MobileStereoNet: Towards Lightweight Deep Networks for Stereo Matching

Faranak Shamsafar*, Samuel Woerz*, Rafia Rahim, Andreas Zell
WSI Institute for Computer Science, University of Tuebingen, Germany

Abstract

Recent methods in stereo matching have continuously improved the accuracy using deep models. This gain, however, is attained with a high increase in computation cost, such that the network may not fit even on a moderate GPU. This issue raises problems when the model needs to be deployed on resource-limited devices. For this, we propose two light models for stereo vision with reduced complexity and without sacrificing accuracy. Depending on the dimension of cost volume, we design a 2D and a 3D model with encoder-decoders built from 2D and 3D convolutions, respectively. To this end, we leverage 2D MobileNet blocks and extend them to 3D for stereo vision application. Besides, a new cost volume is proposed to boost the accuracy of the 2D model, making it performing close to 3D networks. Experiments show that the proposed 2D/3D networks effectively reduce the computational expense (27%/95% and 72%/38% fewer parameters/operations in 2D and 3D models, respectively) while upholding the accuracy. Code: `https://github.com/cogsys-tuebingen/mobilestereonet`.

1. Introduction

Stereo matching is one of the techniques for depth perception of a scene, which is established based on the displacement of the matching points in a binocular camera setup. Given a pair of rectified left/right images, we can compute depth by redirecting to disparity map estimation. Depth prediction is used in many real-world applications, like self-driving cars [15], robotics [26], and object detection [25]. Compared to other techniques for depth perception, like LiDAR and Time-of-Flight sensors, passive stereo vision is more desirable in real-world scenarios because of inherent problems in other techniques, such as sparsity of depth data, incompetency in sunlight or reflective/absorbing surfaces, and limited operating depth range.

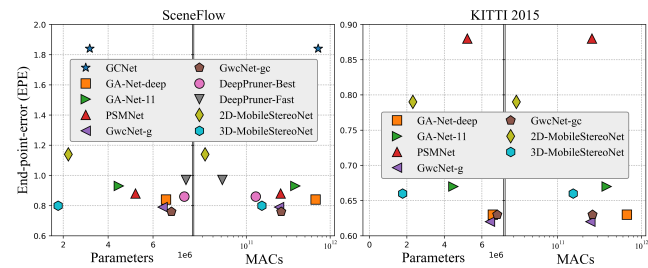A stereo matching algorithm has three main components:

Figure 1: Performance *vs*. computation cost on SceneFlow test set (*Left*) and KITTI 2015 validation set (*Right*): Metrics are EPE, number of parameters ($\times 10^6$), and number of operations (MACs in log scale). For all, the lower is better. By using a new parameterized cost volume, 2D-MobileStereoNet shows closer performance to 3D models with the least MACs. 3D-MobileStereoNet obtains competitive accuracy with the least number of parameters.

feature extraction, regularization, and disparity selection. Before deep learning, numerous algorithms proposed different schemes for each step, like local descriptors (Sum of Absolute Difference or Census Transform [30]) for feature extraction, Semi-global Matching (SGM) [7] for regularization, and winner-take-all (WTA) for final disparity selection. Nowadays, similar to other computer vision tasks, stereo matching has also benefited from deep networks. Primarily, the research incorporated deep learning in individual components of the pipeline, like [31, 17] in matching costs and [18, 24] in regularization. Later, the research stirred towards end-to-end frameworks, taking all the fundamental components into one network [13, 10, 2, 6].

Accordingly, in end-to-end pipelines, depending on the dimension of the cost volume built on top of the unary features, the subsequent convolutional layers in the regularization part (encoder-decoder) can be either 2D convolutions (for 3D cost volume) [13] or 3D convolutions (for 4D cost volume) [10]. We dub the former and the latter group as "2D" and "3D" models, respectively. While the 3D end-to-end networks introduce highly boosted accuracy in disparity estimation, they are computationally costly due to more complex networks. As such, some of these networks cannot fit even on a moderate GPU, raising an out-of-memory (OOM) state. On the other hand, there are many embedded platforms with memory constraints on which the networks

---

should fit and execute efficiently.

In order to reduce the complexity of 3D models, some works reconsider the configuration of the networks. For instance, DeepPruner [4] develops a PatchMatch module to ignore the cost volume evaluation for most of the disparity range. In [29], authors establish an aggregation module on top of a cost volume computed by traditional local descriptors. Thus, skipping the convolutional feature extraction, the network benefits from a lighter learning paradigm. Also, it creates a 3D cost to avoid the curse of 3D convolutions.

In this work, we develop two end-to-end stereo matching networks, which exploit MobileNet-V1 [8] and MobileNet-V2 [23] blocks to mitigate the computational burden in favor of real embedded platforms, like FPGAs or mobile devices. Depending on the cost volume dimension, *i.e.* 3D or 4D, we propose a 2D and a 3D network. Moreover, for the 3D cost volume, a new learning construction module is devised based on interlacing the features from two viewpoints. Although reducing the computation cost is usually accompanied by a degradation in performance, we show that the proposed architectures are competitive with their state-of-the-art counterparts (Fig. 1).

Overall, our main contributions are as follows: i) Two lightweight models (2D/3D) are designed and proposed for stereo matching using MobileNet blocks without sacrificing accuracy. ii) We raise MobileNet blocks from the originally proposed 2D convolutions to 3D for the application of stereo matching. Also, by analyzing their costs, we prove their merit in reducing the computational load when processing 4D data. iii) We introduce a learnable cost volume module for the 2D model to keep the accuracy comparable with over-parameterized 3D models. iv) Extensive experiments for analyzing the accuracy/complexity trade-off in different design choices are conducted. Our findings in the design choice can be applicable to similar 2D/3D networks to reduce their complexity.

