Dissertation

Optimizing 3D Convolutions on Stereo Matching for Resource Efficient Computation

Graduate School of
Natural Science & Techonology
Kanazawa University

Division of Electrical Engineering and Computer Science
:

Studen ID No. : **1824042012**
Name: **Jianqiang Xiao**
Chief advisor: **Satoshi Yamane**
Date of Submission: **January 07,2022**

"The one, simple, elegant equation to explain everything."

The Theroy of Everything

Abstract

Jianqiang Xiao

Optimizing 3D Convolutions on Stereo Matching for Resource Efficient Computation

Humans and most organisms are born with one set of eyes. Our pupillary distance is close, so the visual range of both eyes mostly overlap each other. Both eyes produce similar visual signals to the brain. After processing the signals, we can determine the distance from our eyes to an object. Therefore, we can perceive our position in 3D space.

For computers, we can also simulate biological vision by fixing the focus of two cameras on the same horizontal plane with a close distance. We refer to this as stereo matching. Most of the deep learning approaches extract semantics from each of the two images using 2D CNNs. Since the left and right views mostly overlap, a share-weight siamese network is used to extract semantic informations. Then the left and right feature maps aggregate with each other and a disparity dimension is added to the 3-dimensional feature map and from a 4-dimensional cost volume. However, when 3D CNNs are used to process cost volume, an extra dimension makes the computational cost rise exponentially. Therefore, binocular vision technology is difficult to apply in real-world scenarios that require real-time. Besides stereo matching, 3D CNNs are widely used in other stereo vision subjects, such as multiple view stereo, 3D object detection, visual SLAM (simultaneous localization and mapping), etc. To address this problem, we investigate 2D kernel-based methods, select suitable methods for building 3D CNN kernels, and experiment on the stereo matching algorithm. By using both exact methods

and efficient methods, our model significantly reduces the computational cost and maintains a comparable accuracy without changing the architecture. Our study provides a comprehensive discussion of 3D convolutions in stereo matching algorithms, which has guidance for other stereo vision subjects.

Acknowledgements

Firstly, I would like to thank my supervisor, Dr. Yamane Satoshi, for the support and assistance he has given me throughout my studies. Without his guidence and help, I would never able to complete my work.

Then, I would like to thank my friends, family and colleagues in the computer software laboratory for their support and companionship, giving me the courage to face all the difficulties.

At last, I want to say something not important. As a person who just embarked on the road of academic research by chance, my knowledge is not as good as other peers, but out of interest, I still chose the very competitive computer vision field. When I really started to understand research in the field, besides being amazed at the beauty of science and research, I also fell into self-doubt deeply. I knew I couldn't do as well as others, but it was difficult for me to accept the fact, couldn't start the 'right' work for a long time. Really thanks to my supervisor, family and friends for their support during that time and helping me out of the predicament. Along the way, I caught a glimpse of the beauty of science, met many wonderful peers love research, and were really excited about some marvellous researches. With the end of the journey, I will also strive to improve myself on another platform and make contributions to our social progress.

Contents

List of Abbreviations

AR	Augmented Reality
MR	Mixed Reality
CNNs	Convolutional Neural Networks
MAdd	Multiply-add operations
AD	Absolute Differences
SAD	Sum of Absolute Differences
NCC	Normalized Correlation Coefficient
DSI	Disparity Space Image
WTA	Winner-takes-all
IC	Intensity Consistent
LC	Locally Consistent
ADAS	Advanced Driver Assistance Systems
LIDAR	LIght Detection And Ranging
BB	Bounding Box
AGV	Autonomous Guided Vehicle
SLAM	Simultaneous Localization And Mapping
EPE	End-Point-Error
FLOPs	FLoating Point Operations
MAC	Memory Access Cost

Chapter 1

Introduction

About 543 million years ago, the vast majority of species predated by simply waiting for their prey to enter their range. Creatures with vision ability appeared on Earth, and the Cambrian explosion occurred (Parker, 2004). Since then, organisms with the ability to see quickly gained an advantage, and it made them to become more proactive in their predatory and survival behavior. Under the superiority of the natural environment, now, most organisms have vision ability, which means vision is the most important sense for a living creature.

Most organisms with vision have a pair of eyes, and environmental perception under binocular vision gives organisms greater advantage in the harsh nature environment. Binocular vision refers to the vision produced by organisms when the visual fields of the two eyes overlap with each other. Due to the interpupillary distance between the eyes, they produce a pair of slightly different visual signal on the retina. After this visual signal is transmitted to the brain, the brain integrates the two different images to determine the precise distance from the eye to the object (*Visual Fields: Examination and Interpretation* 2010).

From a technical perspective, computer vision has become one of the most active research fields since AlexNet (Krizhevsky, Sutskever, and Hinton, 2012a) made historical progress with absolute advantages in ILSVRC (ImageNet Large Scale Visual Recognition Challenge) 2014 (Russakovsky et al., 2014). Computer vision has made significant progress in semantic-related tasks, such as classification (He et al., 2015a; Szegedy et al., 2015), object detection (Ren et al., 2015; Lin et al., 2017a) and semantic segmentation (Ronneberger, P.Fischer, and Brox, 2015; He et

al., 2017). Recent studies have shown that in the field of monocular vision, computer has achieved similar performance as human. However, in stereo vision, the computer has encountered some serious problems. The main reason is that in computer vision, when the input is two pictures, the cost of calculation is not increase twice, but increases exponentially. This makes it necessary for researchers to make trade-offs between accuracy and real-time performance, which makes it difficult to apply stereo vision research in the real world. Based on the problem of computational cost and accuracy, this research proposes a convolution-based method, which greatly reduces the computational cost without reducing the accuracy.

1.1 Motivation

Stereo matching plays an important role in 3D computer vision applications, such as augmented reality (AR) (Zenati and Zerhouni, 2007), mixed reality (MR) (Noh, Sunar, and Pan, 2009), autonomous vehicle (Hane, Sattler, and Pollefeys, 2015) and robot navigation (Nalpantidis, Sirakoulis, and Gasteratos, 2007; Samadi and Othman, 2013). It provides accurate disparity by a pair of stereo images. We can calculate the depth value by $D = fB/d$, where d denotes the disparity of the pixel, f is the focal length of the camera and B is the distance between the camera centers (Zbontar and LeCun, 2015a). To get a precise disparity map is one of the most important tasks in stereo vision.

Classic stereo matching algorithms contain four parts: matching cost computation, cost support aggregation, disparity computation and disparity optimization (Scharstein and Szeliski, 2002). Early studies perform machine learning methods to optimize disparity by Markov random field (Zhang and Seitz, 2007), conditional random field (Scharstein and Pal, 2007) or random forest **ref-31**. With the rise of convolutional neural networks (CNNs), CNN-based approaches have been developed progressively. MC-CNN (Haeusler, Nair, and Kondermann, 2013) first investigates CNNs on matching corresponding points for disparity estimation. Geometry and Context Network (GC-Net) (Kendall et al., 2017) makes the training process end-to-end with a differentiable ArgMin operation on disparity estimations.

The Pyramid Stereo Matching Network (PSMNet) (Chang and Chen, 2018) introduces spatial pyramid pooling and a stacked hourglass module for an accurate disparity map. These famous studies form a CNN-based approach framework: 2D Siamese feature extraction, cost volume aggregation, cost volume regularization and disparity regression.

One major problem with current CNN-based stereo matching algorithms is the enormous computation for cost volume regularization. The cost volume aggregation stage builds correspondence between left and right feature maps, aggregating *disparity* as an additional dimension on left feature maps to form 4D cost volumes (Kendall et al., 2017). For the cost volume regularization stage, most CNN-based methods build 3D convolution layers and 3D deconvolution layers, composing a 3D encoder–decoder architecture for regularizing 4D cost volumes. Compared to 2D convolution kernels, the additional dimension forming 3D convolution kernels raises computation complexity exponentially, leading the cost volume regularization to contain most of the computation among the entire architecture. Therefore, we build the 3D version of resource efficient kernel-based methods which can match up with the logic of a 3D encoder–decoder. By classifying all 3D layers in PSMNet according to their functions, we replace the original layers with our lightweight 3D convolution layers and implement a series of comparative experiments. Eventually, compared to the original PSMNet, we save 34.3% parameters and 73.1% multiply-add operations (MAdd) (95.50% paramters and 94.53% MAdd for 3D CNNs) without losing performance. We evaluate our model on Scene Flow and KITTI 2015 public datasets and obtain competitive results with other state-of-the-art stereo matching algorithms.

1.2 Contribution

The contributions of this thesis can be summarized as follows:

- **We compile a full discussion on optimizing 3D convolutional kernels on stereo matching algorithms.** These days, there are several well-known 2D kernel-based methods, some of which are used to reduce the computational

cost of the network, making it lightweight and able to use in mobile devices. Some other kernel-based methods make the network perform significantly better with only a small additional computational cost. In our study, we divided the existing kernel-based methods into lightweight kernels and accurate kernels, and tested their performance on the stereo matching network separately.

- **We explore a network design guideline for optimizing 3D convolution kernels on stereo matching algorithms and achieve an accurate and lightweight stereo matching algorithm.** By classifying the 3D convolution layers in the baseline, for different classes of convolutions, we use the corresponding 3D convolutions for our experiments. Through a series of experiments based on the kernel approach, we obtain the effect of 3D convolution on the stereo matching algorithm at different layers. And since 3D convolution is also widely used for other 3D-related tasks, our study is instructive for some other stereo tasks as well.
- **By following the above guidelines, our model exhibits comparable results and much lower computational cost without changing the network architecture.**

1.3 Thesis Outline

The following thesis is organized as:

In Chapter 2, we present a comprehensive survey of recent research on stereo matching and kernel-based methods.

In Chapter 3, we explained some important concepts and notations. These include our application scenario, training dataset and computational complexity matrics.

In Chapter 4, we explained the structure of the network, including PSMNet and the designed 3D convolutions.

In Chapter 5, we described the entire network design pipeline on optimizing 3D convolutions for resource efficient computation in detail.

In Chapter 6, we compared the resulting network model with state-of-the-art.

In Chapter 7, we discussed the problems in the research and their reasons.

In Chapter 8, we draw a summary of the entire thesis.

Chapter 2

Survey of Related Studies

In this chapter, we present the main r covered in this paper. Through the research on the related literature of stereo matching, we understand the problems faced by this research field, which leads to the solution of the kernel-based methods as one of the solution.

2.1 Stereo Matching

Stereo matching is also known as disparity estimation, or binocular depth estimation. Under stereo matching framework, we use a binocular camera with calibrated rectification as the sensor and get a pair of stereo images. In chapter 1, we mentioned that the stereo matching algorithm can estimate depth information. Now we briefly introduce how to get depth information through the triangulation geometry. **Note:** we only discuss calibrated parallel cameras in the thesis.

2.1.1 Triangulation geometry

In a standard model (calibrated camera) as shown in figure 2.1, O_L and O_R are focuses of the left and right cameras. We have an object P can be shown in left frame in P_L and right frame in P_R. If we need to get the depth D from the point P to the stereo camera, we need to use similar triangle geometry. In the figure we can simply recognize $\triangle_{PO_LO_R} \sim \triangle_{PP_LP_R}$. And we can refer line "P_LP_R" as:

$$P_LP_R = b - (X_L - \frac{L}{2}) - (\frac{L}{2} - X_R) = b - (X_L - X_R). \tag{2.1}$$

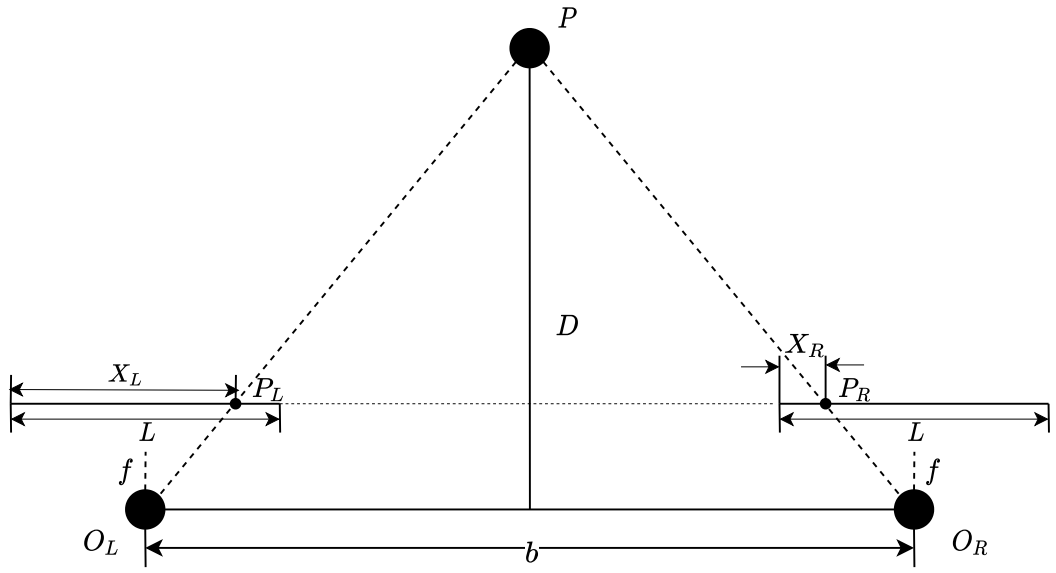


FIGURE 2.1: Triangulation geometry in stereo vision.

As $\triangle PO_LO_R \sim \triangle PP_LP_R$, we have:

$$\frac{b-(X_L-X_R)}{D-f}=\frac{b}{z}. \tag{2.2}$$

So, the depth 'D', can be calculated as:

$$D=\frac{b\times f}{X_L-X_R}. \tag{2.3}$$

And $d=|X_L-X_R|$ is disparity of this pixel. b (the baseline) and f (the focal length) are the intrinsic parameters of the stereo camera. So in the equation 2.3, if we get the disparity P_LP_R, we can get the exactly depth D. And how to get disparity is the theme of stereo matching.

2.1.2 Classic Stereo matching algorithms

In taxonomy (Scharstein and Szeliski, 2002), we know that stereo matching mainly have four stages: matching cost computation, cost support aggregation, disparity computation and disparity refinement.

Matching Cost Computation

The purpose of the matching cost computation is to measure the correlation between the pixel to be matched and the candidate pixel. Whether two pixels are points with the same name or not, the matching cost can be calculated through the matching cost function. The smaller the cost, the greater the correlation and the greater the probability of being a same pixel.

When we search for a pixel corresponding to the left and right image, we set a range $D(D_{min} - D_{max})$ as the maximum disparity searching range. Then we create a $W \times H \times D$ (W is the width of input images, and H is the height of input images) three-dimensional matrix C to store the matching cost value of each pixel within the disparity search range. We usually call matrix C as Disparity Space Image (DSI) (Intille and Bobick, 1994).

There are many ways to calculate the matching cost. For traditional methods, we use Absolute Differences (AD) (Yang, Yuille, and Lu, 1993), Sum of Absolute Differences (SAD) (Hamzah, Rahim, and Noh, 2010), Normalized Correlation Coefficient (NCC) (Zhang et al., 2009) and so on to calculate the matching cost of the DSI matrix.

