electronics

MDPI

Article

An Efficient Stereo Matching Network Using Sequential Feature Fusion

Jaecheol Jeong [1], Suyeon Jeon [2] and Yong Seok Heo [1,2,*]

[1] Department of Electrical and Computer Engineering, Ajou University, Suwon 16499, Korea; jcjong1211@ajou.ac.kr
[2] Department of Artificial Intelligence, Ajou University, Suwon 16499, Korea; suyeon1804@ajou.ac.kr
* Correspondence: ysheo@ajou.ac.kr

Abstract: Recent stereo matching networks adopt 4D cost volumes and 3D convolutions for processing those volumes. Although these methods show good performance in terms of accuracy, they have an inherent disadvantage in that they require great deal of computing resources and memory. These requirements limit their applications for mobile environments, which are subject to inherent computing hardware constraints. Both accuracy and consumption of computing resources are important, and improving both at the same time is a non-trivial task. To deal with this problem, we propose a simple yet efficient network, called Sequential Feature Fusion Network (SFFNet) which sequentially generates and processes the cost volume using only 2D convolutions. The main building block of our network is a Sequential Feature Fusion (SFF) module which generates 3D cost volumes to cover a part of the disparity range by shifting and concatenating the target features, and processes the cost volume using 2D convolutions. A series of the SFF modules in our SFFNet are designed to gradually cover the full disparity range. Our method prevents heavy computations and allows for efficient generation of an accurate final disparity map. Various experiments show that our method has an advantage in terms of accuracy versus efficiency compared to other networks.

Keywords: deep learning; stereo matching; disparity estimation; sequential feature fusion

Citation: Jeong, J.; Jeon, S.; Heo, Y.S. An Efficient Stereo Matching Network Using Sequential Feature Fusion. *Electronics* **2021**, *10*, 1045. https://doi.org/10.3390/electronics10091045

Academic Editor: Kiat Seng Yeo

Received: 4 April 2021
Accepted: 25 April 2021
Published: 28 April 2021

1. Introduction

Stereo matching is a fundamental computer vision problem, and has been studied for decades. It aims to estimate the disparity for every pixel in the reference image from a pair of images taken from different points of view. Disparity is the difference in horizontal coordinates between corresponding pixels in the reference and target stereo images. If the pixel (x, y) in the reference left image corresponds to the pixel $(x - d, y)$ in the target right image, the disparity of this pixel is d. Using the disparity value d, focal length f of a camera, and the distance between centers of two cameras B, depth can be obtained by $\frac{fB}{d}$. Stereo matching allows us to obtain 3D information in a relatively inexpensive manner compared to other methods which leverage active 3D sensors [1] such as LiDAR, ToF, and structured light. The importance of stereo matching is recently increasing, because 3D information is required in various emerging applications, including autonomous driving [2], augmented reality [3], virtual reality [4], and robot vision [5].

Like other computer vision problems, much progress in terms of accuracy has been achieved by employing deep learning for stereo matching. Following conventional stereo matching methods [6], the structure of existing deep learning-based methods includes four steps: feature extraction, cost volume construction, cost volume processing (or aggregation), and final disparity (or depth) map estimation. Early approaches [7–9] using deep learning for stereo matching focus on extracting features using convolutional neural network (CNN) and computing similarity scores for a pair of corresponding image patches. Zbontar and LeCun [7] proposed the first deep learning-based stereo matching network which learns to match the corresponding image patches with CNN. Luo et al. [9] also uses CNN

to compute matching costs by using the extracted robust deep features from a Siamese network. These early approaches show significant increase of accuracy compared to the previous conventional methods which use hand-crafted features. However, these approaches have common limitations that high computations are required to forward pass all potentially corresponding patches. In addition, the increase of accuracy from deep learning is limited, because they still use post-processing functions to obtain a final disparity map.

Mayer et al. [10] proposed the DispNet which is the first end-to-end network including feature extraction, cost volume generation, and disparity regression by processing the cost volume. Pang et al. [11] proposed an encoder-decoder network using 2D convolutions with cascaded residual learning. For the cost volume construction, these approaches [10,11] created a 3D cost volume with dimensions of width, height, and disparity range. To this end, the corresponding deep features are processed in a hand-crafted manner such as correlation between features. Then, cost volume processing using 2D convolution is followed to obtain a final disparity map. However, these methods still suffer from lack of context information, because they still use the hand-crafted operation such as correlation or dot-product between corresponding features for the cost volume generation.

To overcome this limitation, most of the latest stereo estimation networks create a 4D cost volume by stacking the corresponding deep features [12,13], instead of relying on the correlations between corresponding features. A typical 4D cost volume has width, height, disparity range, and feature dimensions. Unlike 3D cost volume, more information can be processed because the 4D cost volume maintains feature dimension. This 4D cost volume is processed and regularized using 3D convolutions [12–14]. In addition, the soft argmin function suggested by [12] is fully differentiable and able to predict smooth sub-pixel disparity. These techniques have become mainstream, because they show excellent performance in terms of accuracy compared to previous methods. The gain in accuracy comes from learning the entire process, including cost volume generation and processing, which is not done in the 2D convolution-based methods.

However, most of the 3D convolution-based methods have an inherent disadvantage in that they require consumption of a large amount of computing resources as the number of elements in the dimensions of cost volume increase. For this reason, a 4D cost volume which is stacked over the full disparity range requires a great deal of memory. In addition, 3D convolutions for processing of the cost volume also require sizeable amounts of computation and memory. These requirements limit their applications for mobile environments, which are inherently constrained in terms of computing hardware. However, the number of applications that demand to predict and use depth directly on mobile devices is steadily increasing. Also, in many real-world applications including autonomous driving, augmented reality and robotics, reliable real-time processing is essential. Therefore, many studies are conducted for the efficient stereo matching network that can be used on mobile devices or can be executed in real time with reliable accuracy. Recently, AnyNet [15] is proposed to deal with this problem. It predicts disparity map from the low scale and subsequently correcting it by the residual error at the up-sampled scale. Because Anynet processes a full range of disparities only at the smallest scale and computations for other scales are performed residually, real-time processing with small computation is realized. However, the accuracy of AnyNet [15] is severely decreased compared to other 3D convolution-based methods. Although both accuracy and consumption of computing resources are important, improving both at the same time is a non-trivial task.

To deal with this problem, we propose a simple yet efficient network, called Sequential Feature Fusion Network (SFFNet) which sequentially generates 3D cost volume and processes it using only 2D convolutions. The main building block of our network is a Sequential Feature Fusion (SFF) module which generates 3D cost volumes to cover a part of the disparity range by shifting and concatenating the target features, and processes the cost volume using 2D convolutions. A series of the SFF modules in our SFFNet are designed to gradually cover the full disparity range. Our method prevents heavy computations

and allows for efficient generation of an accurate final disparity map. More specifically, with small complexity and small number of parameters, our proposed network generates comparable results with previous 3D convolution-based methods.