2. Related work

Stereo vision is one of the popular techniques for estimating the depth from images. In the last decade, machine learning and deep learning approaches have well-progressed in computer vision tasks, including stereo matching. Deep learning-based methods can be categorized into two groups: methods that focus on transferring only one or some of the general pipeline components into a deep learning framework [31, 24, 1], and approaches that formulate the whole process in an end-to-end scheme [13, 10, 2, 6, 32]. Most of the end-to-end methods are developed based on 3D convolutional layers. Although these architectures achieve a substantial increase in accuracy, they require a high amount of memory usage, making them impractical for mobile and real-time applications, such as robotics and autonomous vehicles.

Lighter networks. Lightweight architectures have become an active research domain, ringing the bell that it is getting impractical to slide through complex networks without considering the load of computations. Generally, convolutions entail the most considerable computational load. Some deep networks for stereo reconstruction have been developed to achieve less complexity while being competitive in terms of accuracy with heavy 3D architectures. In [29], an initial matching cost is constructed based on the traditional cost computation. After reducing the channel dimension through 1×1 convolutions, the data is fed into a U-Net to regress the disparity map. In another work, DeepPruner [4] mitigates the computational complexity of 3D convolutional layers by calculating the matching cost for a subset of possible disparity values. Recently, [20] proposed to use feature-wise and feature-disparity-wise separable convolutions for optimizing the 3D stereo models. On the whole, according to the results in [29], the 2D architectures can make a better trade-off between accuracy and speed.

Other works mainly focus on reducing the complexity of 3D convolutional layers for other 3D vision tasks. Qiu *et al*. [19] developed pseudo-convolutions that decouple a 3D convolution into 3D convolutions equivalent to a 2D convolution and a light 3D convolution, which aggregates information only across the third domain. The network is demonstrated on video classification. In [28], authors made use of 3D depth-wise convolutions for 3D reconstruction. Recently, [5] proposed progressively forward expanding a 2D tiny network along multiple axes to get fewer parameters for video classification and detection.

Cost volume computation. In a stereo model, measuring the similarity of left/right features is just as important as feature extraction and regularization. Traditional algorithms utilized simple calculations, like absolute difference, Hamming distance, or correlation. Similar solutions carried on to deep learning-based networks. Specifically, correlation is used on top of unary features to compute a 3D cost volume [3, 12, 13, 31, 9]. Later, Kendall *et al*. [10] proposed to concatenate unary features to make a 4D cost, requiring 3D convolutions in the following. After that, this approach was mainly adapted for 3D models with some modifications to enhance the accuracy, *e.g.* variance-based [21], group-wise correlation [6] and pyramid [27] cost volume.

3. Methodology

As a first step, we reformulate two common light blocks [8, 23] to raise them from 2D convolutions to 3D for the application of stereo matching. We also analyze their computation cost w.r.t. the standard 2D/3D convolution counterparts. The cost of a deep network is measured by the number of operations in MACs (Multiply-Accumulate) and the number of parameters. While the number of parameters is fixed for a model, MACs depend on the input size.
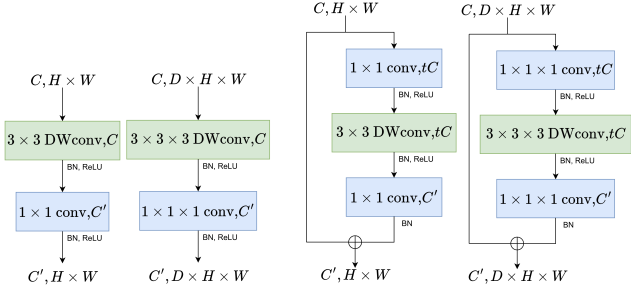
Figure 2: *Left*: MobileNet-V1 block and its extension to 3D, *Right*: MobileNet-V2 block and its extension to 3D.

Operator		Operations – MACs $((D\times)H \times W\times)$	Example for Reduction Factor
2D	Std. Conv.	$k \times k \times C_{in} \times C_{out}$	-
	v_1 Block	$(k \times k \times C_{in})+$ $(C_{in} \times C_{out})$	7.9x
	v_2 Block	$(C_{in} \times t \times C_{in})$ $(k \times k \times t \times C_{in})+$ $(t \times C_{in} \times C_{out})$	2.7x
3D	Std. Conv.	$k \times k \times k \times C_{in} \times C_{out}$	-
	v_1 Block	$(k \times k \times k \times C_{in})+$ $(C_{in} \times C_{out})$	18.9x
	v_2 Block	$(C_{in} \times t \times C_{in})$ $(k \times k \times k \times t \times C_{in})+$ $(t \times C_{in} \times C_{out})$	7.0x

Table 1: Computation cost of the standard convolutions *vs.* MobileNet-V1 (v_1) and MobileNet-V2 (v_2) blocks in 2D and 3D counterparts. In MACs, $(D\times)$ belongs to 3D types. We have ignored the computation cost of batch normalization, ReLU and residual connection of v_2. Example for reduction factor (w.r.t. the standard convolution counterparts) is computed for $k = 3$, $C_{in} = 32$, $C_{out} = 64$, and $t = 2$.

3.1. Light blocks replacing 2D/3D convolutions

As a pioneer work, MobileNet-V1 [8] employs depthwise and point-wise convolutions to produce an output with the same size as the output of a standard convolution, but with fewer computations. Later, MobileNet-V2 [23] was introduced, which formulates its block with point-wise, depth-wise, and once again, point-wise layers. The number of input and output channels are specific for each layer, such that the channel dimension is expanded with an expansion factor (t) within the block. In a non-downsampling layer, a skip connection is included as well, making it a so-called *Inverted Residual* block. In Fig. 2, MobileNet-V1 and MobileNet-V2 blocks (shortened to $v1$ and $v2$ hereon, for simplicity) are shown.

