Journal of Autonomous Intelligence (2024) Volume 7 Issue 5
doi: 10.32629/jai.v7i5.813

ORIGINAL RESEARCH ARTICLE

MSDE: Multi-scale disparity estimation model from stereo images

Ahmed Alghoul[1], Mhd Rashed Al Koutayni[2,*], Ramy Battrawy[2], Didier Stricker[2], Wesam Ashour[1]

[1] *Department of Computer Engineering, Islamic University of Gaza, Gaza 108, Palestine*

[2] *Augmented Vision Department, German Research Center for Artificial Intelligence (DFKI) GmbH, 67663 Kaiserslautern, Germany*

*** Corresponding author:** Mhd Rashed Al Koutayni, mhd_rashed.al_koutayni@dfki.de

Abstract

Most modern stereo matching algorithms predict an accurate disparity map but demand high memory and processing requirements as well as a huge number of floating-point operations. Consequently, their applicability is constrained to high-powered devices with substantial capacities, posing challenges for implementations on low-power devices. To address this problem, we propose MSDE, an efficient end-to-end neural network model designed to strike a balance between estimation accuracy and resource utilization. MSDE is based on hierarchical disparity estimation along with the computation of low-dimensional residual and error cost volumes. To reduce the operations, 3D convolutional layers are factorized into 2D and 1D convolutional layers to improve the efficiency of filtering and the aggregation cost volume features. As a result, the entire model of our MSDE has 48 K parameters, requires 2.5 G floating-point operations (FLOPs), and runs with comparatively small memory footprint of 730 M with an execution time of 29.5 ms for each frame on the RTX 2080Ti GPU. Compared to state-of-the-art methods, our model is more efficient, offers a trade-off between accuracy and efficiency, and it needs low hardware resources.

Keywords: disparity; factorization; computer vision; disparity estimation; stereo matching; 3D convolutions

ARTICLE INFO

Received: 27 June 2023
Accepted: 19 January 2024
Available online: 17 April 2024

1. Introduction

3D sensing has emerged as one of the necessary means to perceive scene geometry, which can play an important role in a variety of computer vision tasks such as autonomous driving, remote sensing, robotics, and object detection, obstacle avoidance, augmented reality and object detection. There are a number of 3D sensing modalities available, such as RGB-D (i.e., kinetic) cameras, stereo systems, structured-light, and LiDARs, which differ in technology and cost. Among these modalities, stereo systems are gaining more attention, because of their ability to operate at long ranges in outdoor scenes compared to RGB-D systems and their low cost compared to LiDAR. To obtain the geometry from stereo images, it is necessary to compute the relative displacement between each pixel in the left view and its corresponding pixel in the right view, so that the disparity map can be estimated[1,2].

Today, end-to-end learning-based disparity estimation approaches from stereo systems[3–6] show very impressive results compared to hand-crafted methods[7]. The accuracy of learning-based approaches is mostly based on the construction of a high-dimensional (4D) cost volume[8–10] including the maximum possible range of disparity or search range matches. This high-dimensional cost volume

then requires further filtering using a large number of 3D convolutional layers convolutional to obtain best matches and decoding them into an accurate disparity map. As a result, these methods are hindered by a large memory footprint, a high computational complexity, and a large number of model parameters, reducing their overall efficiency and requiring more time to predict disparity. To tackle this problem, DeepPruner[11], enable real-time inference by reducing the search range and thus reducing the cost volume dimensionality and follow with an image-guided refinement module to improve the accuracy. All of the aforementioned models require a large number of parameters and operate with a large number of floating operation points (FLOPS), which also requires a large amount of memory.

In our framework, we propose our MSDE—an efficient multi-scale disparity estimation model, which is designed in an end-to-end manner by extracting features at multiple scales and using them to construct multi-scale cost volumes. To reduce the overall cost volume dimensionality, we construct an initial cost volume (ICV) at the coarse scale considering the maximum search range then we shorten this range at the upper scales along with the construction of residual and error cost volumes, (RCV) and (ECV), respectively. In addition, we replace the 3D convolutional layers with a series of 2D and 1D convolutions to improve the overall filtering process of the cost volumes. These two aspects allow our model to operate in real-time with a significant reduction in the number of parameters numbers and with very few floating-point operations (FLOPs), allowing the implementation on mobile devices and Field-Programmable Gate Arrays (FPGAs) with small memory and battery capacity. However, in our paper we do not verify the implementation on mobile devices or FPGAs.

Our contribution can be summarized as follows:

- We propose our MSDE—an end-to-end neural network architecture for disparity estimation. Our model is a lightweight model that estimates disparity at multiple scales from coarse-to-fine and computes a dense disparity map at the full resolution of the given stereo images.
- In our MSDE, we combine the cost volume refinement with residual disparity map estimation to improve our overall accuracy providing a trade-off between accuracy and efficiency.
- For high efficiency, we reduce the overall cost volume dimensionality of our cost volume and instead of using 3D convolutional layers for filtering and refinement, we implement a factorized version of 3D convolutions through a series of 2D and 1D convolutions.
- Our model operates efficiently with a low runtime, a low memory footprint, and a reduced number of parameters, requiring a low number of floating-point operations (FLOPs) compared to the state-of-the-art methods.

The overall structure of this paper consists of five sections, the previous related studies, our proposed model, the experiments and results, and finally, the conclusion and future work.

2. Related work

Many algorithms and techniques have been developed in the past to estimate disparity. These algorithms can be categorized as either hand-crafted algorithms[7] or learning-based methods[3]. Deep learning-based stereo matching methods learn disparity estimation, enabling the generation of accurate and detailed maps that represent the depth variations between the images using convolutional neural networks (CNNs)[3]. CNNs are used to replace one or more parts of a conventional stereo matching pipeline with a matching cost component[5,6]. This substitution leads to the development of non-end-to-end algorithms, which are more efficient than hand-crafted techniques but entail a higher computational cost. CNNs are used instead of the cost-aggregating components[12,13] and in place of the refining blocks[14].

With the advent advances of CNNs, non-end-to-end approaches are evolved into end-to-end learning algorithms. The encoder-decoder class and regularity learning for 3D Convolutions class are two

classifications of this category[4]. In this work, we are also paying close attention to the use of supervised deep learning, which requires labels to guide the training process, as opposed to self-supervised methods due to their performance is not good as supervised methods[15].

The encoder-decoder class: This class uses an encoder and decoder to estimate the disparity maps, such as DispNetC[16], CRL[17], and iResNet[14] models. These models consist of two sub-networks; one for estimating the initial disparity map and the other to regulate the estimated predicted maps. These methods suffer from a large number of parameters and inaccurate disparity estimation in occluded areas and textureless regions. To enhance the performance of the disparity estimation models and to reduce the complexity of generating cost volumes, EdgeStereo[18] and DeepPruner[11] models are introduced. The former utilizes a shallow edge detector sub-network while the latter uses PatchMatch search techniques to reduce the size of generated cost volumes. Instead of relying on the encoder-decoder structure in all phases of the disparity estimation pipeline, our model utilizes the encoder-decoder architecture solely in the feature extraction module. Additionally, it achieves a compact model size by employing multi-scale disparity estimation and incorporating residual disparity. This approach offers a more efficient and accurate solution for disparity estimation tasks.

