Optimizing and Improving Deep Learning Methods for Stereo Matching

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Rafia Rahim
aus Rawalpindi, Pakistan

Tübingen
2024

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	07.05.2025
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Andreas Zell
2. Berichterstatter:	Prof. Dr. Hendrik Lensch

Dedication

This work is dedicated to my parents and my twin daughters, Fatima and Zainab — my greatest blessings.

Abstract

In this thesis, our primary objective is to reduce the computational footprint of state-of-the-art stereo deep neural network (DNN) methods while maintaining their performance. Classical stereo methods used in computer vision applications, such as robotics and autonomous driving, often require complex tuning and struggle to perform well in real-world scenarios. On the other hand, recent end-to-end DNN methods have shown superior performance but come with high computational requirements, making them unsuitable for real-time applications.

To achieve our objective, we pursue two complementary paths. Firstly, we optimize the individual components of state-of-the-art deep neural networks through a detailed empirical evaluation. This evaluation helps us identify the bottlenecks present in state-of-the-art stereo methods. Our findings reveal that the computational load primarily stems from the use of three-dimensional ($3\mathcal{D}$) convolutions in performance-oriented end-to-end stereo methods. Taking inspiration from the success of MobileNet blocks used for two-dimensional ($2\mathcal{D}$) convolutions, we propose a set of separable convolutions in the $3\mathcal{D}$ space. We thoroughly investigate the impact of making convolutions separable in different dimensions and demonstrate significant reductions in computational load without sacrificing performance. In fact, we observe performance improvements. Building on these conclusions, we design a family of networks based on $2\mathcal{D}$ and $3\mathcal{D}$ separable convolutions.

Furthermore, we explore the design of a leaner backbone for real-time stereo networks. We introduce a two-branch-based architecture that explicitly captures pixel-level and semantic-level information from the input images. This design choice results in a lean backbone that reduces computational load, albeit with a slight performance loss. To recover the lost performance, we propose to use learned attention weights based on cost volume combined with LogL1 loss for stereo matching.

In addition to optimizing individual components and modules, we investigate the application of knowledge distillation for designing leaner and faster stereo networks. Leveraging insights from stereo methods and general knowledge distillation techniques, we introduce a novel knowledge distillation pipeline. Through a systematic study of various design choices, we develop a leaner and faster stereo network with competitive performance. We emphasize the importance of carefully selecting distillation points and loss functions in distilling stereo networks, as they have a significant impact on performance. The trained student networks not only rival performance-oriented methods but also gives comparable results to speed-oriented stereo methods.

Overall, our thesis contributes to the development of computationally efficient and

high-performing stereo vision systems. By addressing the computational challenges of state-of-the-art stereo methods and leveraging knowledge distillation techniques, we facilitate the adoption of these methods for real-world systems and applications. We firmly believe that the findings and methodologies presented in this thesis advance the field of stereo vision and pave the way for more practical and effective depth estimation solutions.

Kurzfassung

In dieser Arbeit liegt unser Hauptziel darin, den Rechenaufwand von modernen Stereo-Deep-Neural-Network (DNN)-Methoden zu reduzieren, während wir ihre Leistung beibehalten. Klassische Stereo-Verfahren, die in Computer-Vision-Anwendungen wie Robotik und autonome Fahrzeuge verwendet werden, erfordern oft komplexe Abstimmungen und haben Schwierigkeiten, in realen Szenarien gut zu performen. Andererseits haben neuere End-to-End-DNN-Methoden eine überlegene Leistung gezeigt, erfordern jedoch eine hohe Rechenleistung, was sie für Echtzeit-Anwendungen ungeeignet macht.

Um unser Ziel zu erreichen, verfolgen wir zwei komplementäre Ansätze. Erstens optimieren wir die einzelnen Komponenten von modernen Deep-Neural-Networks durch eine detaillierte empirische Evaluierung. Diese Evaluierung hilft uns, die Engpässe in modernen Stereo-Methoden zu identifizieren. Unsere Ergebnisse zeigen, dass die Rechenlast hauptsächlich aus der Verwendung von dreidimensionalen ($3\mathcal{D}$) Faltungen in leistungsorientierten End-to-End-Stereo-Methoden resultiert. Inspiriert vom Erfolg der MobileNet-Blöcke für zweidimensionale ($2\mathcal{D}$) Faltungen schlagen wir eine Reihe von separablen Faltungen im $3\mathcal{D}$-Raum vor. Wir untersuchen gründlich die Auswirkungen der Separierbarkeit von Faltungen in verschiedenen Dimensionen und zeigen signifikante Reduzierungen der Rechenlast ohne Leistungseinbußen. Tatsächlich beobachten wir Verbesserungen der Leistung. Basierend auf diesen Erkenntnissen entwerfen wir eine Familie von Netzwerken, die auf separablen $2\mathcal{D}$- und $3\mathcal{D}$-Faltungen basieren.