The rest of the paper is organized as follows. Section 2 explains related works, and a detailed explanation of the proposed method follows in Section 3. Various experiments done for the purposes of comparative evaluations are provided in Section 4. Finally, Section 5 concludes the paper.

2. Related Work

2.1. Classical Stereo Matching

Traditional stereo matching essentially consists of four steps: matching cost computation, cost aggregation, disparity computation/optimization, and disparity refinement [6]. These algorithms are divided into global matching methods [16,17] and local matching methods [18–22] according to the optimization method that is used. Although global matching methods usually show higher accuracy than local methods, they are relatively complicated and require a lot of computing resources such as memory. On the other hand, local matching methods have the advantage of being relatively light, but they are less accurate than the global methods. To overcome this limitation, various post-processing methods [23–27] and comprehensive methods [28,29] have been studied. However, these traditional stereo matching algorithms have a common limitation in that their accuracy is good for relatively simple conditions.

2.2. Deep Stereo Matching

Recently, the idea of applying deep learning to stereo matching has been revived. The seminal work of MC-CNN [7] began to establish the basic structure of stereo matching networks. The basic procedure of the classical stereo methods is still reflected in the deep learning-based stereo matching network structure. Some steps in the classical method mentioned above are replaced with convolutional neural networks (CNNs). Previous deep learning-based methods can be categorized into two classes, 2D convolution-based and 3D convolution-based methods, according to the process used for generation and processing of the cost volume. Detailed explanations are given below.

2.2.1. 2D Convolution-Based Methods

Most early works using CNNs for stereo matching are 2D convolution-based methods. These methods leverage CNNs to extract features [9] and/or construct cost volumes and perform matching using 2D convolutions [7,8]. Some of these methods require additional post-processing to obtain the final disparity map. To overcome the drawbacks of these methods, Mayer et al. [10] proposed the first end-to-end network which directly regresses a disparity map by constructing a 3D cost volume using hand-crafted computations such as correlation between corresponding features. CRL [11] improved upon [10] based on cascade residual learning, which refines the initial disparity using residual components across multiple scales. Yang et al. [30] proposed a unified network based on [10] that performs both semantic segmentation and disparity estimation by using semantic features to improve the performance of disparity estimation. Yin et al. [31] proposed a matching network which estimates matching distribution by using feature correlation and composing multiple scale matching densities. Tonioni et al. [32] proposed a fast stereo network to perform effective online adaption.

The above-mentioned methods usually generate a 3D cost volume using the correlations between corresponding features and 2D convolutions for cost volume processing. These methods outperform classical stereo matching methods, such as most deep learning-based computer vision techniques. However, their accuracies are usually not good compared to those of the 3D convolution-based methods which will be described in the following section. Despite their shortcomings, they are often used and studied because of their advantages in terms of computing resources and/or execution time [31,32].

2.2.2. 3D Convolution-Based Methods

The networks based on correlation analysis and 2D convolutions introduced above did not deviate from the existing algorithms in that the matching cost is still generated in a hand-crafted manner. 3D convolution-based methods are designed to transform this step into a learnable form. Instead of constructing a 3D cost volume, GC-Net [12] proposed construction of a 4D cost volume by concatenation of left-right features along the full disparity range. This cost volume is processed using CNNs comprising 3D convolutions with encoder-decoder architectures. Following [12], PSMNet [13] proposed a method of constructing 4D cost volumes using multi-scale features. PSMNet [13] uses a stacked hourglass structure comprised of three encoder-decoder(hourglass) architectures. However, the disadvantage of using 3D convolutions is that doing so significantly increases the consumption of computing resources. To mitigate this increase, building off [12], Lu et al. [33] proposed a method to construct a sparse cost volume with stride to efficiently perform stereo matching. Duggal et al. proposed a deep learning-based matching network with a differentiable patch-match module [14] which prunes out most of the useless disparity range to reduce the complexity of the 3D convolutions. Tulyakov et al. [34] designed a practical network with a smaller memory footprint by compressing the cost volume into compact matching signatures before performing 3D convolution-based regularization.

3. Method

An overview of the proposed network is shown in Figure 1. As in other networks, we generate a feature vector for each pixel using a feature extraction network. Unlike previous works [13,14] which construct a heavy 4D cost volume by stacking corresponding features along the full disparity range and processes the cost volume using 3D convolutions, our method sequentially performs cost volume construction and aggregation using the proposed Sequential Feature Fusion Network (SFFNet). Our SFFNet consists of a sequence of the proposed Sequential Feature Fusion (SFF) modules, where each module is based on the ResNet block structure [35] and Hierarchical Feature Fusion (HFF) [36]. Finally, a refine network is used to further refine the initial disparity map and obtain an accurate final disparity map. The whole structure of our network is summarized in Table 1. Detailed explanations are given in the next subsections.

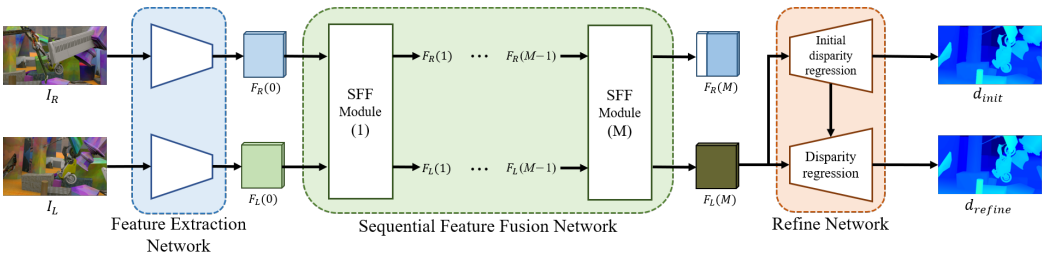


Figure 1. An overview of the proposed method.

3.1. Feature Extraction Network

The feature extraction network extracts a feature representation for each pixel of the input stereo images. Given a pair of stereo images I_L and I_R, features $F_L(0)$ and $F_R(0)$ capable of forming a cost volume are output for each viewpoint. To this end, we employ a 2D convolutional network using the Spatial Pyramid Pooling (SPP) module [37,38], which is similar to [13,14]. By extending pixel-level features to region-level using different pooling sizes, generated features from the SPP module hold incorporated hierarchical context information and it makes feature representations more reliable. The parameters of the feature extraction network of the left and right images are shared. For efficient computation, the size of the output feature map is $1/4$ of the original input image size. This part is commonly used by other networks using 3D convolutions that show the best performance [13,14].