Regularity Learning for 3D Convolutions class: In contrast to networks and their variants[14,16,17], diverse methods use regulation units that consist of a series of 3D convolutions. These convolutions apply to 4D cost volumes. These volumes consist of four dimensions which are height, width, features, and disparity values[19]. Even though this class has better performance than the previous one, it increases the memory footprint requirements and needs high computational resources, which leads to increased inference time. These methods are GC-Net[20], PSMNet[8], SCV-Net[21], PDSNet[22], GA-Net[23], and GWCNet[24]. LRCR[25] uses an innovative end-to-end approach in which disparity estimation is done by utilizing two parallel Long Short-Term Memory (LSTM) convolution networks[26] but this technique needs a long time to estimate the disparity maps.

To reduce the inference time, AnyNet[27] uses a residual disparity map to predict the final disparity maps in three stages. Additionally, lightweight models are introduced to predict the disparity by reducing the number of 3D convolutions such as ES-Net[28] and StereoNet[9]. StereoNet utilizes only five 3D convolutions to estimate an initial disparity map from 4D cost volume after downsampling the input images to a coarse resolution. Moreover, LEAStereo[10] introduces a hierarchical Neural Architecture Search (NAS) which is the first end-to-end framework that improves the accuracy by a large margin. However, the accuracy of all the aforementioned approaches relies on the construction of high-dimensional 4D cost volumes with large search ranges of disparity candidates. As a result, they require a number of 3D convolutional layers for filtering, which increases their overall parameters, and makes their implementation resource-intensive.

Unlike previous methods, our MSDE designs low-dimensional cost volumes at different scales, reducing the overall computations for subsequent filtering.

Cost Volume Filtering: Cost Volume filtering is necessary to minimize and remove the noise in the generated cost volume. It requires high computational resources and memory-intensive 3D convolutional layers to aggregate the 4D cost volumes[9,21,25,28]. To reduce this cost, the number of convolutions is cut down and refinement blocks are used to compensate for the reduction of 3D convolutions and improve the accuracy at the expense of speed[23].

A method previously utilized in CNNs for video analysis and action detection and adapted to ResNet is the factorization of 3D convolutions into 2D convolutions and 1D convolutions[29]. A 3D convolution kernel of size is divided into two parts using this method. The first is the spatial component which uses 2D convolution and the second one is a temporal component that uses 1D convolution[30,31]. Additionally, Gonda

et al.[32] implements a generalized method with a mathematical justification to factorize a 3D convolution block into a 1D convolution after successive 2D convolution blocks along different dimensions in the case that 1D and 2D convolutions are normal to each other.

In our MSDE, we speed up the cost volume filtering module by use replacing the 3D convolutional layers with either 2D convolutions only, 1D convolutions only, or both 2D and 1D convolutions requiring fewer resources.

3. Our proposed architecture

Our MSDE estimates a dense disparity map from stereo images (left and right) at the full input image size with respect to the left view. To this end, our model is designed hierarchically to match each pixel in left with its correspondence in right. We assume that both input images are rectified and are of the same size (i.e., height and width (H, W)). As shown in **Figure 1**, our complete network design consists of five modules: a feature pyramid network module, a cost volume module constructed of three steps, a fast cost filtering module, a disparity regression module, and finally, a residual refinement module. We describe the components of each module in detail in the following sections:

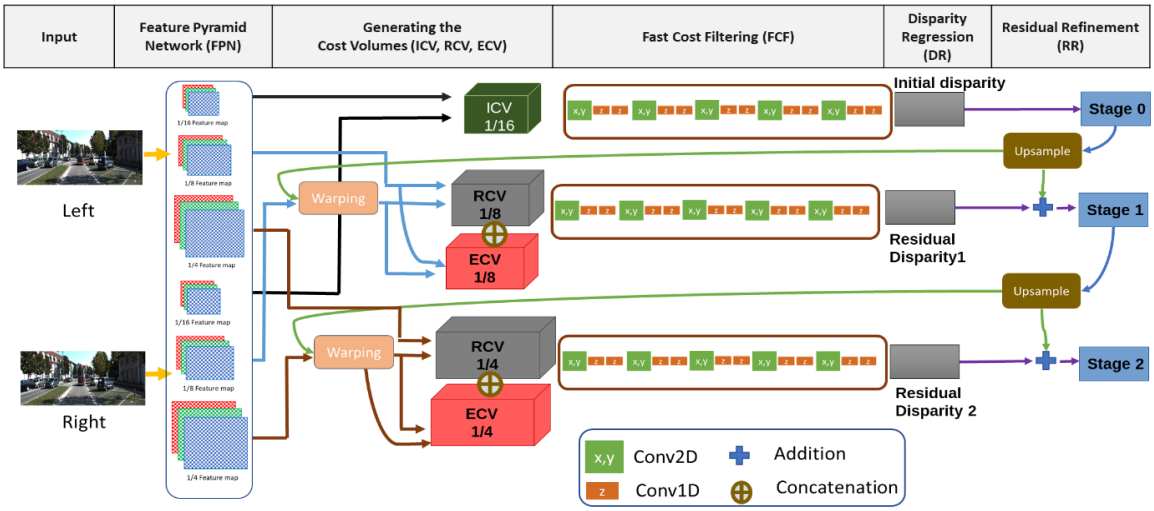

Figure 1. The complete pipeline of the Multi-scale Disparity Estimation (MSDE) model. Black, light blue and brown arrows indicate feature maps at the scales of 1/16, 1/8 and 1/4 respectively. Purple arrows indicate the disparity at each stage. Blue arrows indicate feeding the output of each stage for upsampling. Light green arrows indicate the upsampled disparity maps.

3.1. Feature pyramid network (FPN)

Representing each pixel in the input images with informative and distinctive features is the first key requirement for our pixel-wise matching process. Our feature extraction module consists of two pyramid networks with shared parameters for the hierarchical extraction of two feature sets from left and right images. Inspired by the FPN[33], our pyramid network consists of two pathways for downscaling and upscaling, connected by lateral connections at equal scales to extract hierarchical feature heads.

Contrary to the general design in FPN[33], we build an efficient feature extraction module consisting of only three scales to reduce the computational requirements and the time consumed for feature extraction. Each scale layer in the encoder part consists of two successive layers: First, a 2D convolutional layer with kernel size $k = 7 \times 7$ padding $p = 3$, stride $s = 2$, followed by batch normalization (BN) and ReLU activation function. Second, max-pooling layer and with kernel size $k = 3 \times 3$ and stride $s = 2$. Both layers together downsample the image by factor v and generate three feature maps at three different scales, each corresponding to $1/(2^r v)$ of the full input image size. The shape of each feature map is $(C \times H/(2^r v) \times W/(2^r v))$, where H and W are the height and width of the full input image size, C is the number of channels

given by $2^{(r+1)}v$, and $r \in [0, 1, 2]$ is the scale parameter.