Des Weiteren untersuchen wir den Entwurf eines schlankeren Backbones für Echtzeit-Stereo-Netzwerke. Wir stellen eine Zweigeteilte-Architektur vor, die explizit pixelgenaue und semantische Informationen aus den Eingangsbildern erfasst. Diese Designwahl führt zu einem schlankeren Backbone, der den Rechenaufwand reduziert, jedoch mit einem geringfügigen Leistungsverlust. Um die verlorene Leistung wiederherzustellen, schlagen wir vor, gelernte Aufmerksamkeitsgewichte basierend auf Kosten-Volumen in Kombination mit LogL1 loss zur Stereo-Matching zu verwenden.

Neben der Optimierung einzelner Komponenten und Module untersuchen wir auch den Einsatz von Wissensdestillation zur Gestaltung schlankerer und schnellerer Stereo-Netzwerke. Durch die Nutzung von Erkenntnissen aus Stereo-Verfahren und allgemeinen Wissensdestillationstechniken stellen wir eine neuartige Wissensdestillationspipeline vor. Durch eine systematische Untersuchung verschiedener Designentscheidungen entwickeln wir ein schlankeres und schnelleres Stereo-Netzwerk mit wettbewerbsfähiger Leistung. Wir betonen die Bedeutung der sorgfältigen Auswahl von destillationspunkten und Verlustfunktionen bei der Destillation von Stereo-Netzwerken, da sie einen erheblichen Einfluss auf die Leistung haben. Die trainierten Studentennetzwerke konkurrieren

nicht nur mit leistungsorientierten Methoden, sondern erzielen auch vergleichbare Ergebnisse zu geschwindigkeitsorientierten Stereomethoden.

Insgesamt trägt unsere Arbeit zur Entwicklung von recheneffizienten und leistungsstarken Stereo-Vision-Systemen bei. Indem wir uns mit den rechnerischen Herausforderungen moderner Stereo-Methoden auseinandersetzen und Wissensdestillationstechniken nutzen, erleichtern wir die Anwendung dieser Methoden in realen Systemen und Anwendungen. Wir sind fest davon überzeugt, dass die in dieser Arbeit vorgestellten Erkenntnisse und Methoden das Gebiet der Stereo-Vision voranbringen und den Weg für praktischere und effektivere Tiefenschätzungs-Lösungen ebnen.

Acknowledgments

Undertaking a thesis is a challenging endeavor, and this journey would not have been possible without the invaluable help and support of many individuals. Throughout this journey, I have been fortunate to receive tremendous assistance from numerous people.

First and foremost, I would like to express my deepest gratitude to my professor, Prof. Dr. Andreas Zell. His unwavering support and encouragement have been invaluable. I vividly remember a moment when I was concerned about my mother's health, and Prof. Zell generously offered me the time I needed to care for her. Similarly, during the COVID-19 pandemic, when I found myself isolated and struggling, he showed exceptional compassion by allowing me to be flexible. I am genuinely indebted to him for his unwavering support throughout my thesis.

Furthermore, I am profoundly grateful to my parents for their unwavering belief in my educational pursuits. In a society like Pakistan, their rare and commendable decision to encourage higher education has been an immense source of motivation for me. I owe them a lifetime of gratitude for their sacrifices, constant support, and faith in my abilities.

I am also deeply thankful to my siblings, whose love and encouragement have been a steady source of strength for me. They have always stood by me, offering reassurance during difficult times and celebrating my achievements with unreserved joy. I am forever grateful to them for their support and understanding.

Acknowledgment would be incomplete without mentioning my husband who has been with me every day throughout my thesis. He has been my rock and an incredible source of support, helping me cope with disappointments and low days, and, of course, sharing in the joy of successes as well. His presence and encouragement kept me grounded and motivated during the most challenging times. I am deeply thankful to him for always believing in me and for his support, which made this hard yet rewarding journey possible.