Table 1. Entire architecture.

Name	Layer Definition	Input Dimension	Output Dimension
I_L/I_R		$H \times W \times 3$	
Feature Extraction Network			
I_L /I_R	Input of Network		
$F_L(0)/F_R(0)$	Output of Network		$\frac{H}{4} \times \frac{W}{4} \times 32$
SFF Network (SFF Module $\times$ M)			
$F_L(0)/F_R(0)$	Input of 1^{st} SFF Module		
Cost Volume	concat[$F_L(0)$, $F_R^0(0)$ $\ldots$, $F_R^S(0)$]	$\frac{H}{4} \times \frac{W}{4} \times 32$, $\frac{H}{4} \times \frac{W}{4} \times 32$	$\frac{H}{4} \times \frac{W}{4} \times 128$
branch_1	$\begin{bmatrix} 3\times3, 128, 32 \\ 3\times3, 32, 32 \end{bmatrix}$	$\frac{H}{4} \times \frac{W}{4} \times 128$	$\frac{H}{4} \times \frac{W}{4} \times 32$
branch_2	[1×1, 128, 32]	$\frac{H}{4} \times \frac{W}{4} \times 128$	$\frac{H}{4} \times \frac{W}{4} \times 32$
$F_L(1)$	sum(branch_1, branch_2)		$\frac{H}{4} \times \frac{W}{4} \times 32$
$F_L(1)/F_R(1)$	Output of 1^{st} SFF Module		
Repeat M times.			
$F_L(M)/F_R(M)$	Output of M^{th} SFF Module		
Refine Network			
Initial Disparity Regression			
$F_L(M)$	Input of Initial Disparity Regression		
Init_refine1	$\begin{bmatrix} 1\times1, 32, 16 \\ 1\times1, 16, 1 \end{bmatrix}$	$\frac{H}{4} \times \frac{W}{4} \times 32$	$\frac{H}{4} \times \frac{W}{4} \times 1$
Init_refine2	bilinear interpolation [5× 5, 1, 1]	$\frac{H}{4} \times \frac{W}{4} \times 1$	$\frac{H}{2} \times \frac{W}{2} \times 1$
Init_refine3	bilinear interpolation [5× 5, 1, 1]	$\frac{H}{2} \times \frac{W}{2} \times 1$	$H \times W \times 1$
d_{init}	Output of Initial Disparity Regression		$H \times W \times 1$
Disparity regression			
$F_L(M)$ Init_refine2	Input of Disparity Regression		
Disp_refine1	bilinear interpolation [5× 5, 32, 32]	$\frac{H}{4} \times \frac{W}{4} \times 32$	$\frac{H}{2} \times \frac{W}{2} \times 32$
concat[Disp_refine1, Init_refine2]			$\frac{H}{2} \times \frac{W}{2} \times 33$
refinement	[3× 3, 33, 32] [3× 3, 32, 32] × 2 [3× 3, 32, 16] [3× 3, 16, 16] × 2 [1× 1, 16, 1]	$\frac{H}{2} \times \frac{W}{2} \times 33$	$\frac{H}{2} \times \frac{W}{2} \times 1$
sum(refinement, Init_refine2)			$\frac{H}{2} \times \frac{W}{2} \times 1$
Disp_refine2	bilinear interpolation [5× 5, 1, 1]	$\frac{H}{2} \times \frac{W}{2} \times 1$	$H \times W \times 1$
d_{refine}	Output of Disparity Regression		$H \times W \times 1$

3.2. Sequential Feature Fusion Network (SFFNet)

Figure 2 shows the proposed Sequential Feature Fusion Network (SFFNet) which consists of a series of M SFF modules. The first SFF module takes $F_L(0)$ and $F_R(0)$ from the feature extraction network as input. In addition, the following output of the n^{th} SFF module serves as the input of the next $n+1^{th}$ SFF module. Only $F_L(M)$ from the final SFF module is used in the refine network to produce final disparity map. A single SFF module combines cost volume generation and aggregation for a part of full disparity range using only 2D convolutions. Our SFFNet is motivated by the Hierarchical Feature Fusion (HFF) [36] method used in semantic segmentation. HFF produces a feature map that covers large receptive field without directly performing original convolutions with large sizes. Instead, it hierarchically adds intermediate features with different small receptive fields before concatenating them. We adopt this idea for stereo matching, which processes full range of disparities by connecting modules which processes only a subset of disparity ranges. It is worthy to note that the purpose of the HFF is for efficiently obtaining the feature map with large receptive field in the spatial domain. Meanwhile, the purpose of our SFFNet is to efficiently enlarge the receptive field in the disparity domain.

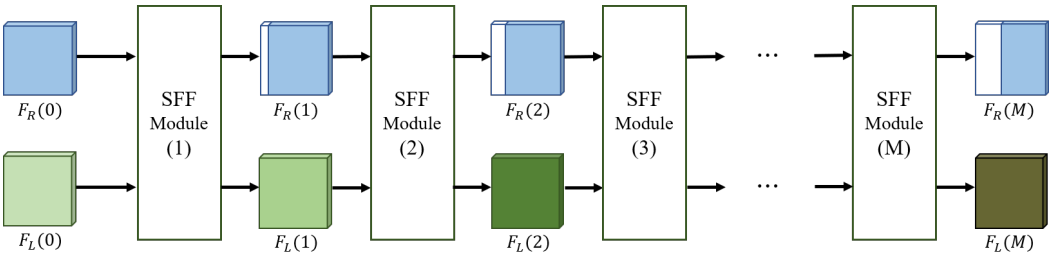


Figure 2. Connections between SFF modules in the SFFNet.

Specifically, the n^{th} SFF module deals with the disparity range $[(n-1)S, nS]$, where S represents a specific disparity range which is processed at a single SFF module. As shown in Figure 3, the $n+1^{th}$ SFF module generates output feature maps $F_L(n+1)$ and $F_R(n+1)$ from input feature maps $F_L(n)$ and $F_R(n)$. Here, $F_L(n+1)$ and $F_R(n+1)$ are defined by

$$\begin{aligned} F_L(n+1) &= f(F_L^+(n)), \\ F_R(n+1) &= F_R^S(n), \end{aligned} \tag{1}$$

where $F_L^+(n)$ is the result of concatenation of various features of the reference (left) and target (right) images, and is defined by

$$F_L^+(n) = F_L(n) \circ F_R^0(n) \circ F_R^1(n) \circ \cdots \circ F_R^S(n), \tag{2}$$

where $\circ$ represents the concatenation operation, and $F_R^i(n)$ denotes the feature that is shifted from the original feature $F_R(n)$ by i pixels in the width direction. Function $f(\cdot)$ in Equation (1) includes sum of results from two 3×3 2D convolutions and one 1×1 2D convolution, as shown in Figure 3. Two 3×3 convolutions are used to increase the receptive field, while one 1×1 convolution plays a role of the projection shortcut [35] to form a residual function.