Raising MobileNet blocks to 3D for stereo network. Originally, $v1$ and $v2$ blocks are designed to replace 2D convolutions and are proved mainly for sparse prediction tasks, like image classification. Still, many other computer vision problems require 3D convolutions to operate on 4D data, *e.g.* tasks with temporal input besides the spatial data. Likewise, dense disparity estimation is such a topic, which explores the space for the 3rd dimension of the scene. This outlook of using 3D convolutions for stereo matching has emerged recently, where they are utilized for processing 4D cost data. Hence, we take $v1$ and $v2$ blocks to their 3D counterparts for stereo vision application and show their necessity for light stereo vision models.

For this, just as in 2D convolutions, we commit the depth-wise and point-wise convolutions in the channel (feature) dimension in 3D convolutions. To be more precise, to raise the convolutions from 2D to 3D, the input data is extended from $C \times H \times W$ to $C \times D \times H \times W$, where H and W indicate the input height and width, respectively; D is the new third dimension, and C is the number of channels. In our formulation for stereo matching, the 4D data of cost volume [10, 27, 6] is likewise of size $C \times D \times H \times W$, where D is the predefined disparity range for building the cost volume. Accordingly, we raise kernels from 2D to 3D, considering the same size for the added dimension, *e.g.* if the 2D kernel is 3×3, the 3D kernel is $3 \times 3 \times 3$. Therefore, taking $v1$ and $v2$ blocks to 3D is straightforward by apply-

ing depth-wise and point-wise convolutions in the channel dimension. Figure 2 displays the new $v1$ and $v2$ blocks raised to 3D.

In Tab. 1, we compute the cost of the standard 2D/3D convolutions and 2D/3D $v1$ and $v2$ blocks. Comparing to standard convolution counterparts, there is a reduction factor in computation cost that depends on the kernel size k, channels (C_{in} and C_{out}) and expansion factor (t) in $v2$ blocks. The example in the last column shows that $v1$ blocks are lighter than $v2$ in both 2D and 3D versions, as expected. Moreover, exploiting $v1$ and $v2$ blocks in 3D type is more desirable as they show the capability for further reduction in operations compared to standard counterparts.

Additionally, we examine the impact of the expansion factor of v_2 blocks in Fig. 3. We consider the common cases in an hourglass module (*e.g.* in [2, 6]) for evaluation, *i.e.* $C_{in} = C_{out} = \{32, 64, 128\}$ with $k = 3$. We see that by increasing t, the reduction factor decreases until a point where the block becomes heavier than a standard convolution counterpart. Also, the 2D block is more sensitive to t, such that the cost is increased beyond the convolution after $t = 5$. Here, once again, we can verify the merit of MobileNet architectures in our reformulation for 3D convolutional layers of 3D networks.

3.2. Proposed models

Here, we describe two end-to-end baselines (2D and 3D) and design MobileStereoNets in reliance on these two models and 2D/3D $v1$ and $v2$. Following the common pipeline in recent work, we first feed the rectified left/right images to the feature extraction backbone and obtain the unary features. The backbone is shared for the left and right images. The results are passed into a cost volume construction module to merge the data from two viewpoints. Finally, an
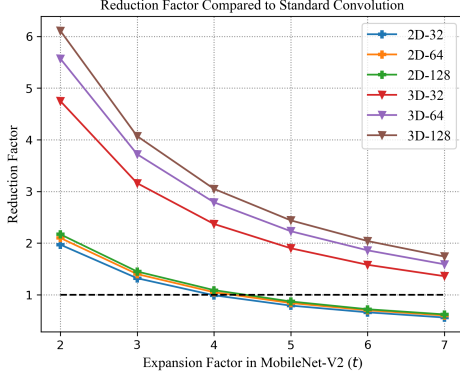
Figure 3: Reduction factor of 2D/3D $v2$ blocks with varying expansion factor (t) w.r.t. standard convolution counterparts. The right hand numbers indicate the channel numbers ($C_{in} = C_{out}$ and $k = 3$).
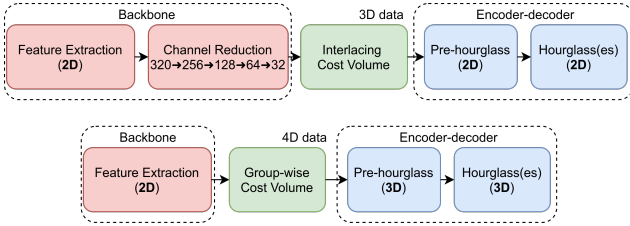

Figure 4: *Top* (2D Baseline): After feature extraction, data is reduced to 32 channels. A 3D cost volume, $(d_{max}/4) \times (H/4) \times (W/4)$, is generated using the proposed interlacing cost volume construction and it is processed by 2D convolutions. *Bottom* (3D Baseline): A 4D cost volume, $40 \times (d_{max}/4) \times (H/4) \times (W/4)$, is computed using Gwc40 [6] and it is processed with 3D convolutions.

encoder-decoder (hourglass) is applied on top of the cost volume to estimate the disparity map.

3D baseline. For this case, we adopt GwcNet-g [6] with only one hourglass (Fig. 4) as it is performing superior to other similar designs, *e.g.* GCNet [10], PSMNet [2], and GA-Net [32]. This baseline utilizes a ResNet-like backbone and an encoder-decoder with 3D convolutions. Namely, a 4D cost volume is constructed by group-wise correlation of unary features, requiring 3D convolutions afterward. The encoder-decoder consists of an hourglass [2, 6] outputting a downsampled disparity map, which after upsampling is compared against the ground-truth with a smooth-$L1$ loss function. For the detailed architecture, we refer the reader to the appendix.

2D baseline. In order to develop a much lighter stereo network, we modify the 3D baseline such that it uses an encoder-decoder with 2D convolutions (Fig. 4). This approach contrasts with the recent trend, where 3D convolutions are deployed to add a feature dimension for disparity via a 4D cost volume. With a ResNet-like backbone similar to the 3D baseline, an input image with resolution $H \times W$ is turned into a feature of size $320 \times (H/4) \times (W/4)$. We

add further processing by four successive point-wise convolutions to reduce the number of channels and attain a size of $32 \times (H/4) \times (W/4)$. In order to aggregate two unary features to form a cost volume, which indicates a similarity measurement in the left/right images across the disparity dimension, we propose a new *Interlacing Cost Volume*. Note that we need a 3D cost volume to retain the encoder-decoder with 2D convolutions. Ignoring the feature dimension for disparity, 3D cost volume is of size $(d_{max}/4) \times (H/4) \times (W/4)$, where d_{max} is the maximum disparity level. Finally, the 3D cost volume is taken into the encoder-decoder module after passing through two convolutions as a pre-hourglass module.