3.2. Cost Volume (CV) module

Cost volume is the key requirement for matching across left and right images. In this context, previous state-of-the-art methods[34] consider the following main steps for robust cost volume construction: 1) Grouping likelihood candidates of correspondences, 2) robust aggregation of correspondence features, and, 3) refinement of cost volume.

Following this strategy, we design our cost volume at multiple scales consisting of three steps: 1) Initial Cost Volume (ICV) with $1/(2^r v)$ resolution at a coarse scale (*i.e.*, $r = 2$), 2) Efficient Filtering Layers, 3) Residual Cost Volume (RCV) and Error Cost Volume (ECV) for refinement at upper scales ($i.e.,\ r = 1\ and\ r = 0$). However, compared to the state-of-the-art methods, we implement an efficient and computationally cheap design in each main step, making our model with low parameters and requiring low number of (FLOPs).

3.2.1. Initial cost volume

After obtaining the feature maps of the left and right images at the coarse scale, we construct the Initial Cost Volume (ICV) to determine an initial estimate of the disparity map at the coarse resolution. Since we assume that both images are rectified, we consider that pixels within a certain range in the corresponding row in the right image are possible likelihood correspondences to each pixel in the left image (i.e., our reference view), so that disparity d of each pixel lies within $[1 \ldots D]$. The central position of each given range is defined based on the pixel position (x, y) in the left image, which we need to find its possible correspondences in row (y) in the right image. This assumption is applied to all pixels at the coarse resolution in the left view to construct the initial cost.

With the left and right images at the coarse scale, we measure the similarity between the features of each pixel (i.e., (x, y)) in the left image and its horizontal counterpart by subtracting their two features. To reduce the dimensionality of the cost volume, which requires low memory and computational resources compared to the cost volume in the state-of-the-art methods, L1-norm is then applied along the channel dimension generating a 2D feature map for every disparity value. These 2D feature maps are concatenated to generate a 3D cost volume with the shape $(D \times H/(2^r v) \times W/(2^r v))$, where $D = D_{max}/(2^r v)$, $r = 2$ for the coarse resolution, and D_{max} is the maximum disparity at full image resolution, which is equal to 192 aligning with values used in other papers.

After generating the ICV, the initial disparity map is predicted after passing it through the fast cost filtering and the disparity regression modules, as shown in **Figure 1** at Stage 0. This map is noisy and inaccurate because the ICV loses spatial and local information, which is not sufficient to obtain an accurate and final disparity map. As a result, we generate additional cost volumes such as RCV and ECV.

3.2.2. Residual and error cost volumes

Since ICV is noisy and imprecise, refinements are required at the upper scales. Therefore, Residual Cost Volume (RCV) and Error Cost Volume (ECV), inspired by Kang et al.[35], are constructed for the upper scales, $i.e.,r = 1\ and\ r = 0$, using a residual disparity Δd, where $\Delta d \in [-d_{offset},\ d_{offset}]$ is an interval of residual disparities and d_{offset} expresses the initial matching accuracy. By employing both RCV and ECV, the accuracy of the initial disparity map will be increased, and the cost of generating and refining the cost volumes will be reduced.

Unlike the maximum disparity at the coarse scale, we consider a different maximum disparity value, denoted as D^r, at the upper scales. The calculation of D^r is determined by Equation (1):

$$D^r = 2 \times (d_{offset} + 1) - 1/sub\_pixel\_acc \quad (1)$$

where sub_pixel_acc is the sub-pixel accuracy which defined as the step value within the range $[-d_{offset}, d_{offset}]$. Decreasing the sub_pixel_acc value results in an increased D^r value, which subsequently raises both the computational cost and the precision of the predicted disparity map. However, as the sub_pixel_acc value increases, the accuracy of the disparity map decreases due to a reduction in the value of D^r. For instance, if $d_{offset} = 2$ and sub_pixel_acc = 1, then D^r = 5, and if sub_pixel_acc = 0.5, then $D^r = 10$. Therefore, in our model, we utilize a sub_pixel_acc value of 1 pixel and a d_{offset} of 2 to achieve a trade-off between computational efficiency and accuracy. Therefore, for all cost volumes of RCV and ECV, the interval of the residual disparities is defined as [−2, 2].

The construction of RCV and ECV is clarified in **Figure 2**. Using bilinear interpolation, we upsample the initial disparity map to the upper scale then we warp the coordinates of the right image to the left one and obtain their corresponding features at the upper scale. Consequently, RCV is constructed to find the correlation between the left feature map F_L and the warped right feature map $F_R^{\grave{\Delta}d}$ for scales 1 and 0. The correlation is done by convolving two windows centered at x_L and $x_R^{\Delta d}$ in F_L and $F_R^{\grave{\Delta}d}$ where x_L and $x_R^{\Delta d}$ are the x-coordinates for the required pixels in the F_L and $F_R^{\grave{\Delta}d}$ respectively. This process yields a correlated cost volume. This correlated cost volume with dimension $(C \times H/(2^r v) \times W/(2^r v))$ is L1-normed along the depth dimension C, resulting in a residual cost channel $C^{\Delta d}$ at Δd with dimensions $(H/(2^r v) \times W/(2^r v))$ as shown in Equation (2), following Kang et al.[35]. Similar to the low dimensional correlation process at the coarse scale, we compute the similarity of the features within the windows size by subtracting the features and applying L1-Norm constructing RCV with dimensions of $(D^r \times H/(2^r v) \times W/(2^r v))$. The used window size is $2b + 1$, where $b \geq 1$ and b is an integer, $*$ is convolution, and $\|.\|$ is L1-norm. Our model uses a tiny window with $b = 1$ to limit the number of correlation operations. Lastly, concatenating all these channels to get the final residual cost volume $(D^r \times H/(2^r v) \times W/(2^r v))$.

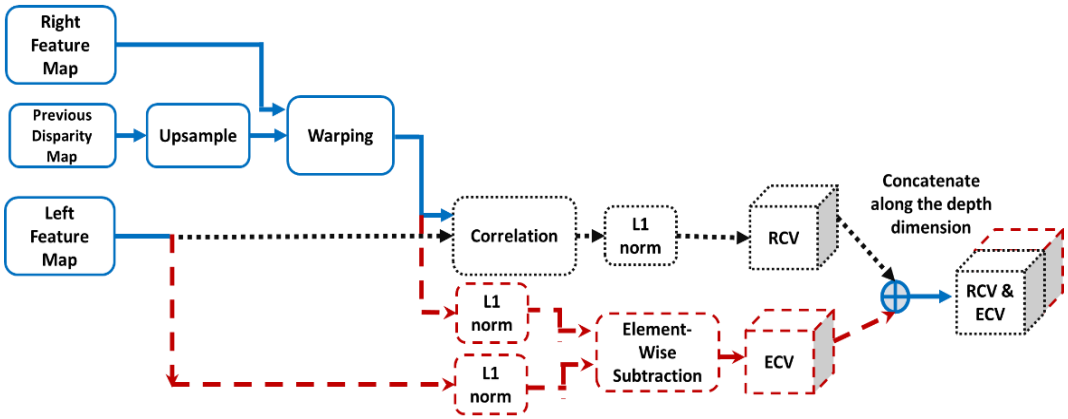

Figure 2. The process of generating both RCV and ECV.