After the $n+1^{th}$ SFF module, a cumulative cost volume for the disparity range $[0, (n+1)S]$ is generated. At the same time, the learning area for disparity of S pixels is widened while processing it using a series of SFF modules. Concretely, $F_L(n+1)$ contains the processed and aggregated cost volume of the reference image for a disparity range of $[0, (n+1)S]$, while $F_R(n+1)$ is the feature map of the target image shifted by $(n+1)S$ pixels for processing next $(n+2)^{th}$ SFF module.

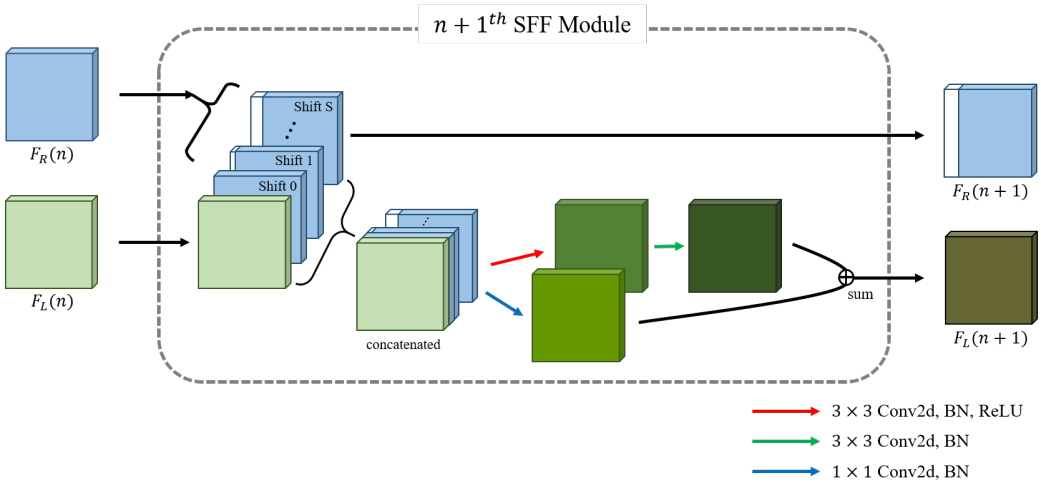


Figure 3. SFF module.

Please note that unlike previous 3D convolution-based approaches which generate a 4D cost volume covering a full disparity range and aggregate it using 3D convolutions in a separate process, our SFFNet simultaneously performs both generating and aggregating cost volume, and gradually increases the range of disparity search. The proposed SFFNet adjusts the full disparity range R through the number of SFF modules M and the number of shifts S, as follows:

$$R = S \times M. \tag{3}$$

Although a large S value allows the network to learn a wide range of disparities in a single SFF module, the disparities cannot be learned in detail in the module. Meanwhile, the number of connections M controls the depth of network, and a high value of M can slow the runtime.

3.3. Refine Network and Loss Function

The feature map $F_L(M)$ generated through the SFFNet is further processed using a light refine network similar to [14] to generate a final disparity map. As shown in Figure 1, the refine network takes $F_L(M)$ obtained from the final M^{th} SFF module in the SFFNet and generates an initial disparity map d_{init} as well as a final disparity map d_{refine}. Use of both the initial disparity map d_{init} and the processed feature maps $F_L(M)$ allows the refine network to focus only on the residual component of the initial disparity map and to improve the quality of the final disparity map d_{refine}. Here, the initial disparity is simply generated by processing the feature map $F_L(M)$ from the SFFNet through the 1×1 convolutional network [39] and bilinear upsampling. Final refined disparity map is generated using the processed feature map $F_L(M)$ and the middle feature map obtained from initial disparity processing. This process is composed of 5×5 convolutional layer and bilinear upsampling.

Now, the total loss function L used to learn the disparity map is defined by

$$L = \gamma_1 V_s(d_{init} - d_{gt}) \; + \; \gamma_2 V_s(d_{refine} - d_{gt}), \tag{4}$$

where d_{init} and d_{refine} denote the initial disparity map and the final disparity map, respectively, and d_{gt} is a ground-truth disparity map. Here, the smoothness L1 loss function $V_s(\cdot)$ [40] is defined by

$$V_s(x) = \begin{cases} 0.5x^2 & if|x| \leq 1 \\ |x| - 0.5 & otherwise \end{cases}. \tag{5}$$

The values of γ_1 and γ_2 in Equation (4) represent the weight of the loss of the initial disparity map and that of the final disparity map in the total loss function, respectively.

4. Experimental Results

We evaluate our network on several datasets and demonstrate that the proposed SFFNet achieves better results in terms of consumption of computing resources vs. accuracy compared to the other methods. For purposes of comparison, we designed all experiments under the same conditions. Also, the training datasets, the maximum disparity range and all evaluation indicators for each network are the same. Next, we describe the experimental setup for each dataset, and then explain the performance using various evaluation indicators.

4.1. Datasets

We conducted experiments on two datasets.

1. Scene Flow [10]: A synthetic stereo dataset which includes ground-truth disparity for each viewpoint generated using computer graphics. It contains 35,454 training and 4370 testing image pairs with $H = 540$ and $W = 960$. EPE (End-Point-Error) was used as an evaluation metric to evaluate the results, where the EPE is defined by the average difference of the predicted disparities and their true ones.
2. KITTI [41,42]: A dataset based on actual images (not synthesized images). The KITTI-2015 version contains training and test sets, each of which have 200 image pairs. The KITTI-2012 version contains 194 image pairs for training and 195 image pairs for testing. The image size is $H = 376$ and $W = 1240$ for both versions. We trained and tested using only the training dataset, which includes ground-truth disparity information. 354 random image pairs from the 2015 and 2012 version training sets were used as the training dataset. For the evaluation indicator, we used the 3-pixel-error (3PE) provided by the benchmark dataset [41,42]. The 3PE represents the percentage of pixels for which the difference between the predicted disparity and the true one is more than 3 pixels.