Cost Support Aggregation

In the previous step, we calculated the matching cost of the corresponding pixel in a certain size range. As for cost support aggregation, the purpose is to enable the matching value to reflect the correlation between the left and right pixels accurately. When matching within a local area, it is easily affected by visual noise. And when the area is a weak texture area or a repeated texture area, the cost matching value may not accurately reflect the correlation between pixels.

The cost aggregation is to introduce the relationship between adjacent pixels (e.g., there should be continuous inspection values between adjacent pixels) to optimize the cost matrix DSI $C(W, H, D)$. This optimization introduces the global information of the image, and the disparity cost of each pixel will be recalculated based

on the disparity value of its neighboring pixels. Therefore, we get a new DSI as $C'(W, H, D')$.

We can think of cost aggregation as a kind of disparity propagation. In the high textural areas, the cost computation is accurate. Then spread from the high textural areas to the low textural areas through cost aggregation, so we can get a relatively realistic global disparity map. Commonly cost aggregation methods include scan-line method (Mei et al., 2011), dynamic programming method (Leung, Appleton, and Sun, 2008), path aggregation method in SGM (Semi-Global Matching) algorithm (Hirschmuller, 2008), etc.

Disparity Computation

In the disparity computation step, WTA (Winner-Takes-ALL) algorithm is widely used for calculating the disparities of every pixel of the input image (Gong et al., 2007). For each pixel, the WTA algorithm selects the smallest cost value of all disparities as the result of the disparity.

Disparity Refinement

Disparity refinement is to further optimize the preliminary disparity map we got in the previous step, which includes steps such as removing noise, smoothing texture, and optimizing pixel accuracy. Generally, the left-right consistency check algorithm is used to eliminate errors caused by occlusion and noise (Hosni et al., 2009). Smoothing algorithms also are used to smooth the disparity map such as median filter (Ma et al., 2013) and bilateral filter (Richardt et al., 2010). There are also some methods that can effectively improve the quality of the disparity map, such as: robustness plane fitting (Hong and Chen, 2004), IC (Intensity Consistent) (Hirschmuller and Scharstein, 2008), LC (Locally Consistent) (Hirschmuller and Scharstein, 2008), etc.

2.1.3 CNNs-based Stereo matching algorithms

Nowadays, due to the development of artificial intelligence and convolutional neural networks, researchers turn their focus on CNNs-based methods. Therefore,

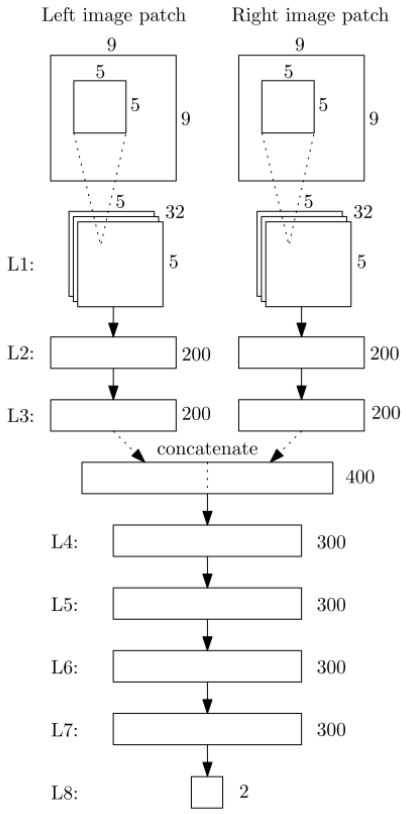


FIGURE 2.2: Network architecture of MC-CNN.

in this section, we introduce some main CNNs-based stereo matching algorithms.

MC-CNN (2015)

In 2015, (Zbontar and LeCun, 2015b) proposed the first stereo matching approach based on CNNs (convolutional neural networks). This paper replaces the matching cost computation step in the traditional stereo matching algorithm by introducing CNNs and set up the architecture of using siamese network to extract information from stereo images. The network architecture is shown in figure 2.2.

GC-Net (2017)

In 2017, GC-Net (Geometry and Context Network) (Kendall et al., 2017) successfully makes the stereo matching training process end-to-end. The entire network complies with a clean and concise structure of 2D CNNs - Cost Aggregation - 3D Encoder Decoder as shown in figure 2.3. This paper introduces a 4-dimensional

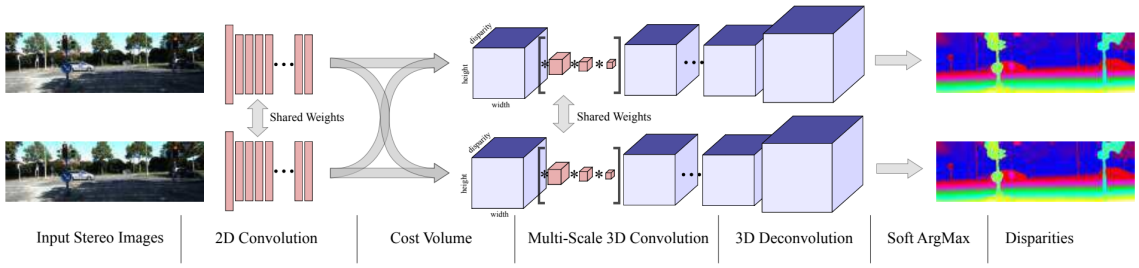


FIGURE 2.3: Network architecture of GC-Net.

cost volume to express the disparity cost in stereo matching, which is a more suitable form for CNNs. Cost volume adds the dimension of disparity cost to the feature map corresponding to each other from left and right to form a 4D cost volume. It adds stereo geometry information to the semantic and textural informations learned by 2D CNNs, which also conforms to the principle of traditional stereo matching.

Because when we use the traditional Argmin loss function on calculating the loss of cost volume, there are two obvious problems. a) The output is discrete and cannot produce sub-pixel disparity estimation. b) It is not differentiable, so it cannot be trained end-to-end when using backpropagation. They produced a soft Argmin that is fully differentiable during training and to calculate a smooth disparity estimation. First, convert the predicted costs c_d from a numerical value to a probability by taking the negative number of the cost value d. Then the entire disparity dimension is regularized by softmax operation $\sigma(.)$. Finally, they sum up all disparity d with the weighted probability. The mathematical expression of Soft Argmin is shown in equation 2.4:

$$Soft\ Argmin := \sum_{d=0}^{D_{max}} \times \sigma(-c_d) \tag{2.4}$$

The loss function is shown in equation4. The model uses absolute error to calculate the loss between predicted disparity d_n and ground truth disparity $\hat{d_n}$ at pixel n.

$$Loss = \frac{1}{N} \sum_{n=1}^{N} \|d_n - \hat{d_n}\|_1 \tag{2.5}$$

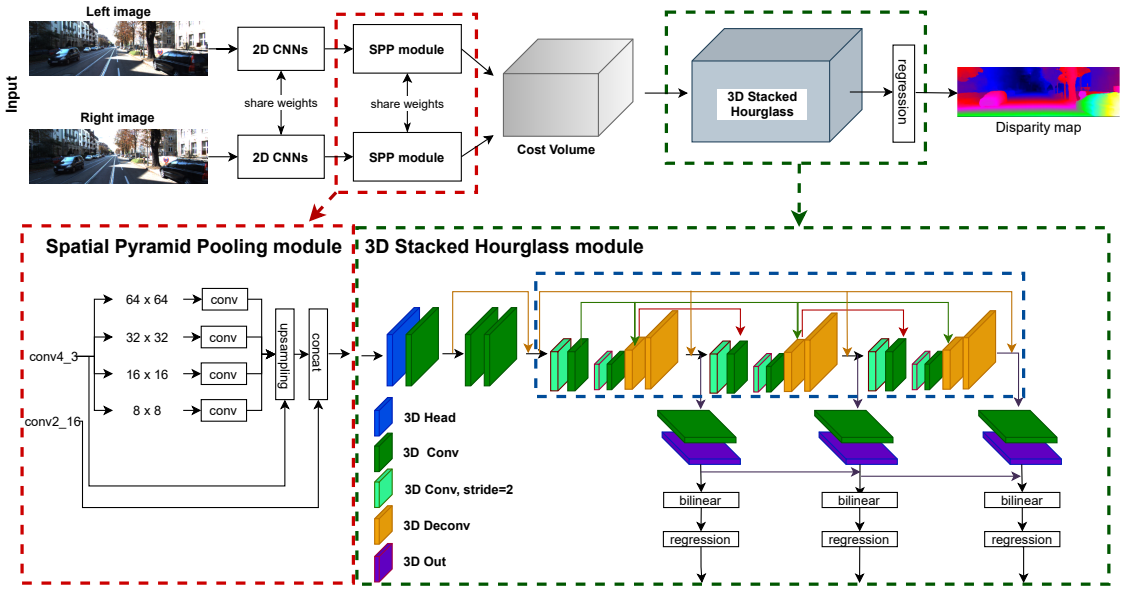


FIGURE 2.4: Network architecture of PSMNet.

PSMNet (2018)

Based on the GC-Net, PSMNet (Pyramid Stereo Matching Network) (Chang and Chen, 2018) optimized the network modules and achieved much better performance. In figure 2.4, PSMNet also follows a 2D CNNs - Cost Aggregation - 3D Encoder Decoder architecture. And PSMNet adds the mechanism of modern computer vision on the GC-Net framework. During 2D CNNs extracting feature maps from the input, the SPP (Spatial Pyramid Pooing) (He et al., 2015b; Zhao et al., 2017) module is added. By introducing the hierarchical features on the last fully-connected layer, they output more accurate semantic and textural feature maps. Also, they also upgrade the 3D encoder-decoder architecture with stacked hourglass (Newell, Yang, and Deng, 2016), which is widely used in human pose estimation and semantic segmentation. It consists of repeated top-down/bottom-up processing combined with intermediate supervision.

StereoNet (2018)

At the same time, part of the research is devoted to improving the real-time performance of the stereo matching algorithm. StereoNet (Khamis et al., 2018) is one of the most representative studies. Compared with an accurate method like PSMNet,

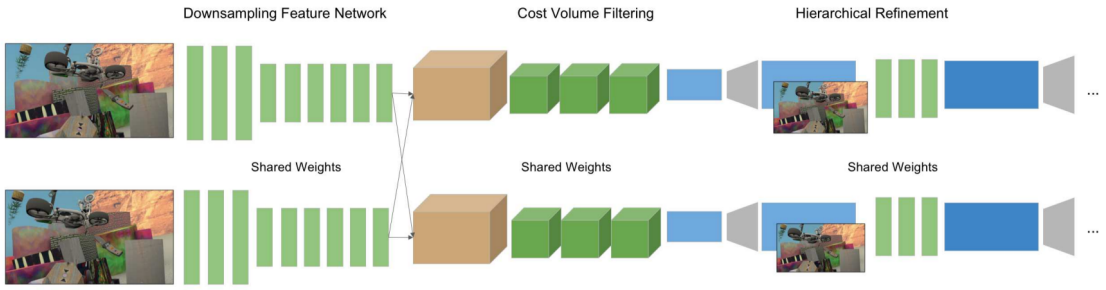


FIGURE 2.5: Network architecture of StereoNet.

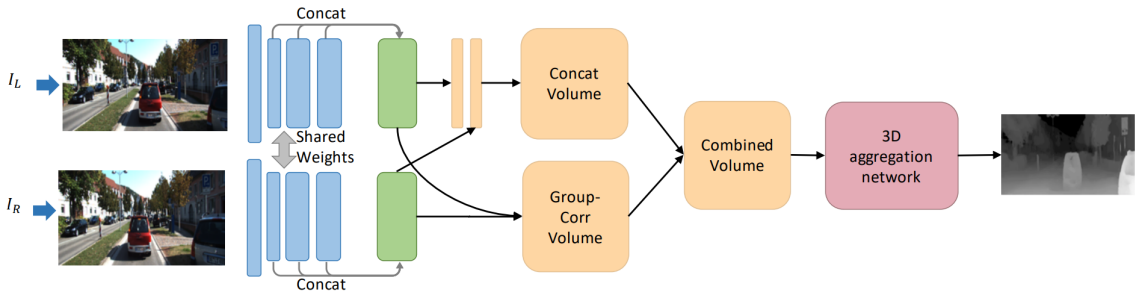


FIGURE 2.6: Network architecture of GwcNet.

StereoNet uses more down-sampling to extract a smaller feature map. Smaller feature maps aggregate a lower-resolution cost volume, which ultimately reduces the amount of calculation for the entire network.

Gwc-Net (2018)

Gwc-net (Group-wise Correlation Network) (Guo et al., 2019) improved the cost aggregation step and proposed group-wise correlation volume to calculate the originally combined cost volume by grouping, thereby it reduce the calculational cost. In figure 2.6, we can find that compared with PSMNet, the main contribution of Gwc-net is how to improve cost aggregation. The rest of steps are basically the same as PSMNet.

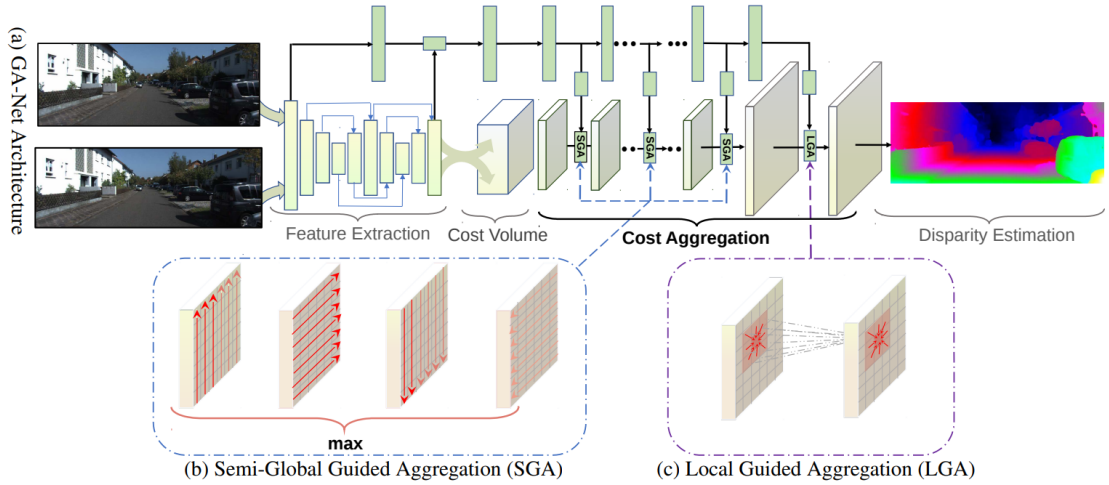


FIGURE 2.7: Network architecture of Ga-net.