$$C^{\Delta d}(x_L, x_R^{\Delta d}) = \left\| \sum_{i \in [-b,b] \times [-b,b]} [F_L(x_L + i) \times F_R^{\grave{\Delta}d}(x_R^{\Delta d} + i)] \right\| \quad (2)$$

To enhance the robustness of our full cost volume, we expand RCV with error cost volume (ECV) which starts by calculating the L1-norm separately for both the left feature map F_L and the warped feature right $F_R^{\grave{\Delta}d}$ along the depth dimension C to get the two maps $\|F_L\|$ and $\|F_R^{\grave{\Delta}d}\|$ with $(H/(2^r v) \times W/(2^r v))$ dimensions. After that, an elementwise subtraction is performed between $\|F_L\|$ and $\|F_R^{\grave{\Delta}d}\|$ using Equation (3) to get the error cost channel $C_E^{\Delta d}$ for every residual Δd. In the optimal case, the error is zero if $F_R^{\grave{\Delta}d}$ is

identical to F_L, but this is not actual case due to downsampling of the feature maps, inaccurate predicted disparity map, and the existence of occlusion regions in which a reconstruction error occurs. In the last step, the error cost channels $C_E^{\Delta d}$, where $\Delta d = \{-2, -1, 0, 1, 2\}$ are concatenated along the depth dimension to get ECV C_{ECV} with dimension $(D^r \times H/(2^r v) \times W/(2^r v))$.

$$C_E^{\Delta d} = \left| \|F_L\| - \left\|F_R'^{\Delta D}\right\| \right| \tag{3}$$

After creating the two cost volumes RCV and ECV, they are concatenated along the depth dimension to generate the final Residual and Error Cost Volume (RECV) with dimension $(2D^r \times H/(2^r v) \times W/(2^r v))$.

3.3. Fast cost filtering module

To aggregate the cost volume features, many approaches apply series of 3D convolutional layers (i.e., kernel size: $k \times k \times k$), which are computationally expensive. We tackle this problem by factorizing the 3D convolutions into low-dimensional convolutional layers, called Fast Cost Filtering (FCF) Module. Based on some experiments shown in the Ablation Study section, the best-factorized version replaced 3D convolutions with one 2D convolutional layer and two of 1D convolutional layers (2D + 1D + 1D). Thus, 2D convolution applies a kernel size of $k \times k \times 1$, and each 1D convolution applies a kernel size of $1 \times 1 \times k$, as illustrated in **Figure 3**. Thus, the 2D convolution is performed in the *x*, and *y* coordinates (i.e., columns and rows) of the constructed cost volume, while the 1D convolutions are conducted along the disparity axis (i.e., z-axis). After factorization, the 2D convolution retains the same number of input channels N_{i-1} as the original 3D convolution. Moreover, the number of output channels N_i for the 3D convolution is equivalent to the number of output channels for the second 1D convolution in the factorized version. We consider M_i as the number of channels for the middle layers, which can be computed in our factorized version as in Equation (4):

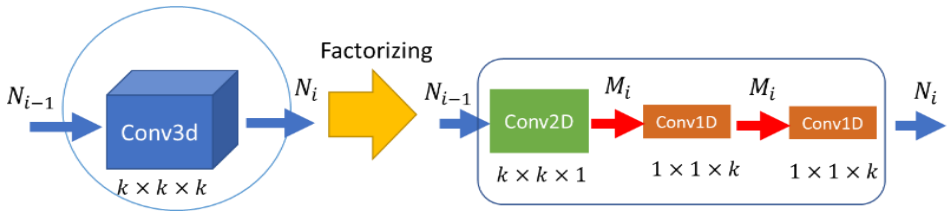

Figure 3. Factorizing 3D convolution into one 2D convolution and two 1D convolutions.

$$M_i = \left\lfloor \frac{k^3 N_{i-1} N_i}{k^2 N_{i-1} + k N_i} \right\rfloor, \tag{4}$$

The total number of parameters of our factorized version is given by Equation (5), where the first, second, and third terms in this equation represent the number of parameters for the 2D convolution, the first 1D convolution, and the second 1D convolution, respectively.

$$Total\ parameters = k^2 N_{i-1} M_i + k M_i^2 + k M_i N_i \tag{5}$$

Contrary to the fact that the number of parameters in (2D + 1D + 1D) is greater than in 3D convolution by kM_i^2, the decomposition of the convolutions into spatial and temporal components yields two benefits. First, the required processing resources are reduced by employing only 2D and 1D kernels instead of 3D kernels. Furthermore, the models that use 2D convolutions have less training error[31].

We apply our FCF five times, each followed by a batch normalization layer[36] and use a LeakyReLU as an activation function[37], as shown in **Figure 4a**, except for the last factorized block in which batch normalization is removed from it. Then, we obtain a filtered cost volume with a dimension of $(D \times H/(2^r v) \times W/(2^r v))$, where $D = D_{max}/(2^r v)$. **Figure 4b** depicts the symbolic representation of this factorized version for a single 3D convolution which is used in **Figure 1**. The green square represents a 2D convolution, while each orange rectangle represents a 1D convolution.

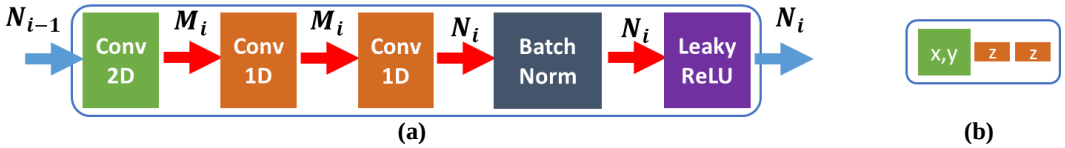

Figure 4. (a) The inner structure of the factorized version of a single 3D convolution; **(b)** The symbolic representation of the factorized version of a single 3D convolution, where the green square represents a 2D convolution with a kernel size of $\boldsymbol{k} \times \boldsymbol{k} \times \mathbf{1}$ in *x* and *y* coordinates, while each orange rectangle represents a 1D convolution with a kernel size of $\mathbf{1} \times \mathbf{1} \times \boldsymbol{k}$ in z coordinates.

3.4. Disparity regression

After constructing the cost volumes, a Disparity Regression (DR) module is designed to use them for predicting 2D disparity maps across all three scales, where an initial disparity map is predicted at the coarse scale and residual disparity maps are predicted at the upper scales.