Interlacing cost volume construction. Traditionally, a cost volume for stereo matching is computed for comparing the descriptors of binocular images across the disparity dimension, mainly as *3D data* as following:

$$C_{3D}(d, x, y) = G(f_L(x, y), f_R(x - d, y)), \quad (1)$$

where (x, y) and d are the spatial location and the disparity value within a range of $(0, d_{max})$, respectively. $f_R(x - d, y)$ is the traversed right feature for a specific disparity level. $G(., .)$ indicates a similarity measurement function that was conventionally chosen as correlation or Hamming distance. With the advent of deep learning in stereo vision, as the unary features raised into 3D data, *i.e.* $f(x, y) \to f(., x, y)$, DispNetC [13] proposed to use a correlation layer (dot product) to merge these data by $f_L(., x, y) \odot f_R(., x - d, y)$. Later, [10] introduced a cost volume as *4D data*, $C_{4D}(d, ., x, y)$, by concatenating the unary features.

These 3D or 4D cost volumes are obtained in an unparameterized module of the network. Thus, we propose a subnetwork, named *Interlacing Cost Volume Construction* with the motivations as following: i) In order to achieve a better aggregation of the two unary features, we propose a parameterized subnetwork to learn the aggregation. ii) By interlacing the left/right unary features, the corresponding features maps are distilled by a kernel. iii) we aim to retain the encoder-decoder with only 2D convolutions (and not 3D, which is the case in recent work), as this can contribute to significant reduction of operations.

Namely, given the left and traversed right unary features, each of size $C \times H \times W$, we first interlace them across the channel dimension to form a data with double-sized channels, $2C \times H \times W$ (Fig. 5). After unsqueezing the output and raising it to 4D data ($1 \times 2C \times H \times W$), a 3D convolutional layer is applied such that the 3D kernels cover a specific number of left/right feature pairs. That is, a kernel of size $2i \times 3 \times 3$ covers i channel from each of the unary features. By increasing the i, the kernel covers more features and thus, integrates more information. The two following layers (Fig. 5) are also 3D convolution with double and same number of kernels' of the first layer. In general, the kernels are of size $m \times 3 \times 3$ with strides as $m \times 1 \times 1$,
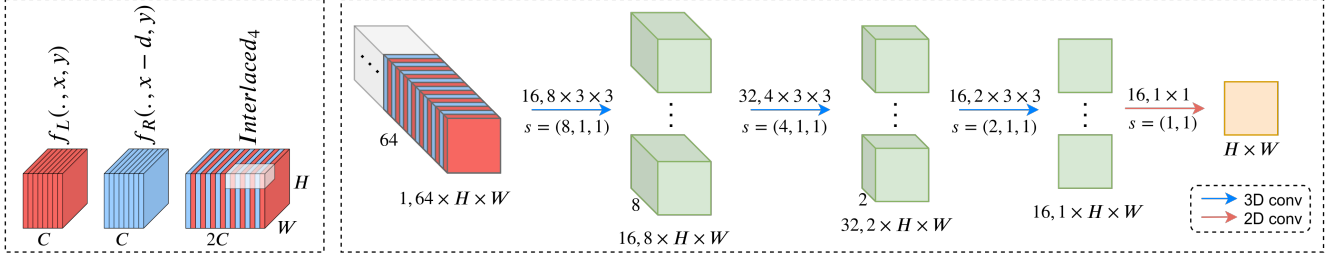
Figure 5: *Left*: Interlacing cost volume construction at a particular disparity from the left (red) and right (blue) unary features. The kernels of the first layer take a group of non-overlapping interlaced features (here, four channels from each unary feature, $Interlaced_4$). *Right*: Processing the data with more convolutional layers to yield the aggregated feature for that disparity level. The numbers above and below the arrows indicate the kernel size and the stride, respectively.

showing that they are covering non-overlapping channels. Finally, we convert the data to a single channel and pass through one 2D convolutional layer. Note that similar to other methods for cost volume, the spatial resolution is unchanged. We can write the general formula for a certain disparity level as Eg. 2. In Sec. 4, we show that the inclusion of learnable weights in stereo network contributes to better aggregation of the left/right features.

$$C_{3D}(d,x,y) = \text{Interlace}\{f_L(.,x,y), f_R(.,x-d,y)\} \quad (2)$$

MobileStereoNets. In our baselines, the feature extraction and channel reduction are essentially processed with 2D convolutions. On the other hand, the hourglass employs 2D or 3D convolutions depending on the constructed cost volume. In order to obtain lighter networks, we replace these components with 2D and 3D counterparts of $v1$ and $v2$ blocks. There are different design choices when exploiting these blocks in individual modules of the networks. Thus, extensive experiments are conducted to answer the following questions: i) Can we replace different modules in the 2D/3D baselines with 2D/3D $v1$ and $v2$ to achieve lighter stereo networks and keep the error rates low? ii) If so, which modules should be replaced with them for a better compromise? iii) Which block type performs better in terms of accuracy and computational load?