Ga-net (2019)

Compared with other CNNs-based stereo matching algorithms, GA-net (Guided Aggregation Net) (Zhang et al., 2019) uses two aggregation methods, SGA (Semi-Global Matching)and LGA (Local Guided Aggregation) to replace 3D convolutions. In traditional patch matching tasks, SGM and LGA are quite famous aggregation methods. In traditional patch matching tasks, SGM and LGA are very well-known aggregation methods. However, there are many difficulties when they are put into end-to-end NN (neural network) for training. The main contribution of Ga-net lies in the design of new aggregation steps so that SGM and LGA can be used in the stereo matching algorithm of end-to-end training. The network architecture is shown in figure 2.7.

DeepPruner (2019)

Deeppruner algorithm (**bibid**) is also a real-time algorithm. The difference between it and StrereoNet is mainly in the method of constructing cost volume. In StereoNet, they use more down-sampling to get a low-resolution cost volume. In DeepPruner, they use PatchMatch module (Barnes et al., 2009) to generate sparse differentiable cost volume. Figure 2.8 shows the architecure of DeepPruner algorithm.

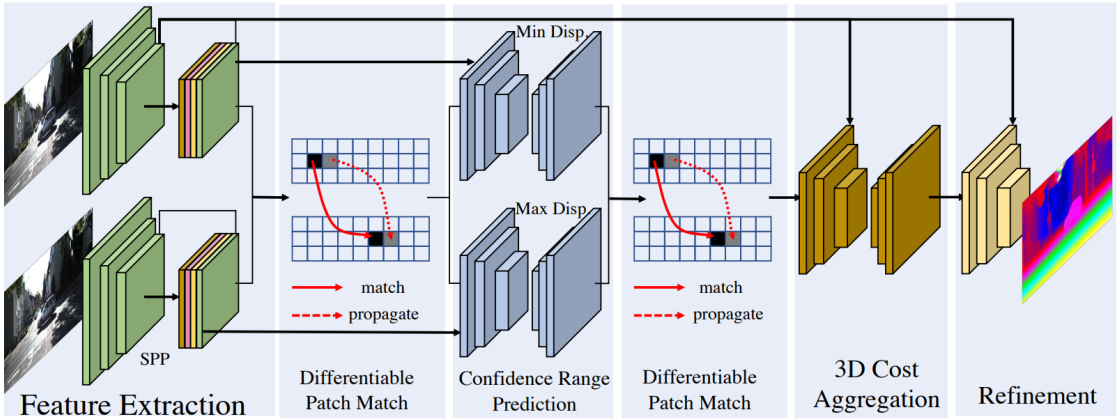


FIGURE 2.8: Network architecture of DeepPruner.

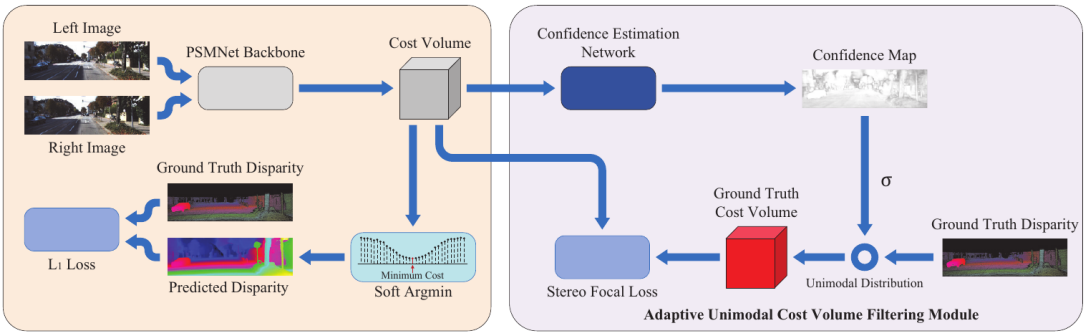


FIGURE 2.9: Network architecture of ACF-Net.

ACF-Net (2020)

Based on the general stereo matching pipeline, ACF-Net (Adaptive unimodal Cost volume Filtering Network) (Zhang et al., 2020) adds Ground Truth disparity and unimodal distribution, generating Ground Truth cost volume for intermediate supervised learning. The confidence estimation network (Fu and Fard, 2018) and stereo focal loss (Lin et al., 2017b) are also performed on the network to further improve the estimation accuracy.

AANet (2020)

In AAnet (Adaptive Aggregation Network) (Xu and Zhang, 2020), two different aggregation methods, ISA (Intra-Scale Aggregation) and SCA (Cross-Scale Aggregation), are proposed to replace 3D convolutions. In figure 2.10, AAnet aggregates multi-scale cost volumes, and forms an adaptive aggregation module through an

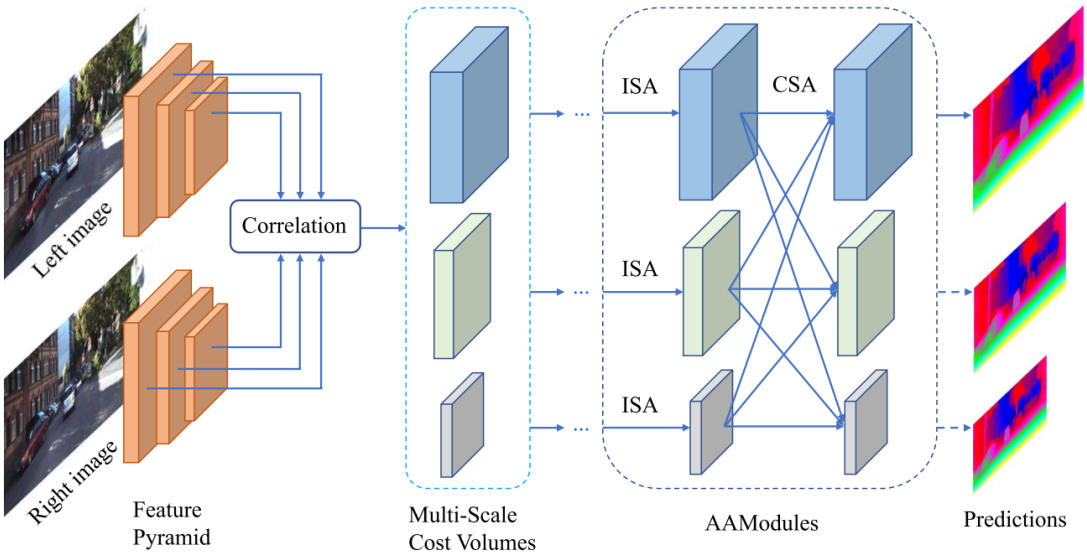


FIGURE 2.10: Network architecture of AANet.

ISA layer aggregated in three scales and a CSA layer aggregated in three cross-scales. Then stack multiple AA modules to complete cost aggregation and disparity computation, and finally output 3 low-resolution disparity maps. Finally, use the module of StereoDRNet (Chabra et al., 2019) to restore the resolution of the disparity map

CSN (2020)

As shown in figure 2.11, CSN (Cascade Stereo Network) (Gu et al., 2020) uses a coarse-to-fine model to build multi-level cost volumes. The cost volume that compares coarse is composed of larger scale semantic 2D features and hypothesis that compares sparse. Then we use the prediction of the previous stage to narrow the range of depth or parallax in the subsequent stage. Using this coarse-to-fine cost volume construction framework reduces the amount of network calculations.

According to previous references, most of the works focus on optimizing the network architecture, especially the cost volume aggregation part, which costs the greatest computational resources. Yet, 3D convolution layers and 3D transposed convolution layers, the basic elements of cost volume aggregation, have been rarely studied.

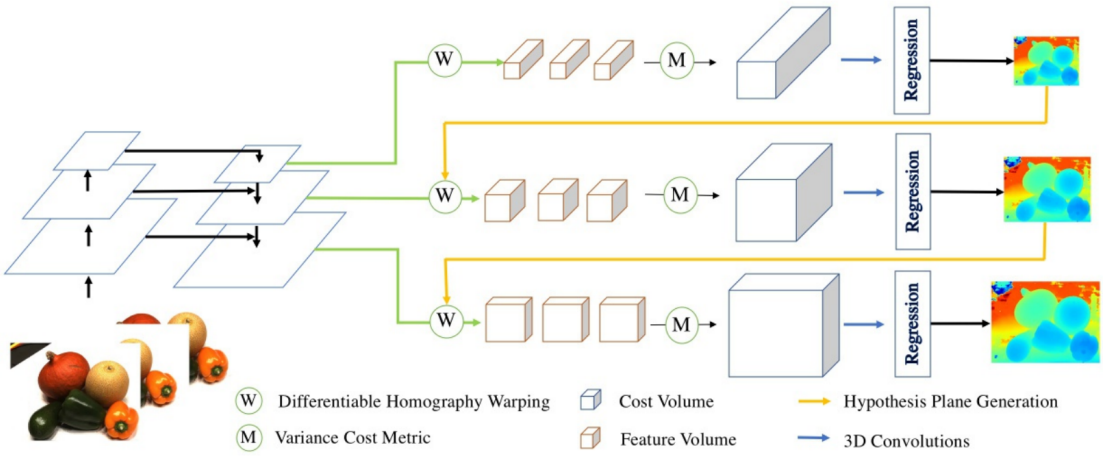


FIGURE 2.11: Network architecture of MVSNet + Cascade Cost Volume.

2.2 Kernel-based Methods

2.2.1 Lightweight Kernel-based Methods

Recent studies focus on building lightweight convolution kernels, which comprise the CNN-based application suitable for low resource devices, such as mobile phone and navigation robot. The origin SqueezeNet (Iandola et al., 2016) achieves AlexNet (Krizhevsky, Sutskever, and Hinton, 2012b) -level accuracy with 50 times fewer parameters. Xception (Chollet, 2017) and MobileNetV1 (Howard et al., 2017) implement depthwise separable convolutions for reducing the model parameters. MobileNetV2 (Sandler et al., 2018) proposes an inverted residual block with channel expansion for boosting the performance. ShuffleNetV1 (Zhang et al., 2018) presents grouped pointwise convolution and a channel shuffle operation to save computation. ShuffleNetV2 (Ma et al., 2018) further considers the relationship between hardware and network design, improving their performance in terms of speed and accuracy.

2.2.2 Channel-wise Attention Mechanism

The channel-wise attention mechanism has proven the potential for enhancing the performance with a small implementation. Squeeze and excitation networks (SENet) (Hu, Shen, and Sun, 2018) firstly present an effective attention mechanism

by aggregating a feature map with global average pooling along the channel and weights it on the respective channel. Selective kernel networks (SK-Net) (Li et al., 2019) improve the performance with optimizing the channel-wise information in two different sizes of receptive fields. Efficient channel attention networks (ECA-Net) (Wang et al., 2020) propose an effective channel attention module for saving computational burden.

Chapter 3

Important Concepts and Notations

In this chapter, we will introduce the application scenarios of stereo matching in the real world, the dataset used in training the algorithm and the matrics to evaluate the computational complexity of the network.

3.1 Application Scenario

3.1.1 Autonomous vehicle

The ADAS (Advanced Driver Assistance Systems) is a system that assists in driving and parking a car. When the system contains a human-computer interaction interface, it can increase vehicle safety and road safety. There are many autonomous driving companies that are researching related technologies, but they each have their own sensor-based research directions. For example, Mobileeye (Mobileye, 2021) uses a monocular camera plus LIDAR (LIght Detection And Ranging) as a solution. The monocular camera mainly relies on the Bounding Box (BB) that detects the target when measuring the distance, but in the scene where there is no object detected, it is difficult for the monocular system to get distance information. And the binocular system can solve this problem. It estimates the depth through disparity and solves the problem of the monocular self-driving system that depends on the detected bounding boxes. Subrau eyesight (Subaru, 2021) is a representative ADAS based on a binocular vision system. As shown in figure 3.1, Baidu's Apollo L4 ADAS (Apollo, 2021) also uses a binocular system. Tesla (Autopilot, 2021), based on a pure vision solution, uses 8 cameras around the body

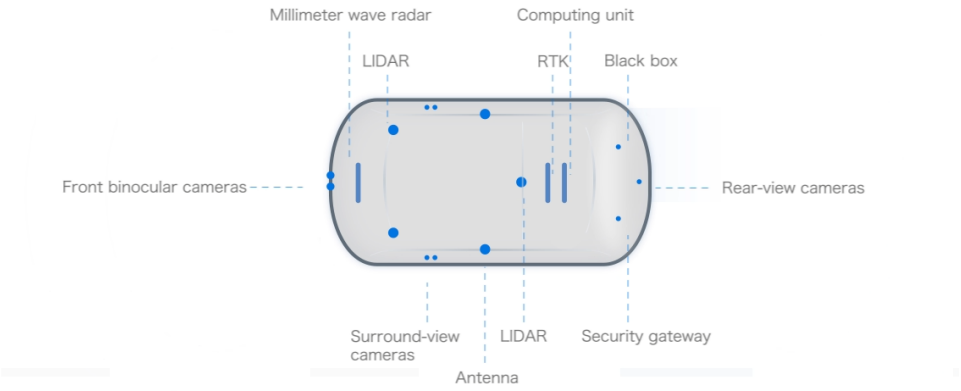


FIGURE 3.1: Sensor placement scheme of Baidu Apollo L4 ADAS.

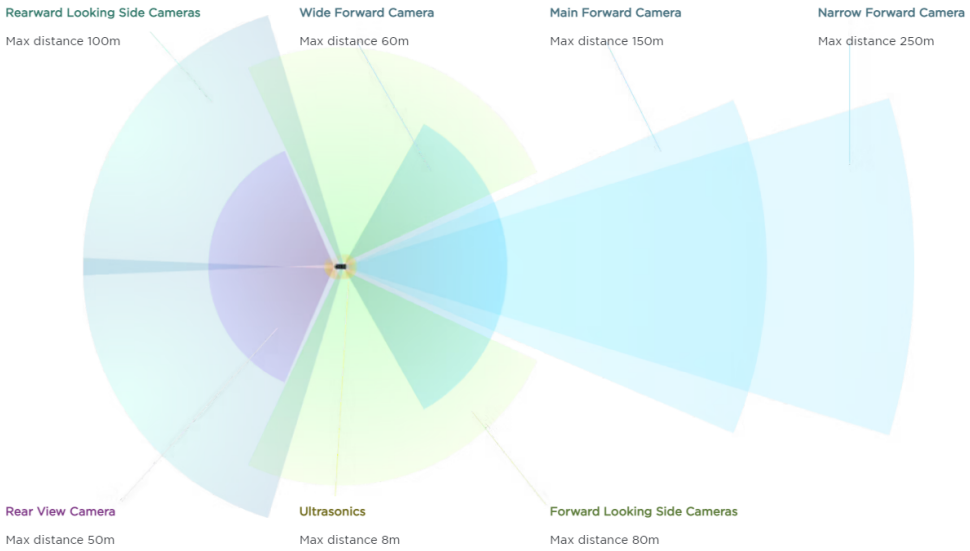
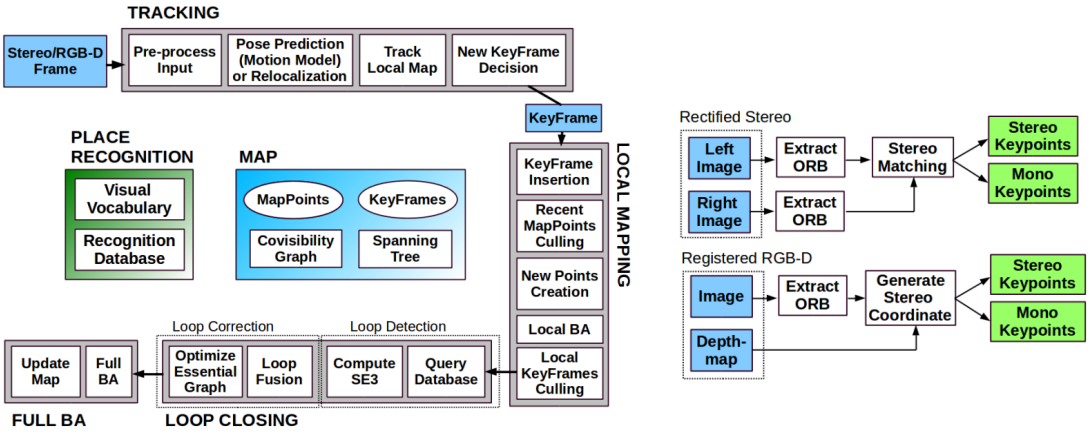


FIGURE 3.2: Sensor placement scheme of Tesla Autopilot.

to provide a 360-degree perception of the environment without blind spots 3.2. Although the binocular camera can measure depth while obtaining semantic information, autonomous driving based entirely on the binocular system requires a huge amount of calculation, so it is difficult to guarantee real-time performance. Therefore, most of the automatic driving solutions are to detect the semantic information of the object through the camera, plus the LIDAR detection distance to implement automatic driving.