Inspired by Kendall et al.[20], to estimate the best possible disparity value with the lowest cost, we apply the SoftMax function $\sigma(.)$ to the output features of the cost volumes over the disparity axis (i.e., z-axis) for each disparity i which is $c_i(d)$, then we softly weight the disparity range by multiplying with resulting in the best disparity prediction for each pixel in the left image using Equation (6). The reason for using the minus sign in this equation is that for higher cost values, the probability is smaller, and vice versa. Therefore, we ensure that the resulting probability reflects the relationship between the cost and the probability, enabling more accurate disparity predictions.

$$d_i = \sum_{d=1}^{D} d \times \sigma(-c_i(d)) \tag{6}$$

Equation (6) is applied at multiple scales from coarse-to-fine. Since the resulting disparity map at the coarse scale is noisy and lacks fine details for small objects and structures, we predict directly through this module residual disparity maps at the upper scales which are then summed to the upsampled disparity maps, which are obtained from the lower scales. We call this process Residual Refinement Module (RR). Two considerations justify these improvements. Firstly, the estimated disparity map is residual, not a full map, at both scales of 1 and 0 from RCV and ECV cost volumes with D^r channels. Secondly, adding the previous disparity map to the residual map improves the accuracy of the initial map.

3.5. Loss function

Our MSDE model is trained in a fully supervised manner using ground truth data labels of disparity maps. Moreover, a hierarchical loss function is used to reduce the amount of error made during training.

Given ground truth data labels of disparity maps $d_i^{\hat{}}$, and the predicted disparity d_i^u at pixel i in the u-th stage, our hierarchical loss function can be computed as follows:

$$L = \sum_{u} f(d_i^u - d_i^{\hat{}}) \tag{7}$$

$$f(x, q) = \sqrt{(x/q)^2 + 1} - 1 \tag{8}$$

where $f(.)$ is a robust one-parameter function with $q = 2$, represents the smooth version of the L1 loss function, inspired by Barron et al.[38].

3.6. Single-scale disparity estimation model

To illustrate the impact of increasing the model's parameters and complexity on the evaluation results and inference time, we introduce a modified model called Single-Scale Disparity Estimation (SSDE) model (**Figure 5**). The purpose of the SSDE model is to predict a disparity map by extracting features from two stereo images using a single coarse scale. The first part of the model is a Siamese feature extraction (FE) layer consisting of two identical subnetworks that share the same weights. This subnetwork is responsible for

extracting representative features from both input stereo images. Specifically, it consists of three downsampling layers, six residual blocks, and a 2D convolution layer. Once the features are extracted, they are element-wisely subtracted for each disparity value. Afterwards, the 3D results are concatenated along the disparity dimension to generate a 4D cost volume of size (C×D×H×W). This is different from MSDE where L1 norm is applied. Afterwards, the 4D cost volume undergoes the Fast Cost Filtering (FCF) which consists of a series of 5 factorized 3D Conv layers, just as in MSDE, resulting in a 3D volume. This is where the Disparity Regression (DR) layer is used to get an initial 2D disparity map. The Hierarchical Refinement (HR) layer consists of three hierarchical stages in which the output of every stage will be one of two concatenated inputs along the channel dimension for the next stage. The first input is the bilinearly upsampled predicted disparity steered with the second input which is the resized reference input image to the same dimension as the first input. These two inputs are concatenated along the channel dimension. We decided to utilize bilinear upsampling and convolutions instead of deconvolutions because deconvolutions have recently been found to generate checkerboard artifacts.

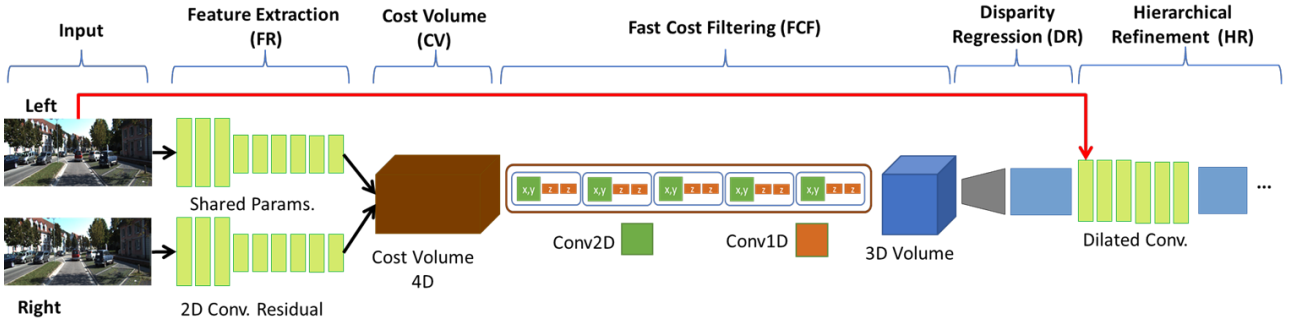

Figure 5. The complete pipeline for the Single-Scale Disparity Estimation (SSDE) model.

4. Experiments and results

We perform several experiments to verify our design decision. First, we compare against the state-of-the-art methods and conduct several experiments on our architecture to verify the effectiveness of our components.

4.1. Datasets and evaluations metrics

To perform the experiments, we consider three datasets: FlyingThings3D[16], KITTI 2012[39], and KITTI 2015[40]. FlyingThings3D is a large-scale synthetic dataset comprising stereo image pairs. It consists of 22872 stereo pairs in the train split and 4370 pairs of images in the test split, with all images having a resolution of 960 × 540. Every pair has an exact ground truth disparity map. To test our MSDE on real-world datasets, we consider KITTI datasets of both versions KITTI 2012 and KITTI 2015. Since the train split of both versions has ground truth, we verify our model on the train splits of KITTI, where KITTI 2012 has 194 pairs and KITTI 2015 has 200 training and 200 testing pairs, both of image resolution 1240 × 376 images. Unlike FlyingThings3D, KITTI provides sparse disparity ground truth for the disparity maps and the stereo pairs in KITTI 2015 have moving cars.

Various metrics are used to evaluate estimated disparity maps. These measurements are based on the difference between the estimated disparity map and the ground truth disparity. These measurements are:

- End Point Error (EPE)[41] is the average absolute difference in pixels between the estimated and the ground truth disparity maps for all valid pixels in these maps. A disparity value for a pixel is valid if it is greater than zero and less or equal to the maximum disparity value.
- Outliers are the percentage of valid pixels whose absolute difference between the estimated disparity value and the ground truth value is more than the threshold value, which is 3 by default.

4.2. Comparison to the state-of-the-art

In this section, we delve into a comprehensive efficiency analysis for our model, unveiling its outcomes and conducting a comparative analysis against state-of-the-art models.

4.2.1. Quantitative results

The proposed MSDE model is a lightweight and efficient model for disparity estimation. It has just 48.2 K parameters, consumes only 729.8 M memory, and takes solely 29.5 ms to infer at RTX 2080Ti GPU as shown in the last row of **Table 1**. This is achieved by restricting our hierarchical design to 3 scales and due to reduced dimensionality of cost volume with our efficient factorized convolutional layers.

We compare the efficiency of our model with state-of-the-art methods in terms of model parameters, memory use, MACs, and time consumption in **Table 1**. The best time is for the BGNet[42] model with a value of 20 ms, and the second-best time is for our model (MSDE) with a value of 29.5 ms. Moreover, our model has the least number of parameters, the smallest memory footprint, and the least number of floating-point operations, with values of 48.2 K, 2.5 G, and 0.7 G, respectively.