4.2. Implementation Details

We trained our network on the Scene Flow dataset and the KITTI training dataset. Input images from these two datasets are randomly cropped with size of $H = 256$ and $W = 512$, and then normalized using the ImageNet [43] statistics (mean : $[0.485, 0.456, 0.406]$, std : $[0.229, 0.224, 0.225]$) at the pre-processing step, similar to [13,14]. Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$) [44] was used as an optimization method for end-to-end training. We implemented our model using PyTorch [45] in Unbuntu 16.04 OS with CUDA version 10.1 with 4 Nvidia Titan-XP GPUs. The hyperparameters for the loss function in Equation (4) were set as $\gamma_1 = 1$ and $\gamma_2 = 1.3$, so that more weight was given to the final result of our network similar to [13,14]. To create the same conditions as used for the other networks, the loss was calculated only for pixels with ground-truth disparity value in the range of 0 to 192. The numbers of S and M in Equation (3) are set as $S = 2$ and $M = 24$ which cover the full disparity range of 192 for $1/4$ of the input image size.

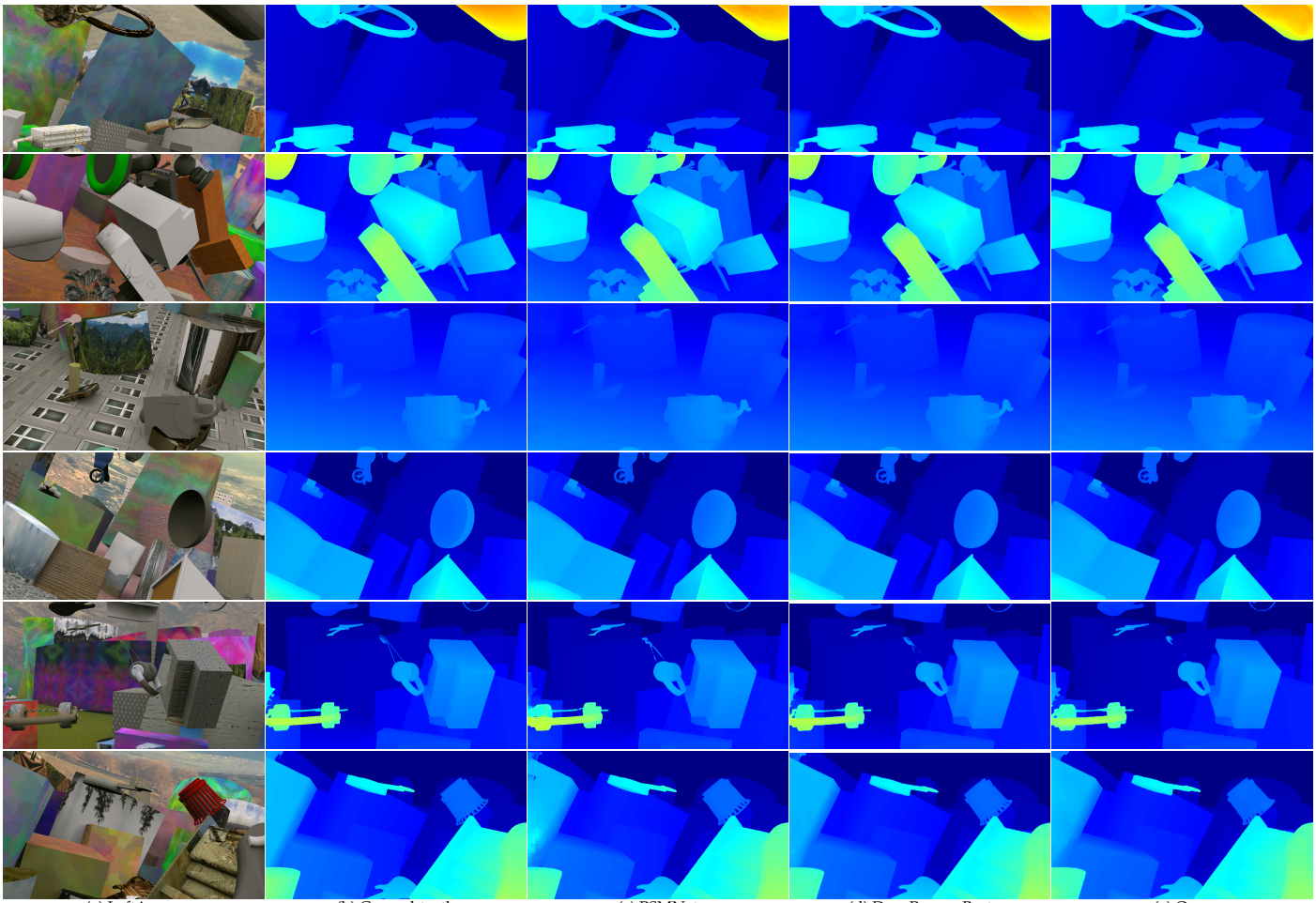
Training was done for a total of 678 epochs on the Scene Flow dataset, with a batch size of 44 and a learning rate of 0.001; the learning rate was re-adjusted to 0.0007, 0.0003, 0.0001, and 0.00007 at epochs 20, 40, 60 and 600, respectively. In the case of the KITTI dataset, the network trained through the Scene Flow dataset was transferred. Concretely, the batch size was 22 and the learning rate was set to 0.0007 and re-adjusted to 0.00004 and 0.00001 at epochs 200 and 900, respectively. We empirically determined these optimal learning rates and number of epochs for training.

4.3. Results and Analysis on the Scene Flow Dataset

Table 2 shows the comparative results of various methods using the test set of the Scene Flow dataset. "Ours (Initial)" represents the result using the initial disparity map of the proposed network without the refine network. "Ours" denotes the result of the proposed network with the refine network. Here, we compare our results with those of other recent 3D convolution-based networks.

Table 3 further compares the runtime and EPE of the top-performing 3D convolution-based methods. For a fair comparison, the same feature extraction network is used for all methods. The results show that our SFFNet achieves lower EPE with lower runtime than PSMNet [13]. The runtime of our network is 2.8 times faster than that of DeepPruner-Best [14], while EPE of ours is 1.2 times higher. These results show that SFFNet is more efficient than other 3D convolution-based cost aggregation networks.

Figure 4 shows a qualitative comparison of our method and others on the Scene Flow test set. Our method generates results that are comparable to those of other state-of-the-art methods [13,14] for most regions, including sharp boundaries and textureless regions.



(a) Left image (b) Ground-truth (c) PSMNet (d) DeepPruner-Best (e) Ours

Figure 4. Comparisons of disparity maps on the Scene Flow test set.

Table 2. Quantitative comparison results on the Scene Flow test set.

GC-Net [12]	SegStereo [30]	CRL [11]	PDS-Net [34]	PSM-Net [13]	DeepPruner-Best [14]	DeepPruner-Fast [14]	DispNetC [10]	Ours (Initial)	Ours
2.51	1.45	1.32	1.12	1.09	0.86	0.97	1.68	1.19	1.04

Table 3. Comparison results of runtime and EPE on the Scene Flow test set. *SPP: Spatial Pyramid Pooling, *PM: Patch Match, *CRP: confidence range predictor, *CA: cost aggregation.