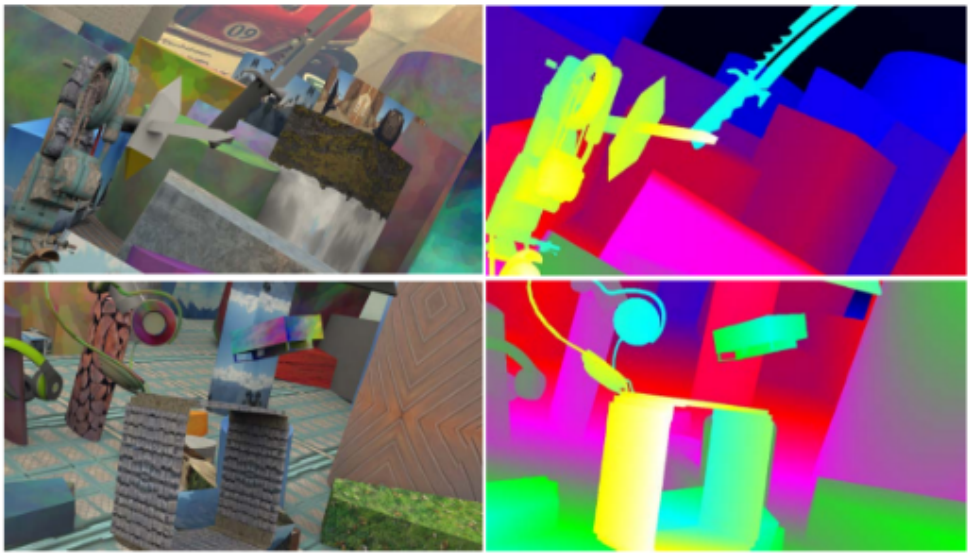


(a) System Threads and Modules. (b) Input pre-processing

FIGURE 3.3: ORB-SLAM2 mainly consists of three parallel threads: tracking, partial mapping and loop closing.

3.1.2 Robot Navigation

For AGV (Autonomous Guided Vehicle), SLAM (Simultaneous Localization And Mapping) is often used to build navigation systems. Early researchers often used LIDAR as a sensor (Chetverikov et al., 2002; Koide, Miura, and Menegatti, 2019; Rozenberszki and Majdik, 2020), because it can output accurate distance information and able to perform in real-time. However, LIDAR sensors have the disadvantages of being very expensive and low accuracy in foggy, rainy and snowy conditions. In order to solve the drawbacks of LIDAR, researchers began to study visual-based SLAM. The most representative of visual slam is ORB-SLAM family (Mur-Artal, Montiel, and Tardós, 2015; Mur-Artal and Tardós, 2017; Campos et al., 2021). We take ORB-SLAM2 as an example to introduce visual SLAM algorithms. As shown in figure 3.3, ORB-SLAM2 mainly includes three parts: tracking, local mapping and loop closing. During the tracking process, the algorithm search for the matched the local map features, and minimize the re-projection errors with motion-only BA (Bundle Adjustment). Local mapping is used to manage the features of local mapping and optimize the matching results through local BA. The loop closing process is used to detect and eliminate accumulated drift in a larger loop. It is mainly achieved through the comparison and optimization of essential frames. After implementing these three basic threads, full BA is used to optimize the pose-graph information and calculate the optimal structure and robot motions.



(a) Left image (b) Ground truth

FIGURE 3.4: Scene Flow dataset. (a) left image of stereo image pair, (b) ground truth disparity.

3.2 Dataset

For all our designed models, we mainly use two convincing stereo image datasets.

3.2.1 Scene Flow

Scene Flow (Mayer et al., 2016) is a large scale dataset with synthetic stereo images. It contains 35,454 training and 4370 testing image pairs with 940×540 resolutions. We report the end-point-error (EPE) for evaluations, where EPE shows the average disparity error in pixels.

3.2.2 KITTI 2015

KITTI 2015 (Menze and Geiger, 2015) contains real street scenes taken by driving a car. It includes 200 training image pairs with ground truth disparity maps collected by LiDAR and 200 other test image pairs without ground truth disparity. The size of the training and test images is 1240×376. We repot D1-all metrics as the official leaderboard.

(a) Left image

(b) Right image

(c) Ground truth disparity

FIGURE 3.5: KITTI 2015 dataset. (a) left image of stereo image pair, (b) right image of stereo image pair, (c) ground truth disparity.

3.3 Computational Complexity Matrics

Before we dive into the network design pipeline, we introduce the metrics for evaluating the computational complexity as follows:

- Parameters are the number of trainable neurons in the designed convolutional neural network;
- Multiply-Add operations(MAdd) describe the accumulated operations when training neural networks. (Molchanov et al., 2016) explain the calculation of floating point operations (FLOPs). MAdd is approximately half of FLOPs;
- Memory Access Cost (MAC) is the amount of allocating computational resource during the training process;
- Model Size shows the storage size of all trained parameters.

Chapter 4

Network Architecture

We first introduce the details about our network structure, including the architecture of the prototype PSMNet (Chang and Chen, 2018), and a series of 3D convolution kernels as the basic components of the network design pipeline.

4.1 PSMNet

In our research, we take the PSMNet as the baseline and explore a series of 3D kernel-based methods to find a lightweight and accuracy model. In Figure 4.1, we separate all 3D convolution kernels into five categories: 3D head, 3D convolution, 3D convolution with $stride = 2$, 3D deconvolution and 3D output. Table 4.1 shows the network settings of PSMNet, we optimize the network design by experimenting with the 3D kernel-based method on different stages of the architecture.

The 4D cost volumes ($disparity \times height \times width \times channel$) are formed by concatenating the left and the right feature maps f_l, f_r ($height \times width \times channel$) in Equation 4.1:

$$C(d, x, y, channel) = (Concat f_l(x, y), f_r(x - d, y), channel). \tag{4.1}$$

The 3D stacked hourglass module performs cost volume regularization. The continuous disparity map is obtained by the disparity regression process in (Kendall et al., 2017). The output disparity $\hat{d}$ is calculated as the summation of each disparity d weighted by corresponding probability $\sigma(-c_d)$,

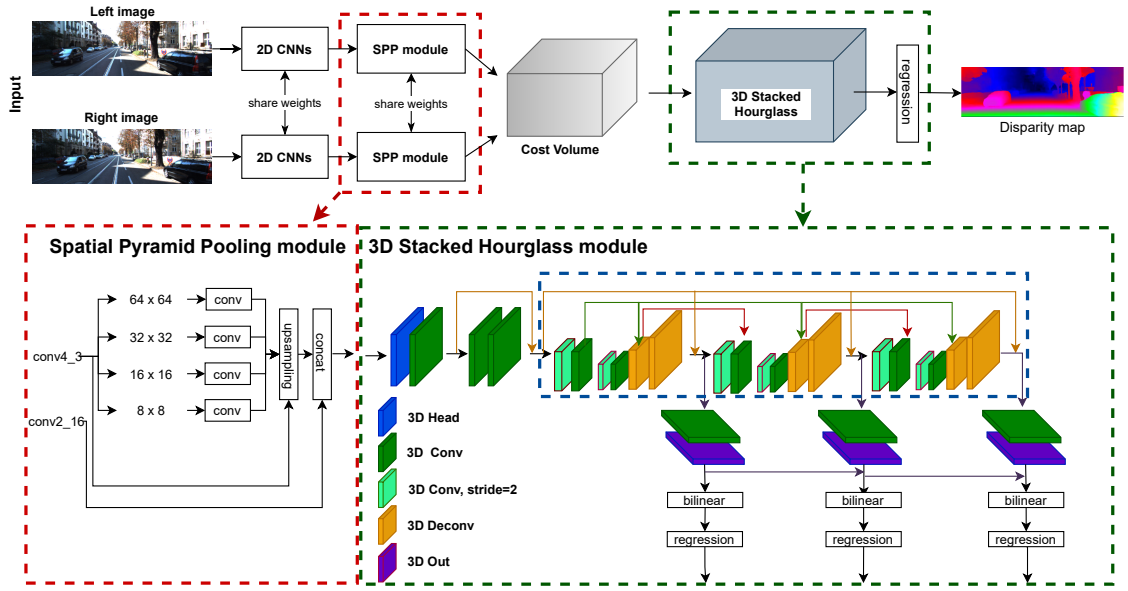


FIGURE 4.1: Architecture of PSMNet: We specified all 3D CNN kernels with different colors representing different interactions on the size and channel of cost volumes.

$$\hat{d} = \sum_{d=0}^{D_{max}} d \times \sigma(-c_d). \tag{4.2}$$

The $\sigma(.)$ denotes softmax operation, and the maximum disparity D_{max} is set to 192.

For generating a smooth disparity map, the PSMNet uses a smooth L1 loss function to train the whole architecture. The loss function of PSMNet is defined as:

$$L(d, \hat{d}) = \frac{1}{N} \sum_{i=1}^{N} smooth_{L1}(d_i - \hat{d}_i), \tag{4.3}$$

where

$$smooth_{L1}(x) = \begin{cases} 0.5x^2, & if \quad |x| < 1 \\ |x| - 0.5, & otherwise \end{cases}. \tag{4.4}$$

N is the amount of label pixels. d and $\hat{d}$ are the ground-truth disparity and predicted disparity, respectively.

4.2 Architecture of 3D Convolution Kernels

In this section, we introduce the architecture of our 3D kernel-based methods. We build all 3D kernels based on their 2D version and fit them to the 3D part of PSM-Net according to categories of layers in Figure 4.2.

4.2.1 3D MobileNetV1

As shown in Figure 4.2, 3D MobileNetV1 (Howard et al., 2017) decomposes a standard $3 \times 3 \times 3$ convolution kernel into a $3 \times 3 \times 3$ depthwise separable convolution and a $1 \times 1 \times 1$ pointwise convolution. The 3D depthwise separable convolution exploits a series of convolutional filters according to the channel number of the input cost volume and extracts local context in a channel-wise manner. Pointwise convolution walks through the cost volumes, restoring spatial information across different channels.

By isolating local context extraction and channel interaction, MobileNetV1 decreases computational complexity and model size significantly. Unlike most recent CNN architecture, MobileNetV1 excludes ResNet-like residual connections (He et al., 2016) or multi-branch operations, which makes it accessible for channel-wise operations (3D Head in Table 4.1).

4.2.2 3D MobileNetV2

Figure 4.3b shows the 3D MobileNetV2 (Sandler et al., 2018) block. It follows the main idea of MobileNetV1 by building depthwise convolution layers and pointwise convolution layers for reducing computational complexity. It also proposes inverted residual blocks with linear bottlenecks and residual connections. The linear bottlenecks increase the cost volume channels with an expansion factor for solving the problem that high dimensional targeting feature expression often collapses when operating the rectified linear unit (ReLU) activation. The 3D MobileNetV2 block with $stride = 1$ (Figure4.3-left) comprising the inverted residual structure helps to construct a deeper model as ResNet **ref-5**, while the block with

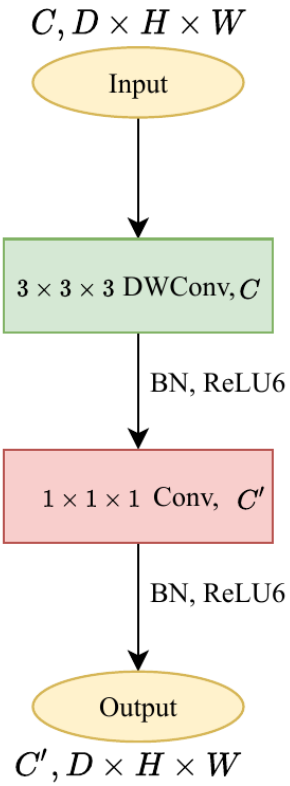


FIGURE 4.2: Architecture of 3D MobileNetV1.

stride $= 2$ (Figure 4.3-right) keeps excluding the residual connections for a smooth channel-wise operation.

4.2.3 3D ShuffleNetV1

Compared to other lightweight CNNs, ShuffleNetV1 (Zhang et al., 2018) uses $1 \times 1 \times 1$ pointwise group convolutions (GConv) for computational efficiency. As shown in figure 4.4, the symbolic channel shuffle operation helps to break through the barriers of different groups to build a more robust model. Unlike MobileNetV2, ShuffleNetV1 follows the residual structure to decrease the feature map channels to make it lightweight.