I would also like to acknowledge the support of my colleagues in the group, especially Martin, Leon, and others.

To all those whose names may not appear in this acknowledgment, your support has been no less significant. I am grateful for each and every person who has played a role in this remarkable journey. Thank you all for being part of this important milestone in my life.

Contents

Chapter 1

Introduction

1.1 Motivation

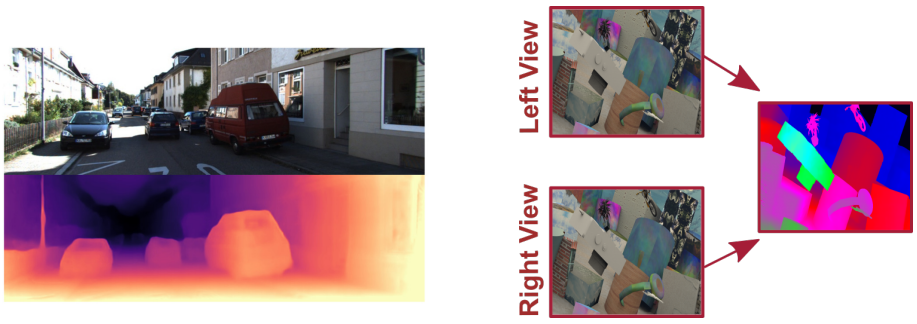
Depth perception is essential for humans and animals, as it plays a crucial role in their cognitive abilities and provides them with a fundamental understanding of the three-dimensional world. This ability empowers them to accurately perceive, navigate, interact with, and manipulate their surroundings. Humans, in particular, heavily rely on depth cues to interpret and comprehend their environment, highlighting the significance of precise depth estimation.

Depth perception is not only crucial for humans and animals but also holds great importance for robots and autonomous systems. Just like humans, robots and autonomous systems require an accurate understanding of the three-dimensional world. Robots equipped with depth perception can better estimate distances to objects, determine object shapes and sizes, and assess the layout of the environment. This information is vital for obstacle avoidance, path planning, and object manipulation tasks. For example, in autonomous vehicles, depth perception allows for precise detection and tracking of other vehicles, pedestrians, and road structures, enhancing safety and enabling reliable decision-making. Moreover, depth perception has crucial applications in fields such as $3\mathcal{D}$ modeling, object recognition, medical imaging, augmented reality, and more, enabling accurate representation, recognition, and interaction with the three-dimensional world.

Thus, given the vast set of applications of depth estimation, it comes as no surprise that the field of depth estimation and its applications remains highly active. Generally, depth estimation techniques can be categorized into two main categories, based on the method used for depth estimation: active and passive sensing.

To estimate depth, active sensing methods project a signal or pattern onto the scene and measure its response, such as the time it takes for the signal to return or the deformation of the projected pattern. Examples of active sensing include: Time-Of-Flight (ToF) Cameras, Structured Light Projection devices, Laser Imaging Detection and Ranging (LIDAR), *etc*. Active sensing methods often provide more accurate depth information and can work in various lighting conditions. However, they require specialized sensors or projectors, are costly and have inherent limitations.

On the other hand, passive depth estimation techniques utilize the available light in



Mono depth estimation Stereo

Figure 1.1: Examples of passive depth sensing techniques: Monocular depth estimation (Godard *et al.*, 2019) (left) and stereo depth estimation (right) (Images Source: SceneFlow dataset (Mayer *et al.*, 2016)).

the environment to capture images using regular cameras and infer the depth of the scene. These techniques include mono depth estimation, structure from motion, and stereo depth estimation – Fig. 1.1. In mono depth estimation, depth information is estimated using a $2\mathcal{D}$ image captured with a single camera. It is a cost-effective as it uses only single camera and computationally efficient approach. However, the accuracy of depth estimation is significantly affected in low-texture surfaces and distant objects. Structure from motion (SfM) infers depth information by analyzing a series of images of moving objects or scenes to capture the motion and relative positions of the objects. SfM is a computationally expensive technique as it requires incorporating multiple images to infer depth information, making it challenging for large-scale applications. Stereo depth estimation, on the other hand, involves inferring depth using a pair of images and provides a good compromise between mono depth estimation and SfM. This technique is not as computationally demanding as SfM and leverages additional information from the second image to achieve better depth estimation compared to mono depth estimation. Moreover, depth estimation using two images closely mimics the natural phenomenon where most species, including humans, use two eyes to perceive the $3\mathcal{D}$ structure of the world.