Model	Feature Extraction	Network Component					Runtime	EPE
PSMNet [13]	CNNs with SPP	Stacked Hourglass					379 ms	1.09
DeepPruner-Best [14]		PM-1	CRP	PM-2	CA	RefineNet	128 ms	0.858
Ours		SFFNet - RefineNet					45 ms	1.04

4.4. Results and Analysis on the KITTI-2015 Dataset

Table 4 shows comparison results for various indicators including runtime, error ratio, number of parameters, and FLOPs of competing algorithms on the KITTI-2015 stereo benchmark [42]. Here, the percentages of erroneous pixels in terms of 3PE averaged over the background (bg) and foreground (fg) regions and all ground-truth pixels (all) are measured separately. Noc (%) and All (%) represent the percentages of erroneous pixels for only non-occluded regions and for all pixels, respectively.

Table 4. Evaluation results on the KITTI 2015 test set.

Method	Network	Runtime	Noc(%)			All(%)			Params	FLOPs
			bg	fg	all	bg	fg	all		
3D conv. method	Content-CNN [9]	1000 ms	3.32	7.44	4.00	3.73	8.58	4.54	0.70 M	978.19 G
	MC-CNN [7]	67,000 ms	2.48	7.64	3.33	2.89	8.88	3.89	0.15 M	526.28 G
	GC-Net [12]	900 ms	2.02	3.12	2.45	2.21	6.16	2.87	2.86 M	2510.96 G
	CRL [11]	470 ms	2.32	3.68	2.36	2.48	3.59	2.67	78.21 M	185.85 G
	PDS-Net [34]	500 ms	2.09	3.68	2.36	2.29	4.05	2.58	2.22 M	436.46 G
	PSM-Net [13]	410 ms	1.71	4.31	2.14	1.86	4.62	2.32	5.36 M	761.57 G
	SegStereo [30]	600 ms	1.76	3.70	2.08	1.88	4.07	2.25	28.12 M	30.50 G
	EdgeStereo [46]	700 ms	1.72	3.41	2.00	1.87	3.61	2.16	-	-
	DeepPruner-Best [14]	182 ms	1.71	3.18	1.95	1.87	3.56	2.15	7.39 M	383.49 G
	DeepPruner-Fast [14]	64 ms	2.13	3.43	2.35	2.32	3.91	2.59	7.47 M	153.77 G
2D conv. method	MAD-Net [32]	20 ms	3.45	8.41	4.27	3.75	9.2	4.66	3.83 M	55.66 G
	DipsNetC [10]	60 ms	4.11	3.72	4.05	4.32	4.41	4.34	42.43 M	93.46 G
	SCV-Net [33]	360 ms	2.04	4.28	2.41	2.22	4.53	2.61	2.32 M	726.48 G
	Ours	76 ms	2.50	5.44	2.99	2.69	6.23	3.28	4.61 M	208.21 G

Among these indicators, the computing resource-related indicators (parameters, FLOPs, runtime) produce ambiguous results, so it is difficult to establish their relationship. For example, if an algorithm that includes correlation [9,10] or patch-match [14] is included in the network, additional parameters are not added, but the floating-point operation might increase. Also, a structure with branches, such as a spatial pyramid pooling method [13,37,38] requires memory access for each branch. This can increase runtime and memory usage but not the number of parameters. As shown in the table, most of the 3D convolution-based methods require significantly more parameters and FLOPs, leading to slower runtime than ours. Concretely, the number of parameters in our method is 4.61 M, while that of the DeepPruner-Fast [14] is 7.47 M, which is 1.62 times more than that of ours. Meanwhile, the runtime and FLOPs are comparable. It is worthy to note that the number of parameters is one of important factors which is a measure of the model complexity, and it is directly related to the efficiency of the deep learning networks [47]. Thus, our method is simpler and more efficient than most of 3D convolution-based methods listed in Table 4 in terms of model complexity. On the other hand, some 2D convolution-based methods require relatively few parameters or FLOPs, leading to faster runtime, but produce error ratios that are higher than those of the 3D convolution-based methods. The results show that our method is superior to the 3D convolution-based methods in terms of runtime, while the accuracy of all tested methods is comparable. Although some of the 2D convolution-based methods are faster than our method, they show lower accuracy. Thus, the accuracy and computing resources of our network represent a compromise between

those of the 2D convolution-based and 3D convolution-based methods. Considering these factors, our method represents a good compromise between the two.

Figure 5 shows the results of qualitative comparisons on the test set of the KITTI-2015 benchmark [42]. The images for each method show the error maps of the predicted disparity maps and the predicted disparity maps for the red rectangle regions, where ground-truth disparities exist. In the error maps, the red and yellow colors represent regions with large errors. From these comparisons, it is observed that our method produces comparable results with other methods for various scenes.