The 3D ShuffleNetV1 block with *stride* $= 1$ (Figure 4.4-left) builds the standard ResNet-like residual connections, while the *stride* $= 2$ version constructs the residual connections in another way. As shown in Figure 4.4-right, the main branch keeps the structure unchanged, downsamples the feature maps by half with a strided depthwise convolution (DWConv). On the other hand, the shortcut branch

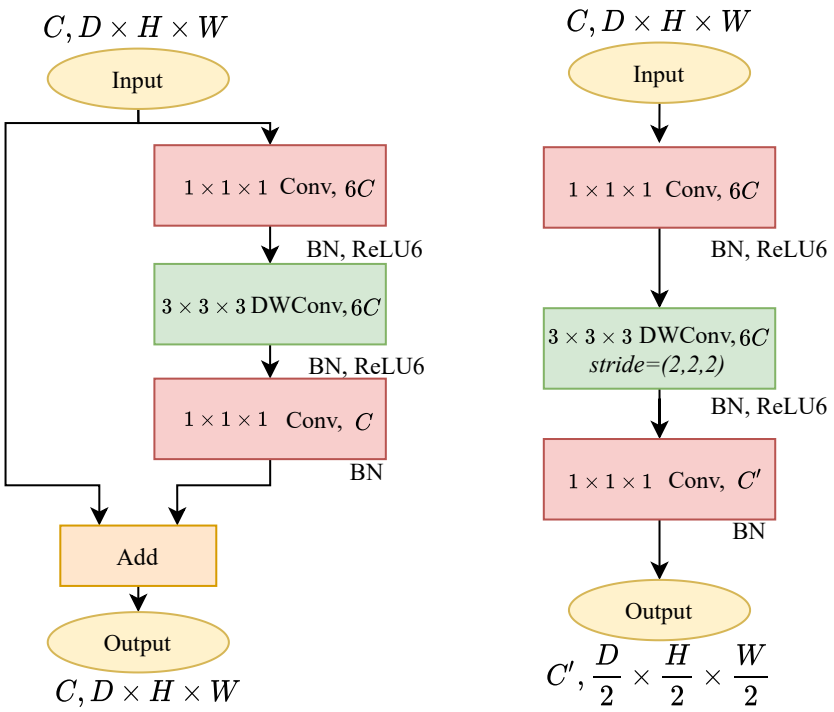


FIGURE 4.3: Architecture of 3D MobileNetV2.

utilizes average pooling to halve the feature maps. As the output feature channels of two branches are C, the concatenation results raise feature channels to $2C$.

3D ShuffleNetV2

Compared to ShuffleNetV1, ShuffleNetV2 (Ma et al., 2018) changes the $1 \times 1 \times 1$ pointwise group convolution into standard pointwise convolution. In figure 4.5, since the pointwise convolution is not grouped, the channel shuffle operation is placed after the two-branches concatenation to enable information communication between two branches.

The 3D ShuffleNetV2 block with $stride = 1$ (Figure 4.5-left) shuffles the feature channels and splits all feature maps by two with a channel split operation. Half of them remain untouched with the residual connection. Another half follows a three convolutions scheme without changing the channels. The $stride = 2$ version (Figure 4.5-right) makes use of all feature maps on each branch. Commonly, the down sampling layers contain channel increases. The main branch (right) accomplishes

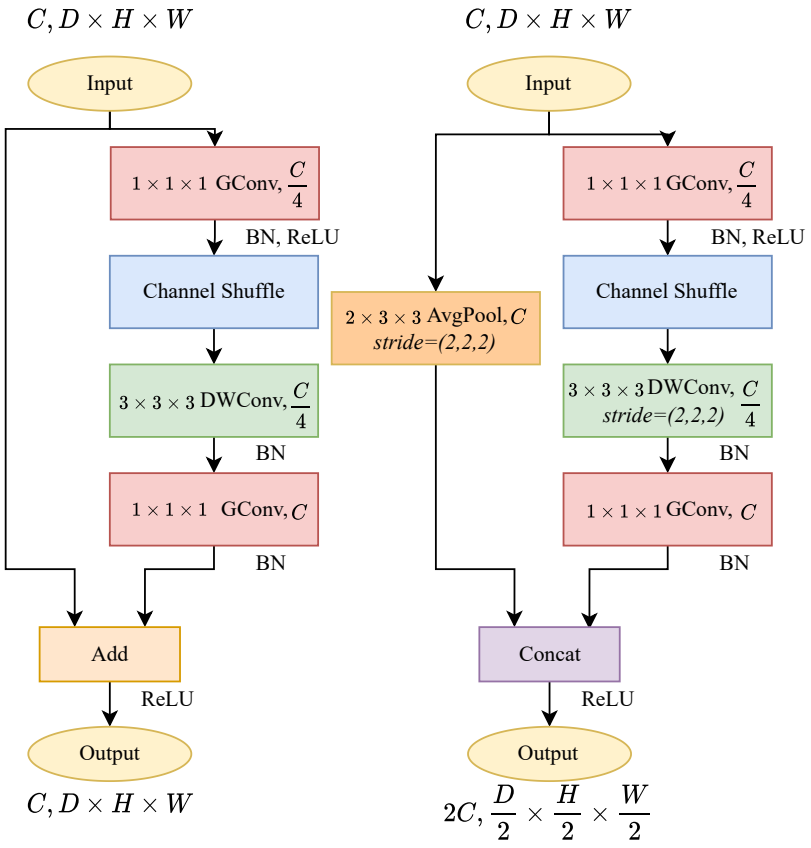


FIGURE 4.4: Architecture of 3D ShuffleNetV1.

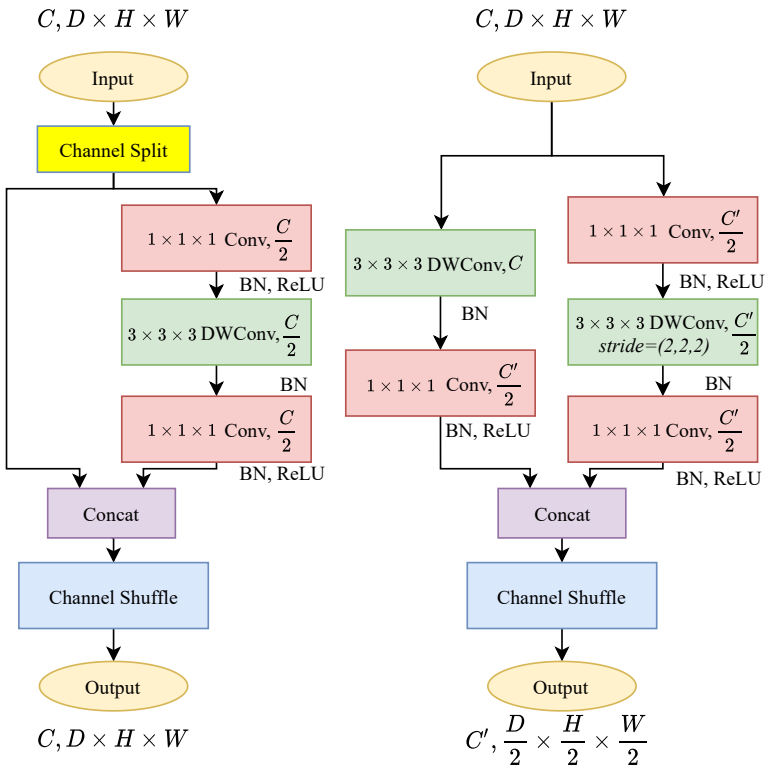


FIGURE 4.5: Architecture of 3D ShuffleNetV2.

the channel variation on the first pointwise convolution. However, the identity branch compiles the channel change after the $3 \times 3 \times 3$ depthwise convolution to keep the channel-wise dependency until the pointwise layer.

3D ECA Blocks

For the outputting cost volume followed by 3D CNNs, we build the 3D ECA (Wang et al., 2020) blocks for optimizing channel-wise attention. Figure 4.6 shows that our 3D ECA blocks aggregate cost volumes ($D \times H \times W$) along the channel with a 3D global average pooling operation. Different from other channel-wise attention modules, ECA blocks do not perform dimensionality reductions when extracting channel-wise information with 3D global average pooling. Then 1D convolution achieves information aggregation on the nearby extracted channel information. After passing a Sigmoid activation function (σ), we implement the

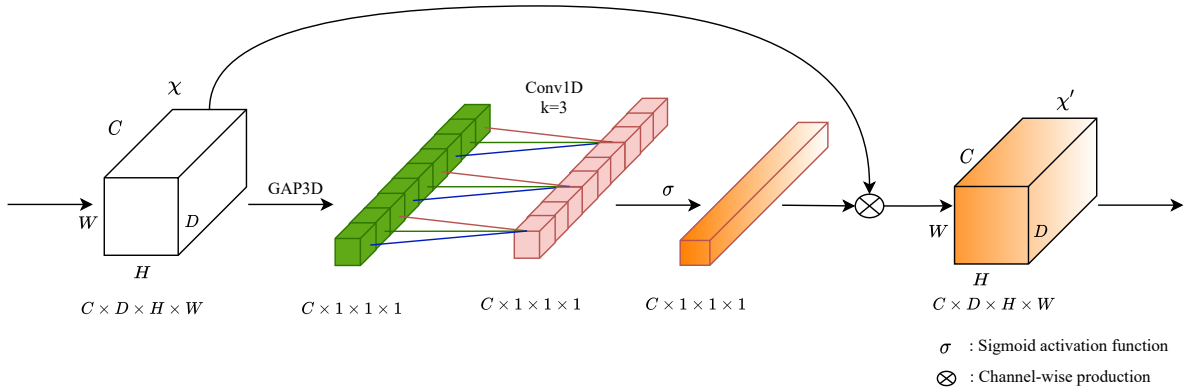


FIGURE 4.6: Architecture of 3D ECA blocks.

channel-wise product on the input cost volume to form a 3D channel-wise attention mechanism.

TABLE 4.1: Parameters of PSMNet architecture. Batch normalization and ReLU layers are used except the summation operations. H and W denote the height and width of an input pair. D means the disparities of the stereo input images.

	2D Part: Feature Extraction			**3D Part: Cost Volume Optimization**	
Layer Name	**Setting**	**Output Dimension**	**Layer Name**	**Setting**	**Output Dimension**
Stereo input	-	$[H \times W \times 3] \times 2$	Concat left and right feature maps	-	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 64$
	2D CNNs			**3D Stacked Hourglass module**	
Conv0_x	$[3 \times 3, 32] \times 3$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$	3DConv0(3D Head)	$3 \times 3 \times 3, 32$	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
Conv1_x	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$	3DConv1(3D Conv)	$[3 \times 3 \times 3, 64] \times 3$	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
Conv2_x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 16$	$\frac{1}{4}H \times \frac{1}{4}W \times 64$	3DStack1_x(3D Conv, $stride = 2$)	$3 \times 3 \times 3, 64, stride = 2$	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
Conv3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$	$\frac{1}{4}H \times \frac{1}{4}W \times 128$	3DStack2_x(3D Conv)	$3 \times 3 \times 3, 64$	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
Conv4_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3, dila = 2$	$\frac{1}{4}H \times \frac{1}{4}W \times 128$	3DStack3_x(3D Conv, $stride = 2$)	$3 \times 3 \times 3, 64, stride = 2$	$\frac{1}{16}D \times \frac{1}{16}H \times \frac{1}{16}W \times 64$
	Spatial Pyramid Pooling (SPP) module		3DStack4_x(3D Conv)	$3 \times 3 \times 3, 64$	$\frac{1}{16}D \times \frac{1}{16}H \times \frac{1}{16}W \times 64$
Branch_1	64×64, 128, Avg_pooling 1×1, 32, Conv Upsample, Biliner interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$	3DStack5_x(3D Deconv)	ConvTranspose3d $3 \times 3 \times 3, 64$	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
Branch_2	32×32, 128, Avg_pooling 1×1, 32, Conv Upsample, Biliner interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$	3DStack6_x(3D Deconv)	ConvTranspose3d $3 \times 3 \times 3, 32$	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
Branch_3	16×16, 128, Avg_pooling 1×1, 32, Conv Upsample, Biliner interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$	Out1_x(3D Conv)	$3 \times 3 \times 3, 32$	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
Branch_4	8×8, 128, Avg_pooling 1×1, 32, Conv Upsample, Biliner interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$	Out2_x(3D Out)	$3 \times 3 \times 3, 1$	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 1$
Concat[Conv2_16,Conv4_3,Branch_1,Branch_2,Branch_3,Branch_4]		$\frac{1}{4}H \times \frac{1}{4}W \times 320$	Upsampling	Trilinear interpolation	$D \times H \times W$
Fusion	3×3, 128 1×1, 32,	$\frac{1}{4}H \times \frac{1}{4}W \times 32$	-	Disparity regression	$H \times W$

Chapter 5

Network Design Pipline

In this chapter, we use the 3D kernel-based methods introduced above to replace the standard 3D CNNs in PSMNet, and design a series of comparative experiments to illustrate the impact of different 3D convolution kernels on the performance. Since we focus on discussing the role of 3D convolution kernels in the stereo matching algorithm, we completely follow the original PSMNet on the network layer setting in Table 4.1.

5.1 Implementation

We train all models with an Adam optimizer on one NVIDIA RTX 3090 GPU. During the training process, all input images are randomly cropped to 512×256. We first train our models from scratch on the Scene Flow dataset for 20 epochs with a batch size of four. The learning rate is 0.0005 constantly. Then we train the models on KITTI 2015 with Scene Flow pre-trained weights for 2000 epochs. The initial learning rate is 0.0005 and is decreased by half at 400th, 600th and 800th epochs. Since the training dataset of KITTI 2015 only contains 200 input pairs, we perform the training process with 10-fold cross validation (Ng et al., 1997) for preventing overfitting. In appendix A, we discuss the difference between normal cross validation and 10-fold cross validation during the re-implementation of the original PSMNet on KITTI 2015 dataset.

5.2 Optimize 3D Convolution Kernels on PSMNet

In Figure 4.1 and Table 4.1, we specify all 3D convolution kernels in PSMNet to five categories: 3D Head, 3D Conv, 3D Conv $stride = 2$, 3D Deconv and 3D Out. We replace the traditional CNNs in PSMNet with the 3D convolution kernels in Section 4.2 according to different functions of 3D convolution kernels.

5.2.1 3D Head

The first 3D convolution layer (3D Head) reduces the number of channels of the cost volume from 64 to 32. Among all 3D convolution kernels, MobileNetV1 builds without residual connection, which is beneficial when operating feature channels. We build 3D MobileNetV1 on the first layer.

5.2.2 3D Convoltion Layers

For all 3D convolution layers (3D Conv and 3D Conv $stride = 2$), we used MobileNetV1, MobileNetV2, ShuffleNetV1 and ShuffleNetV2 to replace the original kernels.

In Figure 4.4-right, ShuffleNetV1 with $stride = 2$ always blocks double the channels to $2C$. However, as shown in Table 4.1, $3DStack3\_x$ layer downsamples the cost volume without changing the channels. To make an impartial comparison with PSMNet, as shown in Figure 5.1, we build a ShuffleNetV1 block with downsampling without changing the number of output channels. In the main branch, we modify the channel of the last pointwise group convolution into $\frac{C}{2}$. As for the identity branch, we add a pointwise group convolution followed by the 3D average pooling layer and decrease the channels to $\frac{C}{2}$. Therefore, the output cost volumes remain with the same channel number as the PSMNet architecture.