Our experiments (*cf.* Sec. 4) lead to our MobileStereoNets by modifying the baselines as follows:

- First Convolutions: Each of the three initial 3×3 convolutions are replaced with one $v2$. Using an expansion factor of $t = 3$ provides a favorable trade-off between the performance and computational complexity.
- Feature Extraction: We retained the original layer architecture and block structure, consisting of two 3×3 convolutions and a residual connection between each block. Substituting these convolutions with $v1$ keeps performance competitive while reducing the computational complexity significantly.
- Channel Reduction: In the 2D baseline, we keep this module, *i.e.* four 1×1 convolutions, unchanged with stan-

dard convolutions as replacing them with lighter blocks deteriorates the performance.
- Pre-hourglass: In the 2D baseline, we replace the two 3×3 convolutions in both of the blocks with $v2$. For 3D-MobileStereoNet, we use our extension of $v2$ to 3D instead of 3D convolutions. In both models, we choose expansion factor as $t = 3$.
- Hourglass: We employ a stack of three hourglasses for both 2D and 3D models. While the 3D network uses the same channel dimension as GwcNet [6] (32), the hourglass width is increased to 48 in the 2D model. 2D-MobileStereoNet uses $v2$ instead of 2D convolutions. In 3D-MobileStereoNet, we once again swap the 3D convolutions for our extension of $v2$ to 3D. For both models, the expansion factor is $t = 2$.

4. Experimental results and discussion

Here, we first evaluate the performance of the proposed interlaced cost volume. Then, extensive experiments for taking the 2D/3D $v1$ and $v2$ blocks into the baselines are elaborated to show the path taken to reach the final architectures. Finally, we compare our developed networks with other methods. Note that d_{max} is 192 in all cases.

In order to analyze different design choices, we use the SceneFlow "final pass" dataset [13], consisting of 35,454/4,370 training/test samples with 540×960 resolution. This dataset can also help to pretrain the networks for limited real datasets. The quantitative evaluation for SceneFlow images is mainly reported with End-Point-Error (EPE), the mean average disparity error in px. Two more errors are also reported, *i.e.* px-3 and D1, which are percentages of the outliers with disparity errors larger than $3px$ and $\max(3px, 0.05 \times \text{ground-truth})$, respectively.

4.1. Cost volume construction

To verify the performance of the interlacing cost volume, we replace the corresponding module in 2D baseline (Fig. 4) with correlation, which is adopted by 2D networks [3, 13, 9]. Table 5 shows the evaluation of this baseline

Method	EPE(px)$\downarrow$	D1(%)$\downarrow$	px-3(%)$\downarrow$
$Concatenation$	1.86	7.46	8.48
$Correlation$	1.71	6.80	7.84
$Interlaced_1$	1.70	6.20	**7.06**
$Interlaced_2$	1.61	6.39	7.31
$Interlaced_4$	**1.55**	**6.15**	**7.06**
$Interlaced_8$	1.64	6.41	7.35
$Interlaced_{16}$	1.73	6.65	7.58

Table 2: Performance evaluation on SceneFlow test set for the proposed 3D cost volume: The costs created by $Interlaced_i$ contribute to lower error rates.

against the model embedded with our interlacing cost volume. $Interlaced_i$ indicates that i channels are taken from each unary feature, and the kernel of the subsequent layer is of size $2i \times 3 \times 3$. For instance, when $i = 4$, four channels from each feature data are combined by $8 \times 3 \times 3$ kernels. Therefore, cases of $i > 1$ can be interpreted as *group-wise* interlacing. We observe that better similarity measurements between the left/right features are achieved by introducing this learnable cost volume construction, resulting in lower EPE. According to the table, the best case is achieved when $i = 4$ (also depicted in Fig. 5), and hence, we consider this case for further experiments in the 2D baseline.

We also investigate the effect of interlacing against direct concatenation of left/right unary features. According to the table, the error has increased in this case, showing that direct concatenation is not efficiently distilling the corresponding left/right features, which is essential for stereo matching.

4.2. Effect of incorporating MobileNet blocks

In this section, we incorporate v_1 and v_2 blocks (either 2D or 3D, depending on the type of convolutional module) in various components of the 2D and 3D baselines. In addition to error metric, we monitor the reduction of computational complexity to help us choose lighter models. We analyze replacing the fundamental modules of the network, *i.e.* feature extraction and hourglass with lighter blocks. The results of substituting these modules with v_1 and v_2 ($t = 2$) for 2D and 3D models are tabulated in Tables 3a and 3b. The best model is selected according to the least EPE obtained in 20 epochs. Also, the input resolution for computing MACs is 256×512. In these cases, the first convolutions of feature extraction are kept in standard type.

We can conclude that: i) In both 2D and 3D baselines, feature extraction is responsible for much of the computational load. ii) Substituting feature extraction with v_1 and hourglass with v_2 yields a better compromise between accuracy and computational complexity. For the 2D baseline, this combination results in the least EPE. We consider this combination for both 2D and 3D models.

We also examine replacing other modules with lighter blocks to make the network even lighter. Namely, the first convolutional layers in feature extraction and pre-hourglass

FE$_{2D}$	HG$_{2D}$	EPE(px)	MACs(G)	Params(M)
conv.	conv.	1.55	74.42	4.07
conv.	v_1	1.62	73.97	3.52
v_1	conv.	1.66	30.43	1.52
v_1	v_1	1.59	29.98	0.98
conv.	v_2	1.63	74.32	3.75
v_2	conv.	1.57	35.54	1.81
v_2	v_2	1.53	35.44	1.49
v_1	v_2	1.50	30.33	1.21
v_2	v_1	1.60	35.10	1.26

(a) 2D models: 3D cost volume using $Interlaced_4$ method

FE$_{2D}$	HG$_{3D}$	EPE(px)	MACs(G)	Params(M)
conv.	conv.	0.97	155.2	4.21
conv.	v_1	0.99	143.66	3.42
v_1	conv.	0.98	111.2	1.66
v_1	v_1	1.03	99.67	0.87
conv.	v_2	0.97	149.01	3.53
v_2	conv.	0.96	116.32	1.94
v_2	v_2	0.97	110.13	1.27
v_1	v_2	0.99	105.01	0.98
v_2	v_1	1.02	104.78	1.15

(b) 3D models: 4D cost volume using Gwc40 method [6]

Table 3: Performance evaluation on SceneFlow test set for variants of (a) 2D and (b) 3D baselines with $1 \times HG$.