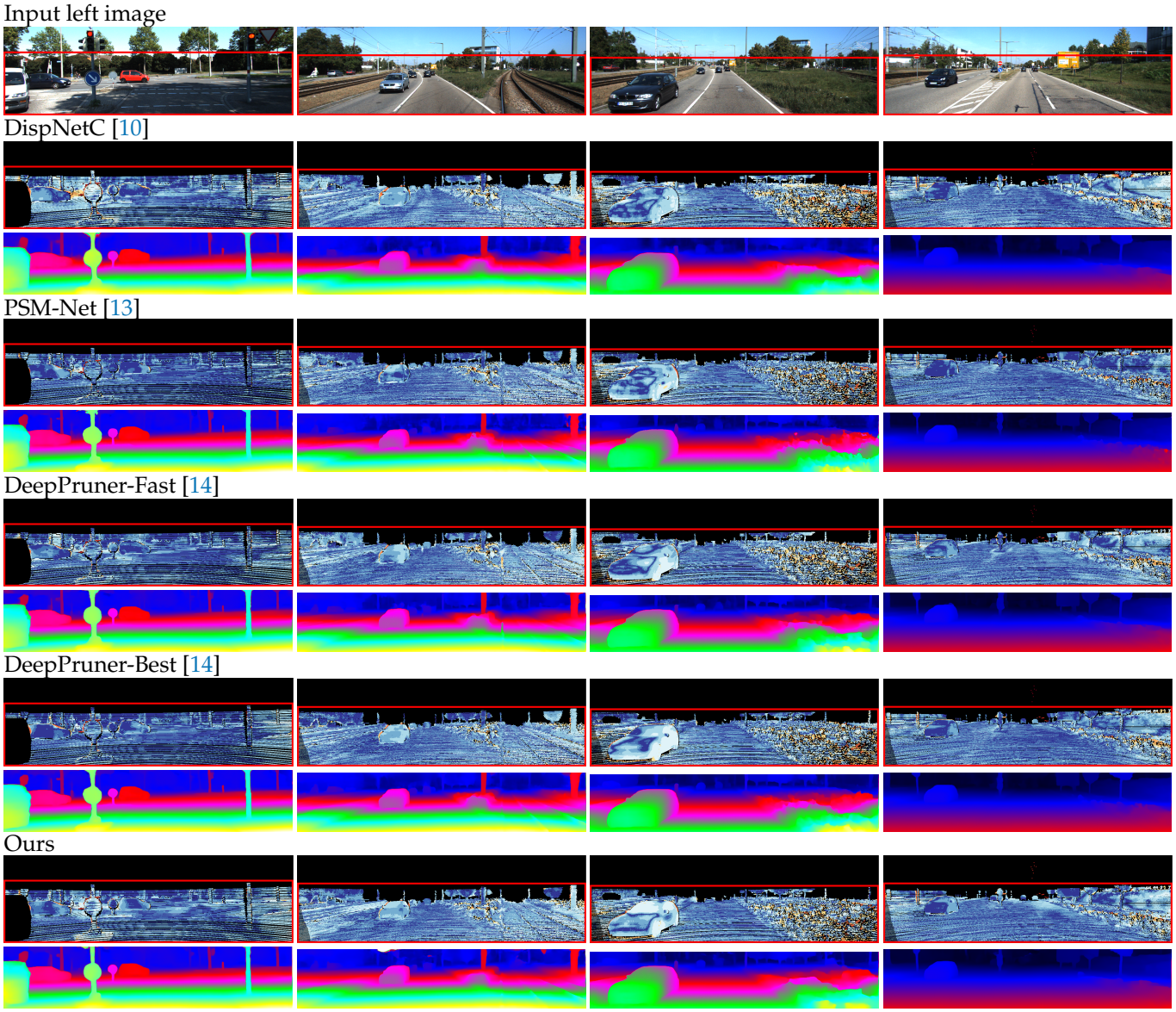


Figure 5. Qualitative comparisons on the KITTI 2015 test set.

4.5. Effect of the Number of Shift S and the Number of Modules M

To investigate the effect of the hyperparameter S and M in Equation (3), we conducted various experiments on the Scene Flow dataset as shown in Table 5. Here, we fixed the value of the full disparity range R, and varied the values of S and M. Except the hyperparameters S and M, all the other settings are the same in these experiments. Because the order of errors in the results at the 100^{th} epoch does not change hereafter, all experiments are conducted only 100 epochs.

Table 5. Number of parameters, runtime, and EPE of our method for different values of S and M on the Scene Flow test set.

	Params	Runtime	EPE
$S = 2, M = 24$	4.6 M	76 ms	1.372
$S = 4, M = 12$	4.2 M	69 ms	1.431
$S = 6, M = 8$	3.9 M	64 ms	1.47

It can be seen that there is a trade-off between EPE, number of parameters, and runtime. As mentioned before, S represents a specific disparity range which is processed in a single SFF module. As the number of S increases, the range of disparity that a single module learns is widen, and the number M of SFF modules required to process the full disparity range decreases. Due to the decreased number of M, parameters of whole network and processing runtime is also reduced. However, it can be seen that EPE increases as S increases. This is because larger value of S requires correspondingly larger receptive field to fully process in a single SFF module. Table 5 shows that EPE reaches the lowest value when $S = 2$ and $M = 24$.

5. Conclusions

In this paper, we propose a simple yet efficient network, called Sequential Feature Fusion Network (SFFNet) for stereo matching. Unlike previous 3D convolution-based networks, our method does not require the construction of heavy 4D cost volume and 3D convolutions for processing it. Instead, our SFFNet sequentially and progressively generates 3D cost volume and processing it using lightweight 2D convolutions. Our SFFNet consists of a series of Sequential Feature Fusion (SFF) modules which sequentially generate 3D cost volumes to cover a part of the disparity range by shifting and concatenating target features, and then process the cost volume using 2D convolutions. Overall, SFFNet prevents heavy computations and allows for efficient generation of an accurate final disparity map. More specifically, with small complexity and small number of parameters, our proposed network generates comparable results with previous 3D convolution-based methods. Various experiments show that our method is relatively faster and require small number of parameters compared to previous 3D convolution-based methods, while achieving comparable accuracy and FLOPs. For example, for the Scene Flow test set, our SFFNet achieves lower EPE with faster runtime and smaller number of parameters than PSMNet. The runtime of our network is 2.8 times faster than that of DeepPruner-Best, while EPE of ours is 1.2 times higher. In future work, we plan to increase the entire performance of our SFFNet. Specifically, to obtain a more accurate final disparity map result in real time, we plan to gradually apply multi-scale approaches to the SFFNet.

Author Contributions: Conceptualization, J.J. and Y.S.H.; software, J.J. and S.J.; validation, Y.S.H.; investigation, J.J. and S.J.; writing—original draft preparation, J.J.; writing—review and editing, S.J. and Y.S.H.; supervision, Y.S.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Ministry of Science and ICT (MSIT), South Korea, under the Information Technology Research Center (ITRC) Support Program supervised by the Institute for Information and Communications Technology Promotion (IITP) under Grant IITP-2021-2018-0-01424.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sansoni, G.; Trebeschi, M.; Docchio, F. State-of-the-art and applications of 3D imaging sensors in industry, cultural heritage, medicine, and criminal investigation. *Sensors* **2009**, *9*, 568–601. [CrossRef]
2. Chen, C.; Seff, A.; Kornhauser, A.; Xiao, J. Deepdriving: Learning affordance for direct perception in autonomous driving. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 2722–2730.
3. Zenati, N.; Zerhouni, N. Dense stereo matching with application to augmented reality. In Proceedings of the IEEE International Conference on Signal Processing and Communications, Dubai, United Arab Emirates, 24–27 November 2007; pp. 1503–1506.