After obtaining the results in Table 5.1 through comparative experiments, we found that 3D ShuffleNetV2 performs the best among all 3D convolution kernels. Then, we added the 3D ECA block on 3D ShuffleNetV2. The original paper only implements the ECA block on residual connection modules. However, the concatenation

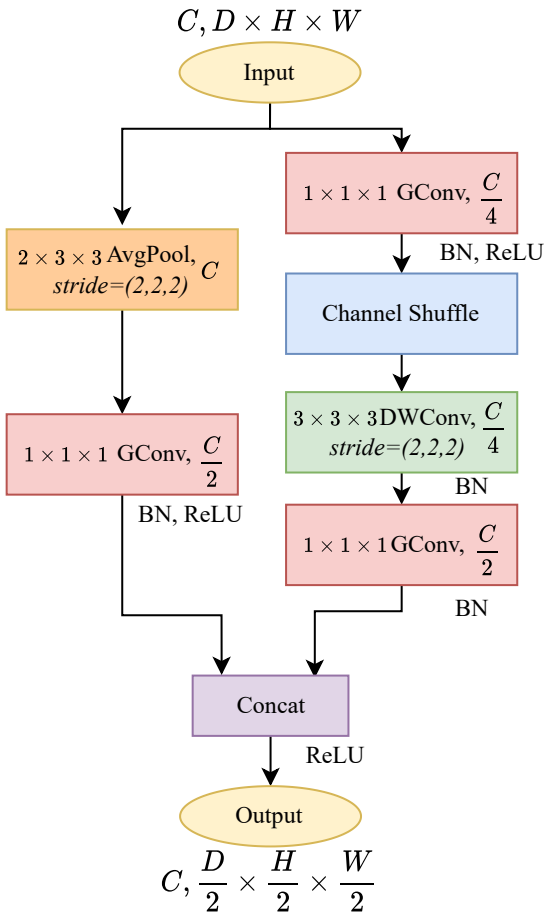


FIGURE 5.1: The implementation of 3D ShuffleNetV1 with $stride = 2$ at $3DStack3\_x$ layer. Following the PSMNet, we downsample the cost volume by half without changing the channels.

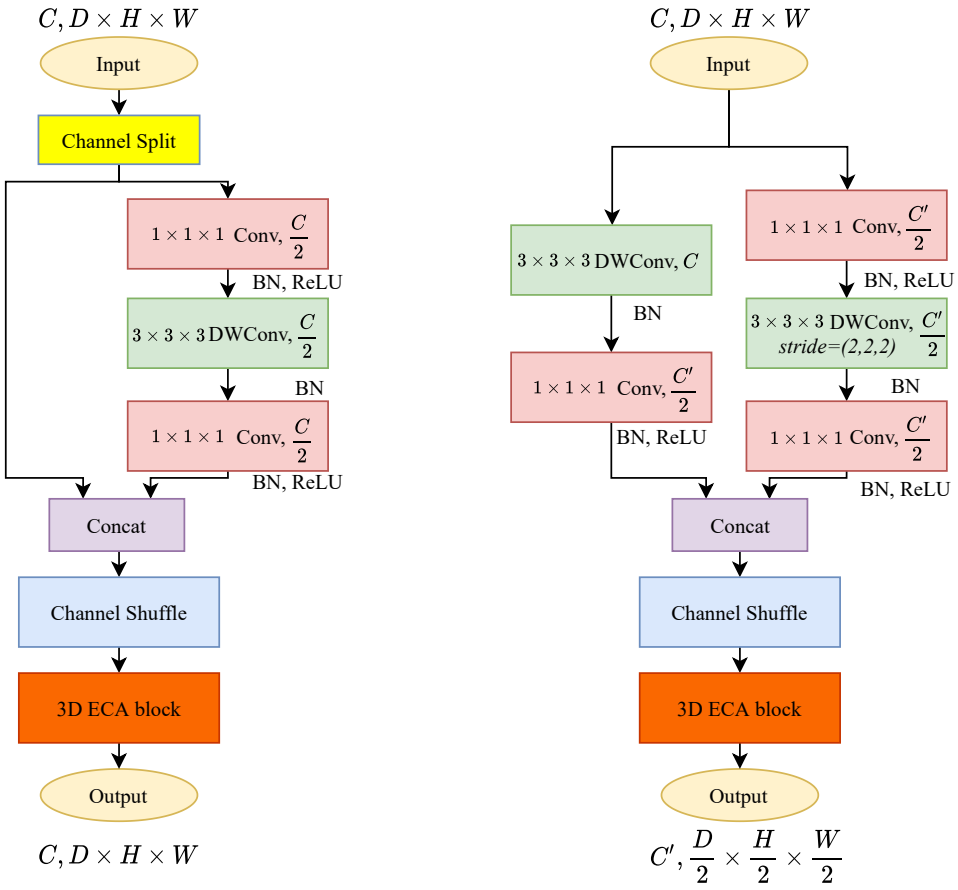


FIGURE 5.2: We build the 3D ECA blocks after the channel shuffle operations on 3D ShuffleNet V2. (a) 3D ECA-ShuffleNetV2 with $stride = 1$. (b) 3D ECA-ShuffleNetV2 with $stride = 2$.

in ShuffleNetV2 and the addition operation in the residual connection reflect different logic when operating the channels of cost volumes. We discussed the most suitable position for inserting a 3D ECA block in ShuffleNetV2 in Appendix B. The 3D ECA-ShuffleNetV2 is shown in Figure 5.2.

5.2.3 3D Deconvoltion Layers

Three dimensional (3D) transposed convolutions upsample the input cost volumes to twice the size. It first expands the input size by zero padding the cost volume of each channel, and then compiles the standard 3D convolution to introduce learning parameters for more refined textural information. In our implementation, we follow the ShuffleNetV2 as the 3D convolution. However, when inserting 0 values

to restore the size, the pointwise convolution will destroy the learned semantic information and textural information. We add upsamping with trilinear interpolation before ShuffleNetV2 for a continuous and rough cost volume, then the ShuffleNetV2 block restores the cost volume with a series of parameterized convolution layers.

Following the architecture of PSMNet in Table 4.1, we build the $3DStack5\_x$ layer by simply adding an upsampling layer as we mentioned. Since the channel needs to be reduced to half, we construct the $3DStack6\_x$ layer as illustrated in Figure 5.3. The channel split operation divides the input channels into two branches. In the main branch (right), the first pointwise convolution decreases the channels to $\frac{C}{4}$ instead $\frac{C}{2}$. In the identity branch (left), we include a pointwise convolution to reduce the channels to $\frac{C}{4}$. Eventually, we get the output cost volume with twice the size and half the channels.

5.2.4 3D Out

For the last 3D CNN layer ($Out2\_x$), we keep the 3D convolution kernel unchanged for establishing the disparity and textural information.

5.2.5 Network Design Overview

As shown in Table 5.1, in the first stage, we discuss the influence of MobileNetV1, MobileNetV2, ShuffleNetV1 and ShuffleNetV2 as the basic 3D convolution kernels of the PSMNet on the accuracy and computational complexity. Then we follow the same strategy for designing comparative experiments. We put the MobileNetV1 on the first layer for saving computation, ECA blocks on every 3D convolution kernel for boosting the model accuracy and transposed ShuffleNetV2 for switching all 3D parts of PSMNet to a lightweight method. Eventually, the computational complexity (MAdd) of PSMNet reduced from 256.66 G to 69.03 G 3D convolution kernels implementation. Parameters and model size also decrease from 5.23 M to 3.43 M, and from 21.1 M to 14.1 M, respectively. MAC increases to build more small layers during training. We reduced the computational complexity of the model

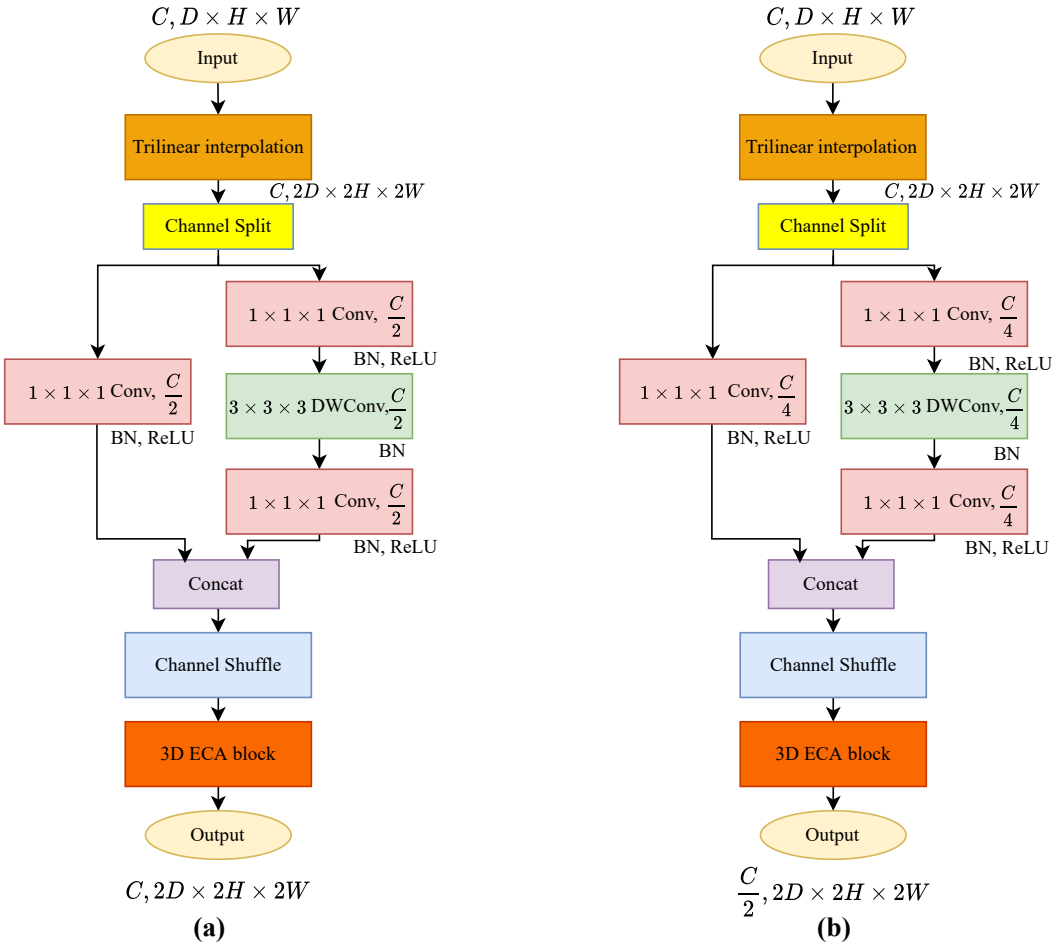


FIGURE 5.3: The trilinear interpolation upsamples the input to roughly twice the size of cost volume, then the 3D ECA-ShuffleNetV2 refines the resolution of cost volume with parameterized layers. (**a**) 3D Transposed ECA-ShuffleNetV2 at $3DStack5\_x$ layer. (**b**) 3D Transposed ECA-ShuffleNetV2 at $3DStack6\_x$ layer.

while keeping the performance almost unchanged. In appendix C, we parameterize of all models in a layer-manner to emphasize how different 3D convolution kernels change the model computational complexity.

TABLE 5.1: Experiments for 3D convolution kernels on PSMNet. We calculate the MAdd with 512×256 resolution as input. EPE is the end-point-error on the Scene Flow dataset. D1-all denotes the evaluation of the KITTI 2015 leaderboard. MobelNetV2* denotes implementation of the kernel with $expansion \quad ratio = 2$ considering the MAC of the model. At each stage, we select the best 3D convolution kernel (bold font) and optimize it on the model.

3D Conv Kernel	EPE	D1-all	Parameters(Million)	MAdd(Gb)	MAC(Mb)	Model size(Mb)
3D Convolution Layers						
Baseline(re-implemented)	1.123	2.41	5.23	256.66	3895	21.1
MobileNetV1	1.252	2.68	3.97	168.96	5421	16.2
MobileNetV2*	1.215	2.60	4.11	175.93	6097	16.8
ShuffleNetV1	1.295	2.88	3.91	164.39	4651	16.0
ShuffleNetV2	1.241	2.62	3.96	168.03	5071	16.2
3D Head						
3D CNN	1.241	2.62	3.96	168.03	5071	16.2
MobileNetV1 (Head)	1.233	2.61	3.91	147.81	5411	16.0
3D ECA Blocks						
Without ECA Blocks	1.233	2.61	3.91	147.81	5411	16.0
ECA Blocks	1.160	2.50	3.91	147.92	5835	16.0
3D Deconvoltion Layers						
3D Transposed CNN	1.160	2.50	3.91	147.92	5835	16.0
Transposed ShuffleNetV2	1.124	2.43	3.43	69.03	6771	14.1

Chapter 6

Benchmark Results

In Section 5, we built 3D convolution kernels and explored the best combination of 3D kernels with comparative experiments. Due to the number of comparative experiments being relatively large, for saving time, we only train all models to close results without convergence. For benchmarking, we train our model on two NVIDIA V100 for setting the batch size to eight. Since now we have a larger batch size, we double the learning rate for two stages of training. For Scene Flow, the learning is 0.001 constantly. For KITTI 2015, the initial learning rate is 0.001 and is decreased by half at the 400th, 600th and 800th epochs. We perform 10-fold cross validation on the first 1000 epochs. Then we train another 1000 epochs without 10-fold cross validation with a 0.000125 learning rate.

We evaluate our model on Scene Flow and KITTI 2015. In Table 6.1, our model achieves accurate results on these datasets with a low-cost MAdd in terms of computation. For Scene Flow, our model outperforms the original PSMNet 0.18 on end-point-errors. As for KITTI 2015, Table 6.2 demonstrates that our model achieves similar results to PSMNet, only taking 26.9% of the MAdd. Meanwhile, our model surpasses the PSMNet significantly in foreground pixels (D1-fg in the table). Figure 6.1 further visualizes the disparity estimation result on the KITTI 2015 test set.

TABLE 6.1: Evaluation results on the Scene Flow dataset. Our model is competitive with other top-performing models.

Method	**Ours**	**Baseline**	**GC-Net**	**GANet**	**DeepPruner-Best**	**DispNetC**	**StereoNet**	**JDCNet**
EPE	0.91	1.09	2.51	0.84	0.86	1.68	1.10	0.83

TABLE 6.2: Evaluation results on the KITTI 2015 dataset. The first seven methods are accurate methods (included the baseline). The other five are considered fast methods. Our model not only achieves results comparable to those of some accurate methods but also requires significantly less computational complexity. We only calculated the MAdd of some representative models.