"FE" and "HG" stand for feature extraction and hourglass.

modules are replaced with $v2$. The reason for choosing $v2$ instead of $v1$ is the higher accuracy $v2$ can maintain after substituting for standard convolutions. We observed that in this case, for the 2D baseline, both the complexity and the error are reduced (tables available in the appendix). We also tried replacing other modules with MobileNet blocks, *i.e.* the channel reduction module and the convolutions in cost volume construction in the 2D baseline. However, since these replacements deteriorate the learning capability of the model, they are kept unchanged with standard convolutions.

4.3. Quantitative and qualitative results

The discussed experiments support our design choice for the final 2D and 3D models, *i.e.* 2D-MobileStereoNet and 3D-MobileStereoNet. To increase the accuracy, we utilize a stack of three hourglasses. Also, we found out that higher t values make the network less accurate and heavier.

SceneFlow dataset. The evaluation of the proposed models on SceneFlow are presented in Tables 4a and 4b. Note that 2D-MobileStereoNet has more parameters than 3D-MobileStereoNet due to i) channel reduction module, ii) parameterized cost volume construction, and iii) wider hourglass. Nevertheless, with the least operations, it can be practical on systems with limited computation capacities. Compared to other 2D models, a lower error is achieved by 2D-MobileStereoNet with 16.6x fewer parameters. Moreover, we observe that in 3D-MobileStereoNet, significantly fewer parameters are achieved, while the performance is competitive with or better than other methods. Compared to GwcNet-gc with the best EPE metric in the table, 3D-MobileStereoNet uses 1.7x fewer parameters (in millions) and 3.9x fewer GigaMACs. Note that although DeepPruner-Fast [4] obtains the least number of operations,

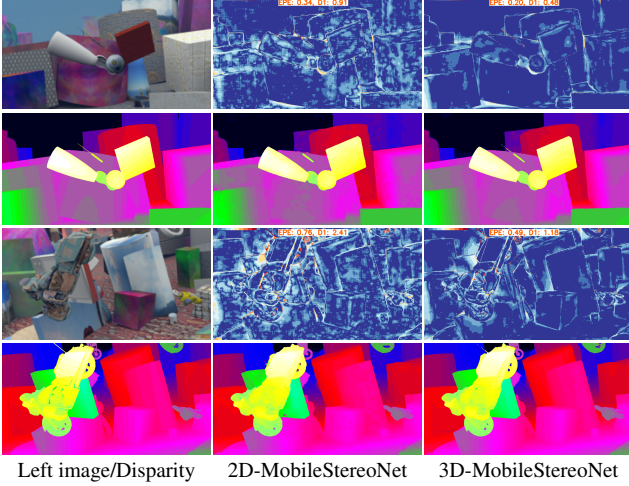Left image/Disparity | 2D-MobileStereoNet | 3D-MobileStereoNet

Figure 6: Qualitative performance on SceneFlow: Every two rows correspond to a test sample. In the left-most column, the samples and the ground-truth disparity maps are illustrated. The following two columns show the disparity and error maps (embedded with error values) estimated by 2D-MobileStereoNet and 3D-MobileStereoNet.

Method	EPE(px)	Params(M)	Red. Params
DispNet-C[9]	1.67	38	17.0x
CRL[16]	1.60	78.77	35.3x
AutoDispNet-C[22]	1.51	37	16.6x
iResNet[11]	1.40	43.11	19.3x
2D-MobileStereoNet	**1.14**	**2.23**	-

(a) 2D models

Method	EPE(px)	MACs(G)	Params(M)	Red. MACs	Red. Params
GCNet[10]	1.84	718.01	3.18	4.7x	1.8x
PSMNet[2]	0.88	256.66	5.22	1.7x	2.9x
GA-Net-deep[32]	0.84	670.25	6.58	4.4x	3.7x
GA-Net-11[32]	0.93	383.42	4.48	2.5x	2.5x
Gwc40-Cat24-Base[6]	1.12	169.42	4.60	1.1x	2.6x
GwcNet-gc[6]	**0.76**	260.49	6.82	1.7x	3.9x
GwcNet-g[6]	0.79	246.27	6.43	1.6x	3.6x
DeepPruner-Best[4]	0.86	129.23	7.39	0.8x	4.2x
DeepPruner-Fast[4]	0.97	**51.83**	7.47	0.3x	4.2x
3D-MobileStereoNet	0.80	153.14	**1.77**	-	-

(b) 3D models

Table 4: Comparison on SceneFlow test set for (a) 2D and (b) 3D models. "Red." indicates the reduction factor of our models compared to other methods.

it is still over-parametrized and this causes issues when finetuning on smaller datasets like KITTI (*cf.* Table 6). Figure 6 shows the disparity estimation results.

KITTI 2015 dataset. This dataset consists of images of real-world driving scenarios [14], with 376×1236 resolution. To evaluate our models on this dataset, which has only ground-truth available for 200 for training samples, we use a 159/40 training/validation split. We finetune the networks that are pretrained on SceneFlow. For a fair comparison, we also train and evaluate other methods with the same data split. As shown in Tab. 5, 2D-MobileStereoNet attains comparable results to PSMNet, which is a 3D model,

Method	EPE(px)	D1(%)	px-3(%)	MACs(G)	Params(M)
PSMNet[2]	0.88	2.00	2.10	256.66	5.22
GA-Net-deep[32]	0.63	1.61	1.67	670.25	6.58
GA-Net-11[32]	0.67	1.92	2.01	383.42	4.48
GwcNet-gc[6]	0.63	1.55	1.60	260.49	6.82
GwcNet-g[6]	**0.62**	**1.49**	**1.53**	246.27	6.43
2D-MobileStereoNet	0.79	2.53	2.67	**32.2**	2.32
3D-MobileStereoNet	0.66	1.59	1.69	153.14	**1.77**

Table 5: Comparison on KITTI 2015 validation set.

