the Computer Vision—ACCV 2016 Workshops, Taipei, Taiwan, 20–24 November 2016; Chen, C.S., Lu, J., Ma, K.K., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 124–135.
20. Bertasius, G.; Shi, J.; Torresani, L. High-for-Low and Low-for-High: Efficient Boundary Detection from Deep Object Features and Its Applications to High-Level Vision. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 504–512.
21. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *Comput. Sci.* **2014**, arXiv:1409.1556.
22. Yu, Z.; Feng, C.; Liu, M.Y.; Ramalingam, S. CASENet: Deep Category-Aware Semantic Edge Detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1761–1770.
23. Kittler, J. On the accuracy of the Sobel edge detector. *Image Vis. Comput.* **1983**, *1*, 37–42.
24. Canny, J. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *PAMI-8*, 679–698.

25. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 3354–3361.
26. Menze, M.; Geiger, A. Object scene flow for autonomous vehicles. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 3061–3070.
27. Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; Schiele, B. The Cityscapes Dataset for Semantic Urban Scene Understanding. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 3213–3223.
28. Xu, P.; Davoine, F.; Bordes, J.B.; Zhao, H.; Denœux, T. Multimodal information fusion for urban scene understanding. *Mach. Vis. Appl.* **2016**, *27*, 331–349.
29. Sengupta, S.; Greveson, E.; Shahrokni, A.; Torr, P.H.S. Urban 3D semantic modelling using stereo vision. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 580–585.
30. Bay, H.; Tuytelaars, T.; Van Gool, L. SURF: Speeded Up Robust Features. In Proceedings of the Computer Vision—ECCV 2006, Graz, Austria, 7–13 May 2006; Leonardis, A., Bischof, H., Pinz, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 404–417.
31. Harris, C. A combined corner and edge detector. *Proc. Alvey Vis. Conf.* **1988**, *1988*, 147–151.
32. Rosten, E.; Drummond, T. Fusing points and lines for high performance tracking. In Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05), Beijing, China, 17–21 October 2005; Volume 2, pp. 1508–1515.
33. Leutenegger, S.; Chli, M.; Siegwart, R.Y. BRISK: Binary Robust invariant scalable keypoints. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2548–2555.
34. Bian, J.; Lin, W.Y.; Matsushita, Y.; Yeung, S.K.; Nguyen, T.D.; Cheng, M.M. GMS: Grid-Based Motion Statistics for Fast, Ultra-Robust Feature Correspondence. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2828–2837.
35. Zhang, C.; Li, Z.; Cheng, Y.; Cai, R.; Chao, H.; Rui, Y. MeshStereo: A Global Stereo Model with Mesh Alignment Regularization for View Interpolation. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 2057–2065.
36. Geiger, A.; Roser, M.; Urtasun, R. Efficient Large-Scale Stereo Matching. In Proceedings of the Asian Conference on Computer Vision (ACCV), Queenstown, New Zealand, 8–12 November 2010.
37. Yamaguchi, K.; McAllester, D.; Urtasun, R. Efficient Joint Segmentation, Occlusion Labeling, Stereo and Flow Estimation. In Proceedings of the Computer Vision—ECCV 2014, Zurich, Switzerland, 6–12 September 2014; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 756–771.