Similarly, regarding the MACs reduction factors, our model surpasses the competition with reductions of 313×, 408×, 20.4×, 26× and 23× when compared to LEAStereo[10], PSMNet[8], BGNet[42], StereoNet[9] and SSDE, respectively. Furthermore, our model demonstrates significant memory efficiency with reduction factors of 24.7×, 13.7×, 1.4×, 2.8× and 3.1× compared to LEAStereo, PSMNet, BGNet, StereoNet and SSDE, respectively. Additionally, when comparing the number of parameters, our model showcases reduction factors of 37.5×, 108.4×, 61.7×, 13× and 12.2× in comparison to LEAStereo, PSMNet, BGNet, StereoNet and SSDE, respectively.

In terms of accuracy, **Table 2** illustrates the percentage of outliers on KITTI 2015 dataset. Although our model has the highest number of outliers among the compared models, it has the best and most efficient scores. Our MSDE falls second to the BGNet inference time of 20ms but is significantly smaller in size than the BGNet model which has 2974 K parameters and uses 1020 GMACs and 1 GB of memory.

Table 1. Comparative analysis of efficiency for state-of-the-art methods.

Method	**Efficient Metrics**				**GPU**
	Time (ms)	**Param. (K)**	**MACs (G)**	**Mem (G)**	
LEAStereo[10]	700 (↑ 23.7×)	1809.8 (↑ 37.5×)	782.5 (↑ 313×)	18 (↑ 25×)	A100 SXM4
PSMNet[8]	410 (↑ 13.9×)	5224.8 (↑ 108×)	1020 (↑ 408×)	10 (↑ 14×)	A100 SXM4
DeepPruner (fast)[11]	60 (↑ 3.00×)	7469.7 (↑ 155×)	218.7 (↑ 87×)	2.7 (↑ 3.8×)	Nvidia TitanX
BGNet[42]	25 (↓ 0.8×)	2974.8 (↑ 61.7×)	50.9 (↑ 20×)	1 (↑ 1.4×)	RTX 2080Ti
StereoNet[9]	31.1 (↑ 1.05×)	626.4 (↑ 13.0×)	64.9 (↑ 26×)	2 (↑ 2.8×)	RTX 2080Ti
Ours (SSDE)	26.7 (↓ 0.9×)	589.1 (↑ 12.2×)	57.6 (↑ 23×)	2.2 (↑ 3.1×)	RTX 2080Ti
Ours (MSDE)	29.5	48.2	2.5	0.7	RTX 2080Ti

4.2.2. Qualitative results

In this part, qualitative results are demonstrated after testing our model on three datasets: FlyingThings3D, KITTI 2012, and KITTI 2015. **Figure 6** compares our models to state-of-the-art models after running the network on our RTX 2080Ti GPU using KITTI 2012 and 2015 datasets. Our model MSDE doesn't have the best-visualized results among these models, but we think its findings are acceptable. The color bar under the figure indicates the depth of the objects in each output, from 1 to 192 disparity values. A ground truth label and colored stereo input images are presented in each to simplify the comparison.

Table 2. Quantitative evaluation of KITTI 2015 benchmark.

Method	D1-bg (%)	D1-fg (%)	D1-all (%)
LEAStereo[10]	1.4	2.91	1.65
AANet+[43]	1.65	3.96	2.03
BGNet+[42]	1.81	4.09	2.19
PSMNet[8]	1.86	4.62	2.32
BGNet[42]	2.07	4.74	2.51
DeepPruner (fast)[11]	2.32	3.91	2.59
FADNet[44]	2.5	3.1	2.6
DispSegNet[45]	4.2	16.97	6.33
StereoNet[9]	6.5	5.63	5.94
Ours (SSDE)	5.15	5.18	5.73
Ours (MSDE)	8.58	9.65	9.61

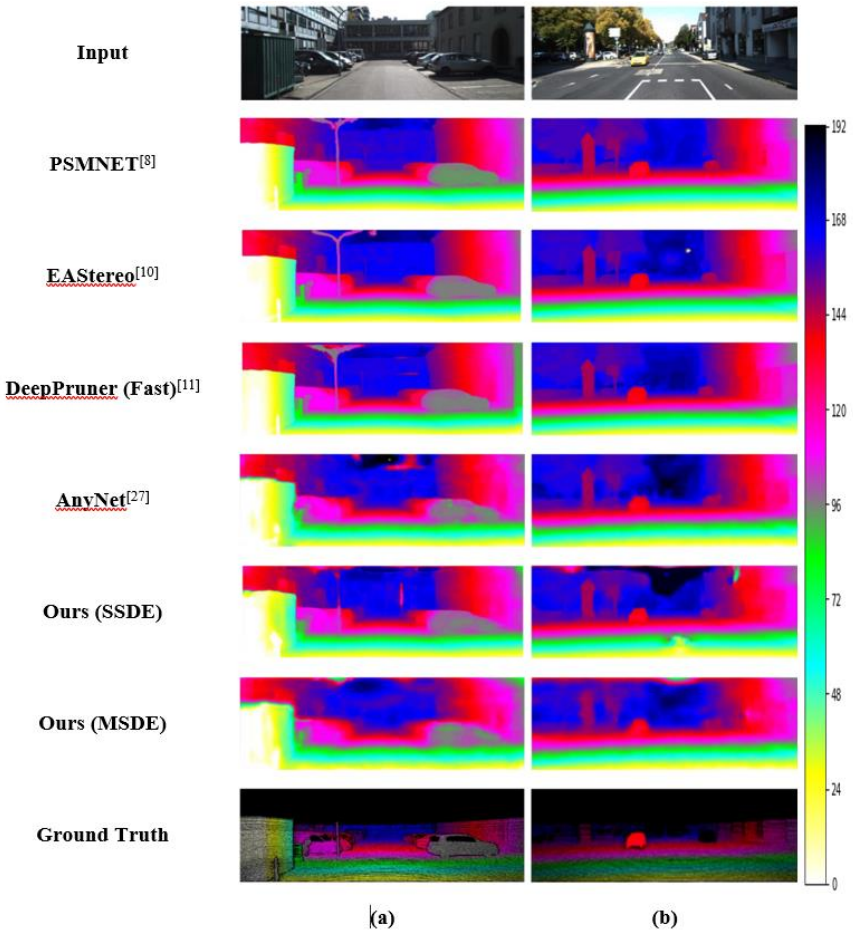

Figure 6. Qualitative results of our models compared with other state-of-art methods on **(a)** KITTI 2012 and **(b)** KITTI 2015.

4.3. Implementation and training

The MSDE is implemented using PyTorch[46] and trained with a batch size of 8 using RMSprop optimizer[47]. While FlyingThings3D images are kept in their original resolution, KITTI images are randomly cropped to 512×256 to create a random subset that enhances the generalization of our model. Input images are normalized between -1 and 1 before training.