Methods	All(%)			Noc(%)		
	$D1_{bg}$	$D1_{fg}$	$D1_{all}$	$D1_{bg}$	$D1_{fg}$	$D1_{all}$
MC-CNN[31]	2.89	8.88	3.89	2.48	7.64	3.33
Fast DS-CS[29]	2.83	4.31	3.08	2.53	3.74	2.73
GCNet[10]	2.21	6.16	2.87	2.02	5.58	2.61
DeepPruner-Fast[4]	2.32	3.91	2.59	2.13	3.43	2.35
PSMNet[2]	1.86	4.62	2.32	1.71	4.31	2.14
AutoDispNet-CSS[22]	1.94	**3.37**	2.18	1.80	**2.98**	2.00
DeepPruner-Best[4]	1.87	3.56	2.15	1.71	3.18	1.95
GwcNet-g[6]	**1.74**	3.93	2.11	**1.61**	3.49	**1.92**
2D-MobileStereoNet	2.49	4.53	2.83	2.29	3.81	2.54
3D-MobileStereoNet	1.75	3.87	**2.10**	**1.61**	3.50	**1.92**

Table 6: Comparison on KITTI 2015 benchmark. 3D-MobileStereoNet requires 98% and 72% fewer parameters compared to AutoDispNet-CSS and GwcNet-g, respectively.

with much less computational load (2.3x/8x fewer parameters/operations). Also, 3D-MobileStereoNet is outperforming PSMNet and GA-Net-11, and is competitive with GA-Net-deep. Compared to GwcNet-g, 3D-MobileStereoNet is lighter with 3.6x/1.6x fewer parameters/operations.

Additionally, we submitted the results of our finetuned models to the KITTI 2015 benchmark. To this end, we finetuned the epoch with the best cross-domain generalizability from SceneFlow to KITTI 2015. Table 6 shows that 2D-MobileStereoNet is surpassing GCNet (a 3D model) with 27%/95% fewer parameters/operations. Also, we can verify that 3D-MobileStereoNet shows superior performance when compared to GCNet and PSMNet and it is surpassing GwcNet-g (in $D1_{fg}$ and in $D1_{all}$ in all pixels) with 72%/38% fewer parameters/operations.

Figure 7 visualizes the results from KITTI 2015 benchmark. Comparing 2D-MobileStereoNet and 3D-MobileStereoNet, we observe that 3D-MobileStereoNet obtains crisp edges due to deploying 3D convolutions in the encoder-decoder. In other words, in the upsampling/downsampling process in encoder-decoder, 3D convolutions can better preserve finer details compared to an encoder-decoder with 2D convolutions. Nevertheless, 2D-MobileStereoNet achieves visually similar outputs to 3D models while requiring considerably fewer operations (87% fewer operations compared to GwcNet-g). Also, 3D-MobileStereoNet visually achieves competitive or better results compared to other methods.

Model size. We also report the memory requirement of our models in Table 7. Both of the proposed methods show smaller memory sizes, which is promising for memory-constrained chips and their power consumption.
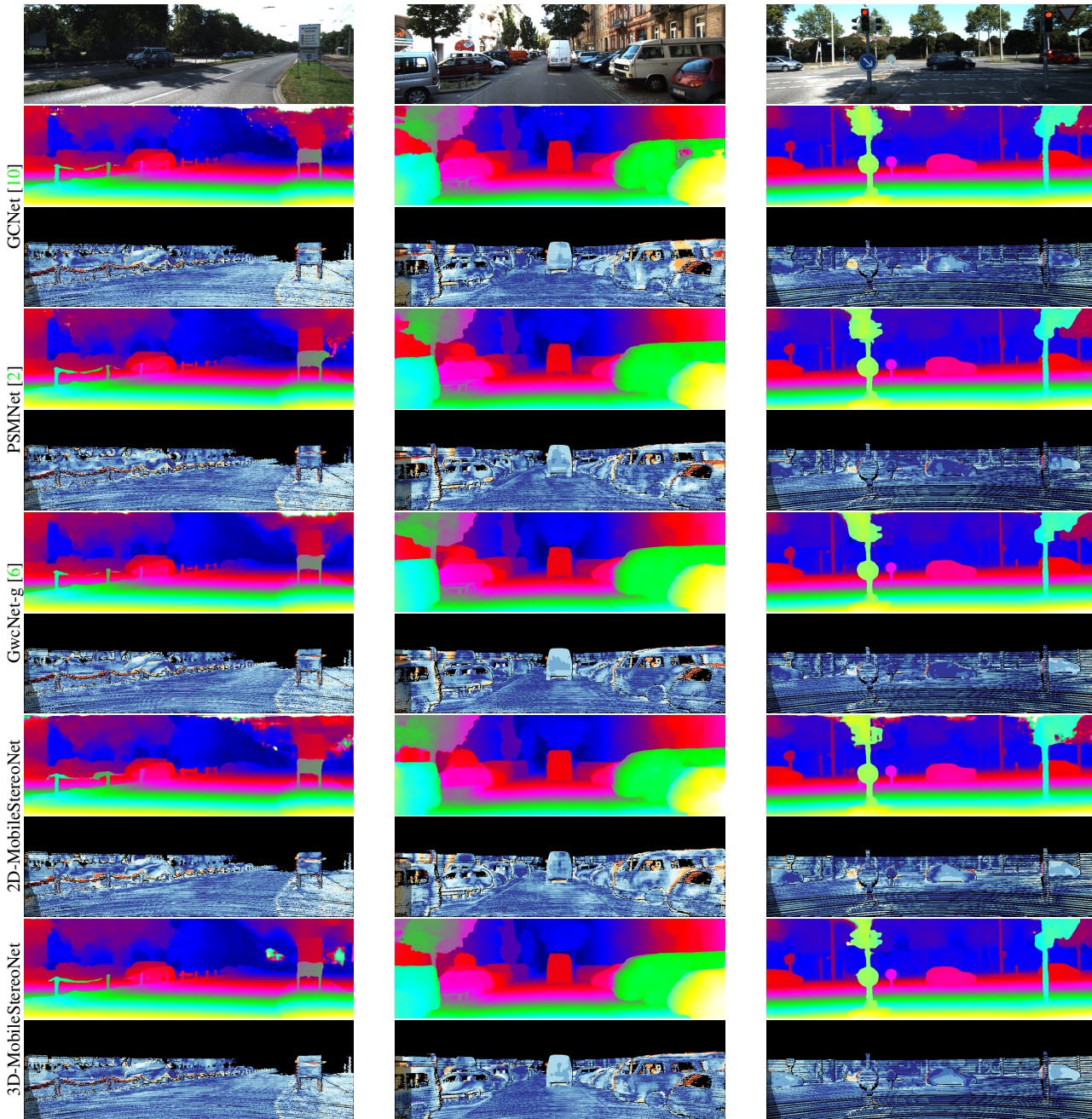
Figure 7: Qualitative performance (disparity images together with error maps) from KITTI 2015 benchmark. Warmer colors in error maps denote larger values.