Method	All(%)			Noc(%)			Runtime(s)	MAdd(G)
	D1-bg	D1-fg	D1-all	D1-bg	D1-fg	D1-all		
Baseline	1.86	4.62	2.32	1.71	4.31	2.14	0.41	256.66
PSMNet-lite (Ours)	1.91	4.56	2.35	1.75	4.06	2.13	0.63	69.03
MC-CNN	2.89	8.88	3.89	2.48	7.64	3.33	67	-
GC-Net	2.21	6.16	2.87	2.02	5.58	2.61	0.9	733.36
GwcNet	1.74	3.93	2.11	1.61	3.49	1.92	0.32	247.6
DeepPruner-Fast	1.87	3.56	2.15	1.71	3.18	1.95	0.18	-
GANet-15	1.55	3.82	1.93	1.40	3.37	1.73	0.36	-
CSN	1.59	4.03	2.00	1.43	3.55	1.78	0.6	-
SMD-Net	1.69	4.01	2.08	1.54	3.70	1.89	0.41	-
StereoNet	4.30	7.45	4.83	-	-	-	0.015	47.08
DispNetC	4.32	4.41	4.34	4.11	3.72	4.05	0.03	-
DeepPruner-Best	2.32	3.91	2.59	2.13	3.43	2.35	0.06	-
AANet	1.99	5.39	2.55	1.80	4.93	2.32	0.062	-
Fast DS-CS	2.83	4.31	3.08	2.53	3.74	2.73	0.02	-
JDCNet	1.91	4.47	2.33	1.73	3.86	2.08	0.079	-

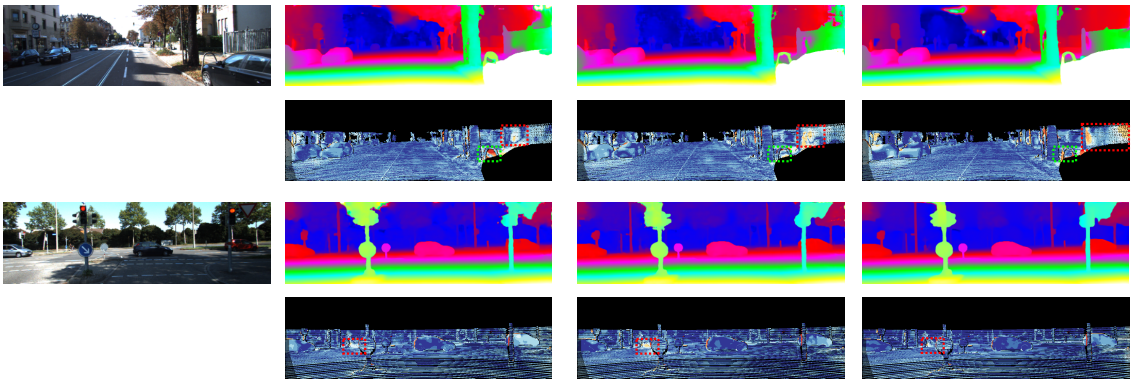


FIGURE 6.1: Visualization of prediction error on KITTI test set (red and yellow pixels denote error disparities). The red boxes denote foreground regions and green boxes denote background regions. (**a**) Left image. (**b**) Our model. (**c**) PSMNet. (**d**) AANet.

Chapter 7

Discussion

In Table 6.2, compared to other studies, our model contains very little computational complexity. However, the runtime is longer than that of the original PSMNet. As mentioned in (Qin et al., 2018; Lu, Zhang, and Wang, 2021), the reason may be that the cuDNN library does not fully support depthwise convolutions and pointwise convolutions. For the GPU platform of the cuDNN library, the optimization of classic convolutions on end-to-end training is better. So, it will be faster than some lightweight convolutions, although it produces more computation theoretically. We present the runtime of all models in Table 7.1.

TABLE 7.1: Inference time of all models in the network design pipeline. We use one NVIDIA RTX 3090 for all implementations. "+ MobileNetV1" replaces the 3*DConv*0 with the 3D MobileNetV1 module. "+ ECA block" add 3D ECA block on all 3D kernels. "+ Transposed ShuffleNetV2" replaces the 3*DStack*5_*x* and 3*DStack*6_*x* deconvolution layers with 3D Transposed ECA-ShuffleNetV2.

Method	**MobileNetV1**	**MobileNetV2**	**ShuffleNetV1**	**ShuffleNetV2**	**+ MobileNetV1 Head**	**+ ECA Blocks**	**+ Transposed ShuffleNetV2**
Runtime (s)	0.48	0.53	0.35	0.40	0.41	0.60	0.63

Chapter 8

Conclusions

In this paper, based on PSMNet as a prototype, we design a series of kernel-based methods aiming for a lightweight and accurate model without modifying the original architecture. By optimizing 3D convolution kernels with corresponding kernel-based methods, our model greatly reduces computational complexity and achieves comparable results to the modern stereo matching algorithms. In future work, as we mentioned in 7, we will investigate the implementation of 3D depthwise convolutions and 3D pointwise convolutions on the cuDNN library and improve our model to become faster in training and inference.

Appendix A

10-Fold Cross Validation

Since KITTI 2015 has a small amount of training set, during the re-implementation of PSMNet, we examine the normal training manner and 10-fold cross validation with the same pretrained model on Scene Flow data set.

TABLE A.1: The re-implementation of PSMNet with cross validation and 10-fold cross validation. Both follow the same implementation details in Section 5.1. We calculate the average 3-pixel error of all image pairs in the whole training set with the model we submitted for evaluation.

Method	**All(%)**			**Noc(%)**			**Loss**	**3-pixel Error in Training Set**
	D1-bg	**D1-fg**	**D1-all**	**D1-bg**	**D1-fg**	**D1-all**		
Cross Validation	2.03	4.89	2.51	1.86	4.53	2.30	0.291	0.707
10-fold Cross Validation	1.94	4.76	2.41	1.75	4.42	2.22	0.314	0.747

As illustrated in Table A.1, the cross validation training manner leads to a lower loss and more accurate disparity map on the training set, while the evaluation result is worse. This situation reflects it has an explicit overfitting on the training set.

Appendix B

3D ECA-ShuffleNetV2 Blocks

In order to achieve the attention mechanism on each channel of the cost volumes, we build the 3D ECA block after concatenation of the main branch and the identity branch. Two potential 3D ECA-ShuffleNetV2 blocks are shown in figure B.1. For saving time, we only train both potential 3D ECA-ShuffleNetV2 blocks on Scene Flow data set for 10 epochs and evaluate the model by training loss.

We show the training performance in Figure B.2, the (b) architecture slightly outperforms than (a) architecture. For model (a), after it learning the channel-wise attention information, the channel shuffle operation disrupts the channel order of cost volumes. Based on the results, we think that the order of channels contains part of the spatial representation ability of the model.

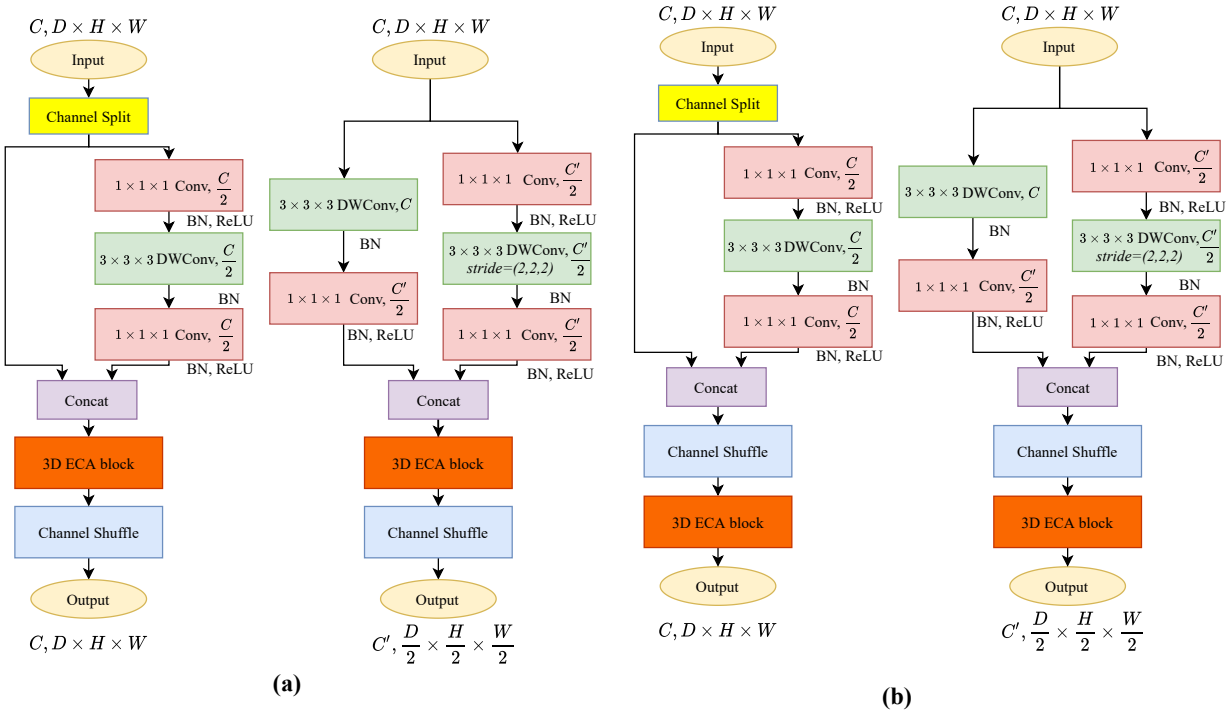


FIGURE B.1: We built ECA block both before and after channel shuffle operation to find a more appropriate architecture of 3D ECA-ShuffleNetV2. (a) before channel shuffle operation. (b) after channel shuffle operation.

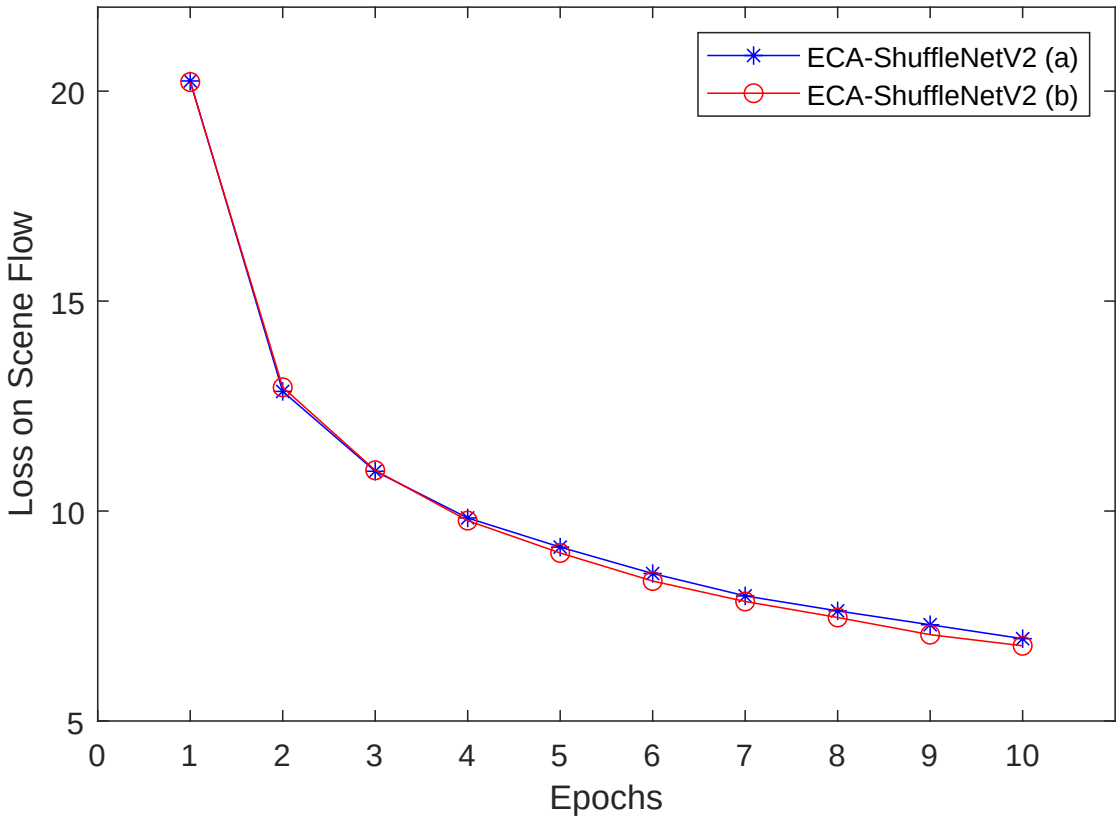


FIGURE B.2: Training loss on Scene Flow data set with 3D ECA-ShuffeNetV2 (a) & (b).

Appendix C

Parameterize Model Details on Optimizing 3D Convolution Kernels

In order to reflect the influence of the reduction of computational complexity on building different 3D convolution kernels, we show the parameterize model details on Tabel C.1. Since our work mainly focused on the regularization of the cost volume, we omit the model details of feature extraction to one output as $2DCNN$. When only considering the 3D CNNs in the model, we reduce the parameters and MAdd from $1.887M$ to $0.085M$ (95.50%) and from $198.47G$ to $10.84G$ (94.53%).

TABLE C.1: Parameterize model details on all 3D convolution kernels in our network design pipeline. The numbers in bold denote the modified layers compare to former stages.