4. El Jamiy, F.; Marsh, R. Distance estimation in virtual reality and augmented reality: A survey. In Proceedings of the IEEE International Conference on Electro Information Technology, Brookings, SD, USA, 20–22 May 2019; pp. 063–068.
5. Huang, J.; Tang, S.; Liu, Q.; Tong, M. Stereo matching algorithm for autonomous positioning of underground mine robots. In Proceedings of the International Conference on Robots & Intelligent System, Changsha, China, 26–27 May 2018; pp. 40–43.
6. Scharstein, D.; Szeliski, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vis.* **2002**, *47*, 7–42. [CrossRef]
7. Zbontar, J.; LeCun, Y. Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. *J. Mach. Learn. Res.* **2016**, *17*, 2287–2318.
8. Zagoruyko, S.; Komodakis, N. Learning to compare image patches via convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 4353–4361.
9. Luo, W.; Schwing, A.G.; Urtasun, R. Efficient deep learning for stereo matching. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 5695–5703.
10. Mayer, N.; Ilg, E.; Hausser, P.; Fischer, P.; Cremers, D.; Dosovitskiy, A.; Brox, T. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 4040–4048.
11. Pang, J.; Sun, W.; Ren, J.S.; Yang, C.; Yan, Q. Cascade Residual Learning: A Two-Stage Convolutional Neural Network for Stereo Matching. In Proceedings of the IEEE International Conference on Computer Vision Workshops, Venice, Italy, 22–29 October 2017; pp. 887–895.
12. Kendall, A.; Martirosyan, H.; Dasgupta, S.; Henry, P.; Kennedy, R.; Bachrach, A.; Bry, A. End-to-end learning of geometry and context for deep stereo regression. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 66–75.
13. Chang, J.R.; Chen, Y.S. Pyramid Stereo Matching Network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 5410–5418.
14. Duggal, S.; Wang, S.; Ma, W.C.; Hu, R.; Urtasun, R. Deeppruner: Learning efficient stereo matching via differentiable patchmatch. In Proceedings of the IEEE International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 4384–4393.
15. Wang, Y.; Lai, Z.; Huang, G.; Wang, B.H.; Van Der Maaten, L.; Campbell, M.; Weinberger, K.Q. Anytime stereo image depth estimation on mobile devices. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 5893–5900.
16. Hirschmuller, H. Stereo processing by semiglobal matching and mutual information. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *30*, 328–341. [CrossRef] [PubMed]
17. Birchfield, S.; Tomasi, C. Depth discontinuities by pixel-to-pixel stereo. *Int. J. Comput. Vis.* **1999**, *35*, 269–293. [CrossRef]
18. Hamzah, R.A.; Abd Rahim, R.; Noh, Z.M. Sum of absolute differences algorithm in stereo correspondence problem for stereo matching in computer vision application. In Proceedings of the International Conference on Computer Science and Information Technology, Chengdu, China, 9–11 July 2010; Volume 1, pp. 652–657.
19. Hirschmuller, H.; Scharstein, D. Evaluation of cost functions for stereo matching. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, 17–22 June 2007; pp. 1–8.
20. Yoo, J.C.; Han, T.H. Fast normalized cross-correlation. *Circuits, Syst. Signal Process.* **2009**, *28*, 819–843. [CrossRef]
21. Zabih, R.; Woodfill, J. Non-parametric local transforms for computing visual correspondence. In Proceedings of the European Conference on Computer Vision, Stockholm, Sweden, 2–6 May 1994; pp. 151–158.
22. Geng, N.; Gou, Q. Adaptive color stereo matching based on rank transform. In Proceedings of the International Conference on Industrial Control and Electronics Engineering, Xi'an, China, 23–25 August 2012; pp. 1701–1704.
23. Lu, H.; Meng, H.; Du, K.; Sun, Y.; Xu, Y.; Zhang, Z. Post processing for dense stereo matching by iterative local plane fitting. In Proceedings of the IEEE International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, Las Vegas, NV, USA, 30 June–2 July 2014; pp. 1–6.
24. Xu, L.; Jia, J. Stereo matching: An outlier confidence approach. In Proceedings of the European Conference on Computer Vision, Marseille, France, 12–18 October 2008; pp. 775–787.
25. Aboali, M.; Abd Manap, N.; Yusof, Z.M.; Darsono, A.M. A Multistage Hybrid Median Filter Design of Stereo Matching Algorithms on Image Processing. *J. Telecommun. Electron. Comput. Eng.* **2018**, *10*, 133–141.
26. Ma, Z.; He, K.; Wei, Y.; Sun, J.; Wu, E. Constant time weighted median filtering for stereo matching and beyond. In Proceedings of the IEEE International Conference on Computer Vision, Sydney, Australia, 1–8 December 2013; pp. 49–56.
27. Sun, X.; Mei, X.; Jiao, S.; Zhou, M.; Wang, H. Stereo matching with reliable disparity propagation. In Proceedings of the 2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission, Hangzhou, China, 16–19 May 2011; pp. 132–139.
28. Wu, W.; Zhu, H.; Yu, S.; Shi, J. Stereo matching with fusing adaptive support weights. *IEEE Access* **2019**, *7*, 61960–61974. [CrossRef]
29. 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. [CrossRef]

30. Yang, G.; Zhao, H.; Shi, J.; Deng, Z.; Jia, J. Segstereo: Exploiting semantic information for disparity estimation. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 636–651.
31. Yin, Z.; Darrell, T.; Yu, F. Hierarchical discrete distribution decomposition for match density estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 6044–6053.
32. Tonioni, A.; Tosi, F.; Poggi, M.; Mattoccia, S.; Stefano, L.D. Real-time self-adaptive deep stereo. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 195–204.
33. Lu, C.; Uchiyama, H.; Thomas, D.; Shimada, A.; Taniguchi, R.i. Sparse cost volume for efficient stereo matching. *Remote Sens.* **2018**, *10*, 1844. [CrossRef]
34. Tulyakov, S.; Ivanov, A.; Fleuret, F. Practical Deep Stereo (PDS): Toward applications-friendly deep stereo matching. In Proceedings of the International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; pp. 5875–5885.
35. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
36. Mehta, S.; Rastegari, M.; Caspi, A.; Shapiro, L.; Hajishirzi, H. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 552–568.
37. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [CrossRef] [PubMed]
38. Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid scene parsing network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2881–2890.
39. Lin, M.; Chen, Q.; Yan, S. Network in network. *arXiv* **2013**, arXiv:1312.4400.
40. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
41. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 3354–3361.
42. Menze, M.; Geiger, A. Object scene flow for autonomous vehicles. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3061–3070.
43. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]
44. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations (Poster), San Diego, CA, USA, 7–9 May 2015.
45. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 8026–8037.
46. Song, X.; Zhao, X.; Hu, H.; Fang, L. Edgestereo: A context integrated residual pyramid network for stereo matching. In Proceedings of the Asian Conference on Computer Vision, Perth, Australia, 2–6 December 2018; pp. 20–35.
47. Bianco, S.; Cadene, R.; Celona, L.; Napoletano, P. Benchmark Analysis of Representative Deep Neural Network Architectures. *IEEE Access* **2018**, *6*, 64270–64277. [CrossRef]