Method	GA-Net-deep[32]	GA-Net-11[32]	GwcNet-gc[6]	GwcNet-g[6]	PSMNet[2]	2D-MobileStereoNet	3D-MobileStereoNet
Memory (MB)	26.4	18.0	27.9	26.3	21.1	10.03	7.99

Table 7: Comparison of model size. Our two proposed methods yield more compact models compared to other works.

5. Conclusion

This paper presented lightweight stereo networks to alleviate high memory usage on embedded or mobile devices. Namely, we proposed two models (2D and 3D) with the primary goal of reducing the cost (in terms of parameters, operations, and model size) by using MobileNet blocks. To increase the accuracy of the 2D model, we also designed a new cost volume to *learn* the similarity of unary features. Yielding a favorable accuracy/complexity trade-off, these MobileStereoNets are promising for deploying end-to-end stereo networks on edge devices.

References

[1] Konstantinos Batsos, Changjiang Cai, and Philippos Mordo-hai. CBMV: A coalesced bidirectional matching volume for disparity estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2060–2069, 2018.

[2] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018.

[3] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[4] Shivam Duggal, Shenlong Wang, Wei-Chiu Ma, Rui Hu, and Raquel Urtasun. DeepPruner: Learning efficient stereo matching via differentiable patchmatch. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4384–4393, 2019.

[5] Christoph Feichtenhofer. X3D: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 203–213, 2020.

[6] Xiaoyang Guo, Kai Yang, Wukui Yang, Xiaogang Wang, and Hongsheng Li. Group-wise correlation stereo network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3273–3282, 2019.

[7] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 807–814, 2005.

[8] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[9] Eddy Ilg, Tonmoy Saikia, Margret Keuper, and Thomas Brox. Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In *Proceedings of the European Conference on Computer Vision*, pages 614–630, 2018.

[10] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 66–75, 2017.

[11] Zhengfa Liang, Yiliu Feng, Yulan Guo, Hengzhu Liu, Wei Chen, Linbo Qiao, Li Zhou, and Jianfeng Zhang. Learning for disparity estimation through feature constancy. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2811–2820, 2018.

[12] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5695–5703, 2016.

[13] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. 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*, pages 4040–4048, 2016.

[14] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3061–3070, 2015.

[15] Anwesan Pal, Sayan Mondal, and Henrik I Christensen. "looking at the right stuff"-guided semantic-gaze for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11883–11892, 2020.

[16] Jiahao Pang, Wenxiu Sun, Jimmy SJ Ren, Chengxi Yang, and Qiong Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 887–895, 2017.

[17] Haesol Park and Kyoung Mu Lee. Look wider to match image patches with convolutional neural networks. *IEEE Signal Processing Letters*, 24(12):1788–1792, 2016.

[18] Min-Gyu Park and Kuk-Jin Yoon. Leveraging stereo matching with learning-based confidence measures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 101–109, 2015.

[19] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5533–5541, 2017.

[20] Rafia Rahim, Faranak Shamsafar, and Andreas Zell. Separable convolutions for optimizing 3d stereo networks. In *Proceedings of the IEEE International Conference on Image Processing*, 2021.

[21] Zhibo Rao, Mingyi He, Yuchao Dai, Zhidong Zhu, Bo Li, and Renjie He. NLCA-Net: A non-local context attention network for stereo matching. *APSIPA Transactions on Signal and Information Processing*, 9:e18, 2020.

[22] Tonmoy Saikia, Yassine Marrakchi, Arber Zela, Frank Hutter, and Thomas Brox. AutoDispNet: Improving disparity estimation with automl. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1812–1823, 2019.

[23] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[24] Akihito Seki and Marc Pollefeys. SGM-Nets: Semi-global matching with neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 231–240, 2017.

[25] Weijing Shi and Raj Rajkumar. Point-GNN: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1711–1719, 2020.

[26] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019.

[27] Zhenyao Wu, Xinyi Wu, Xiaoping Zhang, Song Wang, and Lili Ju. Semantic stereo matching with pyramid cost volumes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7484–7493, 2019.

[28] Rongtian Ye, Fangyu Liu, and Liqiang Zhang. 3d depthwise convolution: Reducing model parameters in 3d vision tasks. In *Canadian Conference on Artificial Intelligence*, pages 186–199, 2019.

[29] Kyle Yee and Ayan Chakrabarti. Fast deep stereo with 2d convolutional processing of cost signatures. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 183–191, 2020.

[30] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In *Proceedings of the European Conference on Computer Vision*, pages 151–158, 1994.

[31] Jure Zbontar, Yann LeCun, et al. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17(1):2287–2318, 2016.

[32] Feihu Zhang, Victor Prisacariu, Ruigang Yang, and Philip HS Torr. GA-Net: Guided aggregation net for end-to-end stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 185–194, 2019.