LayerMethod	Basline		M_V1		M_V2		S_V1		S_V2		+M_V1_Head		+ECA		+Transposed S_V2	
	Para(M)	MAdd(G)	Para(M)	MAdd(G)	Para(M)	MAdd(G)	Para(M)	MAdd(G)	Para(M)	MAdd(G)	Para(M)	MAdd(G)	Para(M)	MAdd(G)	Para(M)	MAdd(G)
2D CNNs	3.340	58.191	3.340	58.191	3.340	58.191	3.340	58.191	3.340	58.191	3.340	58.191	3.340	58.191	3.340	58.191
3DConv0	0.055	21.781	0.055	21.781	0.055	21.781	0.055	21.781	0.055	21.781	**0.004**	**1.560**	**0.004**	**1.573**	0.004	1.573
3DConv1	0.083	32.716	**0.006**	**2.378**	**0.006**	**2.416**	**0.001**	**0.566**	**0.003**	**1.265**	0.003	1.265	**0.003**	**1.302**	0.003	1.302
3DStack1_1	0.055	2.727	**0.003**	**0.153**	**0.008**	**1.183**	**0.001**	**0.117**	**0.005**	**0.642**	0.005	0.642	**0.005**	**0.645**	0.005	0.645
3DStack2_1	0.111	5.442	**0.006**	**0.299**	**0.020**	**1.019**	**0.001**	**0.060**	**0.003**	**0.156**	0.003	0.156	**0.003**	**0.159**	0.003	0.159
3DStack3_1	0.111	0.681	**0.006**	**0.037**	**0.020**	**0.496**	**0.002**	**0.026**	**0.008**	**0.143**	0.008	0.143	**0.008**	**0.143**	0.008	0.143
3DStack4_1	0.111	0.681	**0.006**	**0.037**	**0.020**	**0.127**	**0.001**	**0.007**	**0.003**	**0.019**	0.003	0.019	**0.003**	**0.020**	0.003	0.020
3DStack5_1	0.111	5.442	0.111	5.442	0.111	5.442	0.111	5.442	0.111	5.442	0.111	5.442	0.111	5.422	**0.003**	**0.159**
3DStack6_1	0.055	21.768	0.055	21.768	0.055	21.768	0.055	21.768	0.055	21.768	0.055	21.768	0.055	21.768	**0.002**	**0.755**
3DStack1_2	0.055	2.727	**0.003**	**0.153**	**0.008**	**1.183**	**0.001**	**0.117**	**0.005**	**0.642**	0.005	0.642	**0.005**	**0.645**	0.005	0.645
3DStack2_2	0.111	5.442	**0.006**	**0.299**	**0.020**	**1.019**	**0.001**	**0.060**	**0.003**	**0.156**	0.003	0.156	**0.003**	**0.159**	0.003	0.159
3DStack3_2	0.111	0.681	**0.006**	**0.037**	**0.020**	**0.496**	**0.002**	**0.026**	**0.008**	**0.143**	0.008	0.143	**0.008**	**0.143**	0.008	0.143
3DStack4_2	0.111	0.681	**0.006**	**0.037**	**0.020**	**0.127**	**0.001**	**0.007**	**0.003**	**0.019**	0.003	0.019	**0.003**	**0.020**	0.003	0.020
3DStack5_2	0.111	5.442	0.111	5.442	0.111	5.442	0.111	5.442	0.111	5.442	0.111	5.442	0.111	5.422	**0.003**	**0.159**
3DStack6_2	0.055	21.768	0.055	21.768	0.055	21.768	0.055	21.768	0.055	21.768	0.055	21.768	0.055	21.768	**0.002**	**0.755**
3DStack1_3	0.055	2.727	**0.003**	**0.153**	**0.008**	**1.183**	**0.001**	**0.117**	**0.005**	**0.642**	0.005	0.642	**0.005**	**0.645**	0.005	0.645
3DStack2_3	0.111	5.442	**0.006**	**0.299**	**0.020**	**1.019**	**0.001**	**0.060**	**0.003**	**0.156**	0.003	0.156	**0.003**	**0.159**	0.003	0.159
3DStack3_3	0.111	0.681	**0.006**	**0.037**	**0.020**	**0.496**	**0.002**	**0.026**	**0.008**	**0.143**	0.008	0.143	**0.008**	**0.143**	0.008	0.143
3DStack4_3	0.111	0.681	**0.006**	**0.037**	**0.020**	**0.127**	**0.001**	**0.007**	**0.003**	**0.019**	0.003	0.019	**0.003**	**0.020**	0.003	0.020
3DStack5_3	0.111	5.442	0.111	5.442	0.111	5.442	0.111	5.442	0.111	5.442	0.111	5.442	0.111	5.422	**0.003**	**0.159**
3DStack6_3	0.055	21.768	0.055	21.768	0.055	21.768	0.055	21.768	0.055	21.768	0.055	21.768	0.055	21.768	**0.002**	**0.755**
Out1_1	0.028	10.91	**0.002**	**0.793**	**0.002**	**0.805**	**0.0004**	**0.189**	**0.001**	**0.422**	0.001	0.422	**0.001**	**0.434**	0.001	0.434
Out2_1	0.001	0.340	0.001	0.340	0.001	0.340	0.001	0.340	0.001	0.340	0.001	0.340	0.001	0.340	0.001	0.340
Out1_2	0.028	10.91	**0.002**	**0.793**	**0.002**	**0.805**	**0.0004**	**0.189**	**0.001**	**0.422**	0.001	0.422	**0.001**	**0.434**	0.001	0.434
Out2_2	0.001	0.340	0.001	0.340	0.001	0.340	0.001	0.340	0.001	0.340	0.001	0.340	0.001	0.340	0.001	0.340
Out1_3	0.028	10.91	**0.002**	**0.793**	**0.002**	**0.805**	**0.0004**	**0.189**	**0.001**	**0.422**	0.001	0.422	**0.001**	**0.434**	0.001	0.434
Out2_3	0.001	0.340	0.001	0.340	0.001	0.340	0.001	0.340	0.001	0.340	0.001	0.340	0.001	0.340	0.001	0.340
3D CNNs	1.887	198.470	0.631	110.766	0.772	117.737	0.571	106.194	0.619	109.842	0.568	89.621	0.568	89.728	0.085	10.840
Full Model	5.227	256.661	3.971	168.957	4.112	175.928	3.912	164.385	3.959	168.033	3.908	147.812	3.908	147.919	3.425	69.031

Bibliography

Apollo (2021). *Autonomous Driving Solution*. Tech. rep. Baidu. URL: https://apollo.auto/minibus/index.html.

Autopilot (2021). *Future of Driving*. Tech. rep. Tesla. URL: https://www.tesla.com/autopilot.

Barnes, Connelly et al. (2009). "PatchMatch: A randomized correspondence algorithm for structural image editing". In: *ACM Trans. Graph.* 28.3, p. 24.

Campos, Carlos et al. (2021). "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM". In: *IEEE Transactions on Robotics*.

Chabra, Rohan et al. (2019). "Stereodrnet: Dilated residual stereonet". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11786–11795.

Chang, Jia-Ren and Yong-Sheng Chen (2018). "Pyramid Stereo Matching Network". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE. DOI: 10.1109/cvpr.2018.00567.

Chetverikov, Dmitry et al. (2002). "The trimmed iterative closest point algorithm". In: *Object recognition supported by user interaction for service robots*. Vol. 3. IEEE, pp. 545–548.

Chollet, François (2017). "Xception: Deep learning with depthwise separable convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258.

Fu, Zehua and Mohsen Ardabilian Fard (2018). "Learning confidence measures by multi-modal convolutional neural networks". In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, pp. 1321–1330.

Gong, Minglun et al. (2007). "A performance study on different cost aggregation approaches used in real-time stereo matching". In: *International Journal of Computer Vision* 75.2, pp. 283–296.

Gu, Xiaodong et al. (2020). "Cascade cost volume for high-resolution multi-view stereo and stereo matching". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2495–2504.

Guo, Xiaoyang et al. (2019). "Group-wise correlation stereo network". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3273–3282.

Haeusler, Ralf, Rahul Nair, and Daniel Kondermann (2013). "Ensemble Learning for Confidence Measures in Stereo Vision". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. DOI: 10.1109/cvpr.2013.46.

Hamzah, Rostam Affendi, Rosman Abd Rahim, and Zarina Mohd Noh (2010). "Sum of Absolute Differences algorithm in stereo correspondence problem for stereo matching in computer vision application". In: *2010 3rd International Conference on Computer Science and Information Technology*. IEEE. DOI: 10.1109/iccsit.2010.5565062.

Hane, Christian, Torsten Sattler, and Marc Pollefeys (2015). "Obstacle detection for self-driving cars using only monocular cameras and wheel odometry". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. DOI: 10.1109/iros.2015.7354095.

He, Kaiming et al. (Dec. 2015a). "Deep Residual Learning for Image Recognition". In: arXiv: 1512.03385 [cs.CV].

– (2015b). "Spatial pyramid pooling in deep convolutional networks for visual recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 37.9, pp. 1904–1916.

– (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

He, Kaiming et al. (2017). "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE. DOI: 10.1109/iccv.2017.322.

Hirschmuller, Heiko (2008). "Stereo Processing by Semiglobal Matching and Mutual Information". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.2, pp. 328–341. DOI: 10.1109/TPAMI.2007.1166.

Hirschmuller, Heiko and Daniel Scharstein (2008). "Evaluation of stereo matching costs on images with radiometric differences". In: *IEEE transactions on pattern analysis and machine intelligence* 31.9, pp. 1582–1599.

Hong, Li and George Chen (2004). "Segment-based stereo matching using graph cuts". In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 1. IEEE, pp. I–I.

Hosni, Asmaa et al. (2009). "Local stereo matching using geodesic support weights". In: *2009 16th IEEE International Conference on Image Processing (ICIP)*. IEEE, pp. 2093–2096.

Howard, Andrew G et al. (2017). "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861*.

Hu, Jie, Li Shen, and Gang Sun (2018). "Squeeze-and-excitation networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141.

Iandola, Forrest N et al. (2016). "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size". In: *arXiv preprint arXiv:1602.07360*.

Intille, Stephen S. and Aaron F. Bobick (1994). "Disparity-space images and large occlusion stereo". In: *Computer Vision — ECCV '94*. Springer Berlin Heidelberg, pp. 179–186. DOI: 10.1007/bfb0028349.

Kendall, A. et al. (2017). "End-to-End Learning of Geometry and Context for Deep Stereo Regression". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, pp. 66–75. DOI: 10.1109/ICCV.2017.17. URL: https://doi.ieeecomputersociety.org/10.1109/ICCV.2017.17.

Khamis, Sameh et al. (2018). "Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 573–590.

Koide, Kenji, Jun Miura, and Emanuele Menegatti (2019). "A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement". In: *International Journal of Advanced Robotic Systems* 16.2, p. 1729881419841532.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012a). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e9Paper.pdf.

– (2012b). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25, pp. 1097–1105.

Leung, Carlos, Ben Appleton, and Changming Sun (2008). "Iterated dynamic programming and quadtree subregioning for fast stereo matching". In: *Image and Vision Computing* 26.10, pp. 1371–1383. DOI: 10.1016/j.imavis.2007.11.013.

Li, Xiang et al. (2019). "Selective kernel networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 510–519.

Lin, Tsung-Yi et al. (2017a). "Focal Loss for Dense Object Detection". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE. DOI: 10.1109/iccv.2017.324.

Lin, Tsung-Yi et al. (2017b). "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988.

Lu, Gangzhao, Weizhe Zhang, and Zheng Wang (2021). "Optimizing Depthwise Separable Convolution Operations on GPUs". In: *IEEE Transactions on Parallel and Distributed Systems*.

Ma, Ningning et al. (2018). "Shufflenet v2: Practical guidelines for efficient cnn architecture design". In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131.

Ma, Ziyang et al. (2013). "Constant time weighted median filtering for stereo matching and beyond". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 49–56.

Mayer, Nikolaus et al. (2016). "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*, pp. 4040–4048.

Mei, Xing et al. (2011). "On building an accurate stereo matching system on graphics hardware". In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE. DOI: 10.1109/iccvw.2011.6130280.

Menze, Moritz and Andreas Geiger (2015). "Object scene flow for autonomous vehicles". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3061–3070.

Mobileye (2021). *Our Technology*. Tech. rep. Mobileye. URL: https://www.mobileye.com/our-technology/.

Molchanov, Pavlo et al. (2016). "Pruning convolutional neural networks for resource efficient inference". In: *arXiv preprint arXiv:1611.06440*.

Mur-Artal, Raúl, J. M. M. Montiel, and Juan D. Tardós (2015). "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics* 31.5, pp. 1147–1163. DOI: 10.1109/TRO.2015.2463671.

Mur-Artal, Raul and Juan D Tardós (2017). "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras". In: *IEEE transactions on robotics* 33.5, pp. 1255–1262.

Nalpantidis, Lazaros, Georgios Ch. Sirakoulis, and Antonios Gasteratos (2007). "Review of stereo matching algorithms for 3D vision". In.

Newell, Alejandro, Kaiyu Yang, and Jia Deng (2016). "Stacked hourglass networks for human pose estimation". In: *European conference on computer vision*. Springer, pp. 483–499.

Ng, Andrew Y et al. (1997). "Preventing" overfitting" of cross-validation data". In: *ICML*. Vol. 97. Citeseer, pp. 245–253.

Noh, Zakiah, Mohd Shahrizal Sunar, and Zhigeng Pan (2009). "A Review on Augmented Reality for Virtual Heritage System". In: *Learning by Playing. Game-based*

Education System Design and Development. Springer Berlin Heidelberg, pp. 50–61. DOI: 10.1007/978-3-642-03364-3_7.

Parker, Andrew (2004). *In the blink of an eye : how vision sparked the big bang of evolution*. New York: Basic Books. ISBN: 9780465054381.

Qin, Zheng et al. (2018). "Diagonalwise refactorization: An efficient training method for depthwise convolutions". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–8.

Ren, Shaoqing et al. (2015). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

Richardt, Christian et al. (2010). "Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid". In: *European conference on Computer vision*. Springer, pp. 510–523.

Ronneberger, O., P.Fischer, and T. Brox (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Vol. 9351. LNCS. (available on arXiv:1505.04597 [cs.CV]). Springer, pp. 234–241. URL: http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a.

Rozenberszki, Dávid and András L Majdik (2020). "LOL: Lidar-only Odometry and Localization in 3D point cloud maps". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 4379–4385.

Russakovsky, Olga et al. (Sept. 2014). "ImageNet Large Scale Visual Recognition Challenge". In: arXiv: 1409.0575 [cs.CV].

Samadi, Masoud and Mohd Fauzi Bin Othman (2013). "A new fast and robust stereo matching algorithm for robotic systems". In.

Sandler, Mark et al. (2018). "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520.

Scharstein, Daniel and Chris Pal (2007). "Learning Conditional Random Fields for Stereo". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. DOI: 10.1109/cvpr.2007.383191.

Scharstein, Daniel and Richard Szeliski (2002). "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms." In: *International Journal of Computer Vision* 47.1/3, pp. 7–42. DOI: 10.1023/a:1014573219977.

Subaru (2021). *Subaru EyeSight*. Tech. rep. Subaru. URL: https://www.subaru.com/engineering/eyesight.html.

Szegedy, Christian et al. (Dec. 2015). "Rethinking the Inception Architecture for Computer Vision". In: arXiv: 1512.00567 [cs.CV].

Visual Fields: Examination and Interpretation (Nov. 2010). OXFORD UNIV PR. 336 pp. ISBN: 0195389689. URL: https://www.ebook.de/de/product/13154314/visual_fields_examination_and_interpretation.html.

Wang, Qilong et al. (2020). "ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11531–11539. DOI: 10.1109/CVPR42600.2020.01155.

Xu, Haofei and Juyong Zhang (2020). "Aanet: Adaptive aggregation network for efficient stereo matching". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1959–1968.

Yang, Y., A. Yuille, and J. Lu (1993). "Local, global, and multilevel stereo matching". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Comput. Soc. Press. DOI: 10.1109/cvpr.1993.340969.

Zbontar, Jure and Yann LeCun (2015a). "Computing the stereo matching cost with a convolutional neural network". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. DOI: 10.1109/cvpr.2015.7298767.

– (2015b). "Computing the stereo matching cost with a convolutional neural network". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1592–1599.

Zenati, Nadia and Noureddine Zerhouni (2007). "Dense Stereo Matching with Application to Augmented Reality". In: *2007 IEEE International Conference on Signal Processing and Communications*. IEEE. DOI: 10.1109/icspc.2007.4728616.

Zhang, Feihu et al. (2019). "Ga-net: Guided aggregation net for end-to-end stereo matching". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 185–194.

Zhang, Ke et al. (2009). "Robust stereo matching with fast Normalized Cross-Correlation over shape-adaptive regions". In: *2009 16th IEEE International Conference on Image Processing (ICIP)*. IEEE. DOI: 10.1109/icip.2009.5413502.

Zhang, Li and Steven M. Seitz (2007). "Estimating Optimal Parameters for MRF Stereo from a Single Image Pair". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.2, pp. 331–342. DOI: 10.1109/tpami.2007.36.

Zhang, Xiangyu et al. (2018). "Shufflenet: An extremely efficient convolutional neural network for mobile devices". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848–6856.

Zhang, Youmin et al. (2020). "Adaptive unimodal cost volume filtering for deep stereo matching". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07, pp. 12926–12934.

Zhao, Hengshuang et al. (2017). "Pyramid scene parsing network". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2881–2890.