The FPN in the MSDE model generates three feature maps at three scales $1/4$, $1/8$, and $1/16$ of the

full input image size. The MSDE model is trained from scratch on FlyingThings3D for 290 epochs on the FlyingThings3D dataset to construct a pre-trained model with an initial learning rate of 0.001 for the first 20 epochs. Thereafter, the learning rate decays by 0.9 every 10 epochs. For generalization on the KITTI dataset, we finetune the pre-trained model for 3500 epochs maintaining a learning rate of 0.001 throughout. Moreover, training and fine-tuning take 12 and 1 day, respectively, for MSDE on the NVIDIA GeForce RTX 2080Ti GPU. Additionally, the dataset split ratio is 80:20 for the KITTI dataset. That is, 80% of the images are used for training while the other 20% are used for testing.

4.4. Ablation study

Six ablation studies are performed on the FlyingThings3D dataset to choose the optimal variant of the factorized version of 3D convolution, as shown in **Table 3**. 2D indicates a 2D convolution, 1D means a 1D convolution, and BN represents a batch-normalized layer. This table shows the factorized layers of every ablation study and their kernel sizes. The factorized version consists of 2D and 1D convolutions with different numbers and orders.

This ablation study is made by replacing the cost filtering module in the StereoNet model which consists of five 3D convolutions with different factorized versions. Then, this new model is trained on the training pairs of the Flyingthings3D dataset using multiple permutations of factorized 3D convolution in the FCF module. After that, it is evaluated on the testing pairs of the FlyingThings3D dataset as shown in **Table 4**. This table compares the EPE and outliers values for the output of stage 0 (initial disparity map) and stage 2 (the final result) for each ablation study. The best option is AB6, which consists of one 2D convolution with a kernel size of $3 \times 3 \times 1$ and two 1D convolutions with a kernel size of $1 \times 1 \times 3$, plus a batch-normalized BN layer at the end. **Table 4** shows that the EPE and outliers for AB6 are 1.842% and 10.41% for the final output.

Table 3. Ablation studies using different permutations of factorizing 3D convolution.

Ablation Study #	Layers of the factorized 3D Convolution	Kernel size for each convolution
AB1	2D + 1D + BN	(1 × 3 × 3) + (3 × 1 × 1)
AB2	2D + BN + 1D	(1 × 3 × 3) + (3 × 1 × 1)
AB3	2D + BN + 1D + BN	(1 × 3 × 3) + (3 × 1 × 1)
AB4	1D + 1D + 1D + BN	(1 × 3 × 1) + (1 × 1 × 3) + (3 × 1 × 1)
AB5	2D + 1D + 1D + BN	(3 × 3 × 1) + (3 × 1 × 1) + (3 × 1 × 1)
AB6	2D + 1D + 1D + BN	(3 × 3 × 1) + (1 × 1 × 3) + (1 × 1 × 3)

Table 4. The evaluation results of ablation studies on the FlyingThings3D test dataset.

Ablation Study #	EPE (D1-all) %		Outliers (D1-all) %	
	Stage 0 (Initial Disparity Map)	Stage 2 (Final Output)	Stage 0 (Initial Disparity Map)	Stage 2 (Final Output)
AB1	4.153	1.98	25.029	11.206
AB2	3.897	1.887	23.446	11.022
AB3	4.032	1.991	24.536	11.209
AB4	4.553	2.129	26.742	12.581
AB5	3.906	1.866	24.057	11.301
AB6	3.897	1.842	23.446	10.41

5. Conclusion

We presented our MSDE—an efficient end-to-end neural architecture design for disparity estimation,

which represents a trade-off between accuracy and efficiency. To this end, we used multiple low-dimensional cost volumes at different scales along with residual disparity to predict disparity maps in a highly efficient manner. Utilizing the factorization technique, our MSDE decomposed 3D convolutions into 2D and 1D convolutions for filtering the constructed cost volumes achieving low computations and low memory requirements. Although our proposed model did not achieve the same level of accuracy as state-of-the-art methods, the inference time of the MSDE was on par with the top models and stood out for its real-time performance and efficiency with fewer parameters, floating-point operations, and memory footprint compared to state-of-the-art approaches. This drives MSDE as a highly suitable option for predicting disparity maps with limited computational resources and memory availability.

Further work is needed to improve the accuracy of our proposed models and reduce the number of outliers using different approaches, such as introducing occlusion blocks and utilizing an attention block after the fast cost filtering module.

Author contributions

Conceptualization, AA, MRAK and RB; methodology, MRAK; software, AA; validation, RB and MRAK; formal analysis, AA, MRAK and RB; investigation, MRAK and RB; resources, MRAK and RB; data curation, RB; writing—original draft preparation, AA; writing—review and editing, MRAK and RB; visualization, AA and MRAK; supervision, DS and WA; project administration, DS and WA; funding acquisition, DS and WA. All authors have read and agreed to the published version of the manuscript.

Funding

This work was partially funded by the Federal Ministry of Education and Research Germany under the project DECODE (01IW21001).

Acknowledgments

we would like to thank the German Research Centre for Artificial Intelligence (DFKI) for providing the needed environment and supporting us in this research.

Conflict of interest

The authors declare no conflict of interest.

Abbreviation

MSDE, multi-scale disparity estimation; FLOPS, floating-point operations; FPGA, field-programmable gate arrays; CNN, convolutional neural network; LSTM, long short-term memory; NAS, neural architecture search; FPN, feature pyramid network; BN, batch normalization; CV, cost volume; ICV, initial cost volume; RCV, residual cost volume; ECV, error cost volume; RECV, residual and error cost volume; FCF, fast cost filtering; DR, disparity regression; EPE, end point error.

References

1. O'Riordan A, Newe T, Dooly G, et al. Stereo Vision Sensing: Review of existing systems. International Conference on Sensing Technology (ICST), 2018. doi: 10.1109/icsenst.2018.8603605
2. Mendoza Guzmán VM, Mejía Muñoz JM, Moreno Márquez NE, et al. Disparity map estimation with deep learning in stereo vision. CEUR Workshop Proc. 2018, 2304: 27–40.
3. Hamid MS, Manap NA, Hamzah RA, et al. Stereo matching algorithm based on deep learning: A survey. Journal of King Saud University—Computer and Information Sciences. 2022. doi: 10.1016/j.jksuci.2020.08.011
4. Zhou K, Meng X, Cheng B. Review of Stereo Matching Algorithms Based on Deep Learning. Computational Intelligence and Neuroscience, 2020. doi: 10.1155/2020/8562323

5. Luo W, Schwing AG, Urtasun R. Efficient Deep Learning for Stereo Matching. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. doi: 10.1109/cvpr.2016.614
6. Zbontar J, LeCun Y. Computing the stereo matching cost with a convolutional neural network. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015. doi: 10.1109/cvpr.2015.7298767
7. Hirschmuller H. Stereo Processing by Semiglobal Matching and Mutual Information. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2008. doi: 10.1109/tpami.2007.1166
8. Chang JR, Chen YS. Pyramid Stereo Matching Network. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018. doi: 10.1109/cvpr.2018.00567
9. Khamis S, Fanello S, Rhemann C, et al. StereoNet: Guided Hierarchical Refinement for Real-Time Edge-Aware Depth Prediction. Lecture Notes in Computer Science, 2018. doi: 10.1007/978-3-030-01267-0_35.
10. Cheng X, Zhong Y, Harandi M, et al. Hierarchical Neural Architecture Search for Deep Stereo Matching. arXiv. 2020, arXiv:2010.13501.
11. Duggal S, Wang S, Ma WC, et al. DeepPruner: Learning Efficient Stereo Matching via Differentiable PatchMatch. IEEE/CVF International Conference on Computer Vision (ICCV), 2019. doi: 10.1109/iccv.2019.00448
12. Knobelreiter P, Reinbacher C, Shekhovtsov A, et al. End-to-End Training of Hybrid CNN-CRF Models for Stereo. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. doi: 10.1109/cvpr.2017.159.
13. Seki A, Pollefeys M. SGM-Nets: Semi-Global Matching with Neural Networks. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. doi: 10.1109/cvpr.2017.703
14. Liang Z, Feng Y, Guo Y, et al. Learning for Disparity Estimation Through Feature Constancy. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018. doi: 10.1109/cvpr.2018.00297
15. Zhong Y, Dai Y, Li H. Self-Supervised Learning for Stereo Matching with Self-Improving Ability. ArXiv. 2017.
16. Mayer N, Ilg E, Hausser P, et al. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. doi: 10.1109/cvpr.2016.438
17. Pang J, Sun W, Ren JSJ, et al. Cascade Residual Learning: A Two-Stage Convolutional Neural Network for Stereo Matching. IEEE International Conference on Computer Vision Workshops (ICCVW), 2017. Published online October 2017. doi: 10.1109/iccvw.2017.108
18. Song X, Zhao X, Hu H, et al. EdgeStereo: A Context Integrated Residual Pyramid Network for Stereo Matching. Lecture Notes in Computer Science, 2019. doi: 10.1007/978-3-030-20873-8_2
19. Dovesi PL, Poggi M, Andraghetti L, et al. Real-Time Semantic Stereo Matching. IEEE International Conference on Robotics and Automation (ICRA), 2020. doi: 10.1109/icra40945.2020.9196784
20. Kendall A, Martirosyan H, Dasgupta S, et al. End-to-End Learning of Geometry and Context for Deep Stereo Regression. IEEE International Conference on Computer Vision (ICCV), 2017. doi: 10.1109/iccv.2017.17
21. Lu C, Uchiyama H, Thomas D, et al. Sparse Cost Volume for Efficient Stereo Matching. Remote Sensing, 2018. doi: 10.3390/rs10111844
22. Tulyakov S, Ivanov A, Fleuret F. Practical deep stereo (PDS): Toward applications-friendly deep stereo matching. Adv Neural Inf Process Syst, 2018, 5871–5881.
23. Zhang F, Prisacariu V, Yang R, et al. GA-Net: Guided Aggregation Net for End-To-End Stereo Matching. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019. doi: 10.1109/cvpr.2019.00027
24. Guo X, Yang K, Yang W, et al. Group-Wise Correlation Stereo Network. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019. doi: 10.1109/cvpr.2019.00339
25. Jie Z, Wang P, Ling Y, et al. Left-Right Comparative Recurrent Model for Stereo Matching. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018. doi: 10.1109/cvpr.2018.00404
26. Shi X, Chen Z, Wang H. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. arXiv. 2015, arXiv:1506.04214v2.
27. Wang Y, Lai Z, Huang G, et al. Anytime Stereo Image Depth Estimation on Mobile Devices. International Conference on Robotics and Automation (ICRA), 2019. doi: 10.1109/icra.2019.8794003
28. Huang Z, Norris TB, Wang P. ES-Net: An Efficient Stereo Matching Network. Available online: http://arxiv.org/abs/2103.03922 (accessed on 13 October 2021).
29. He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. doi: 10.1109/cvpr.2016.90
30. Qiu Z, Yao T, Mei T. Learning Spatio-Temporal Representation with Pseudo-3D Residual Networks. IEEE International Conference on Computer Vision (ICCV), 2017. doi: 10.1109/iccv.2017.590
31. Tran D, Wang H, Torresani L, et al. A Closer Look at Spatiotemporal Convolutions for Action Recognition. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018. doi: 10.1109/cvpr.2018.00675
32. Gonda F, Wei D, Parag T, Pfister H. Parallel separable 3d convolution for video and volumetric data understanding. ArXiv. 2018.
33. Li X, Lai T, Wang S, et al. Weighted Feature Pyramid Networks for Object Detection. IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), 2019. doi: 10.1109/ispa-bdcloud-sustaincom-socialcom48970.2019.00217

34. Gao Q, Zhou Y, Li G, et al. Compact StereoNet: Stereo Disparity Estimation via Knowledge Distillation and Compact Feature Extractor. IEEE Access, 2020. doi: 10.1109/access.2020.3029832
35. Kang J, Chen L, Deng F, et al. Improving disparity estimation based on residual cost volume and reconstruction error volume. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2020. doi: 10.5194/isprs-archives-xliii-b2-2020-135-2020
36. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 32nd International Conference on Machine Learning. ICML, 2015.
37. Maas AL, Hannun AY, Ng AY. Rectifier Nonlinearities Improve Neural Network Acoustic Models, 2013.
38. Barron JT. A General and Adaptive Robust Loss Function. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019. doi: 10.1109/cvpr.2019.00446
39. Geiger A, Lenz P, Urtasun R. Are we ready for autonomous driving? The KITTI vision benchmark suite. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012. doi: 10.1109/cvpr.2012.6248074
40. Menze M, Heipke C, Geiger A. Joint 3D estimation of vehicles and scene flow. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2015. doi: 10.5194/isprsannals-ii-3-w5-427-2015
41. Liang Z, Guo Y, Feng Y, et al. Stereo Matching Using Multi-Level Cost Volume and Multi-Scale Feature Constancy. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2021. doi: 10.1109/tpami.2019.2928550
42. Xu B, Xu Y, Yang X, et al. Bilateral Grid Learning for Stereo Matching Networks. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021. doi: 10.1109/cvpr46437.2021.01231
43. Xu H, Zhang J. AANet: Adaptive Aggregation Network for Efficient Stereo Matching. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020. doi: 10.1109/cvpr42600.2020.00203
44. Wang Q, Shi S, Zheng S, et al. FADNet: A Fast and Accurate Network for Disparity Estimation. IEEE International Conference on Robotics and Automation (ICRA), 2020. doi: 10.1109/icra40945.2020.9197031
45. Zhang J, Skinner KA, Vasudevan R, et al. DispSegNet: Leveraging Semantics for End-to-End Learning of Disparity Estimation From Stereo Imagery. IEEE Robotics and Automation Letters, 2019. doi: 10.1109/lra.2019.2894913
46. Paszke A, Gross S, Chintala S, et al. Automatic differentiation in pytorch. 2017.
47. Lyon RF. Neural Networks for Machine Learning. Human and Machine Hearing. 2017. doi: 10.1017/9781139051699.031