Passive depth estimation methods offer more flexibility in terms of sensor requirements, but they can be challenged by low-light conditions, severe weather environments, and complex scenes. To achieve a universal depth estimation method, modern systems often combine active and passive sensing techniques, resulting in more robust and accurate results. Overall, among these techniques, passive stereo vision remains a favorable choice due to its balance between cost, complexity, and accuracy. Thus, in this thesis our primary focus is on designing and building efficient stereo depth estimation techniques.

Stereo methods can be broadly classified into traditional and modern approaches. Traditional stereo methods involve matching corresponding image points in a pair of stereo images to calculate disparity. Popular examples of traditional stereo methods include

Block Matching (Tao *et al.*, 2008), Semi-Global Matching (SGM) (Hirschmuller, 2005), Graph Cuts (Kolmogorov and Zabih, 2001), and Dynamic Programming (Scharstein and Szeliski, 2002), *etc*. These traditional stereo methods, while widely used and foundational in nature, have certain limitations. For instance, they often struggle with handling occlusions, textureless regions, and complex scenes. Furthermore, the reliance on handcrafted features and traditional matching algorithms can result in inaccurate depth estimations and limited robustness.

In recent years, there has been a significant shift towards modern methods that leverage neural networks and deep learning techniques for stereo depth estimation. This shift has been driven by the ability of neural networks to learn and extract features automatically, capturing more complex patterns and relationships in the stereo images. Modern methods (Mayer *et al.*, 2016; Kendall *et al.*, 2017; Chang and Chen, 2018; Guo *et al.*, 2019a) based on neural networks have demonstrated improved performance in handling challenging scenarios, achieving higher accuracy, and overcoming the limitations of classical methods.

Deep neural networks, despite being adopted relatively late for stereo vision, have played a crucial role in significantly improving the overall performance of stereo methods, leading to notable advancements in stereo vision. However, they do present challenges in terms of large model size (measured by the number of parameters) and high computational requirements (measured by the number of Floating-Point Operations - FLOPs). These limitations hinder the practical deployment of deep stereo networks in resource-constrained scenarios, such as edge devices or real-time systems. Therefore, there is a pressing need to optimize stereo methods to make them more lightweight and efficient, without compromising their performance.

In this thesis, the primary focus is on optimizing state-of-the-art end-to-end deep neural network-based stereo methods by reducing their size and computational requirements while preserving their performance. The goal is to develop lightweight and efficient stereo estimation networks suitable for real-world applications.

This research explores two complementary approaches to achieve this objective. Firstly, the thesis investigates methods to optimize individual network operators, layers, and modules. This involves analyzing and refining these components to decrease their computational demands and memory footprint, while ensuring accuracy is not compromised. Techniques such as $2\mathcal{D}$ and $3\mathcal{D}$ separable convolutions (Sandler *et al.*, 2018), efficient two-branch backbones (Yu *et al.*, 2021), and other architectural modifications are explored and proposed to achieve this optimization.

Secondly, the thesis explores the use of knowledge distillation to construct end-to-end efficient student networks. Knowledge distillation involves transferring knowledge from a larger, more complex teacher network to a smaller, more compact student network. By distilling the crucial information and performance of the teacher network, the aim is to create a more efficient student network that can achieve competitive results in stereo estimation.

By investigating both the optimization of individual network components and the uti-

lization of knowledge distillation, novel approaches are developed to create lightweight and efficient stereo estimation networks. The thesis includes comprehensive analysis and experimentation to validate the effectiveness of the proposed optimizations and techniques. Overall, this research contributes to the advancement of end-to-end deep neural network-based stereo methods by addressing the challenges of size and computational requirements. By enabling practical implementations in real-world applications, this work aims to bridge the gap between high-performance stereo estimation and resource-constrained environments.

1.2 Outline and Contributions

This thesis contributes the following peer reviewed papers to the computer vision community, where * indicates equal contribution:

1. Rafia Rahim, Faranak Shamsafar, and Andreas Zell. **"Separable Convolutions for Optimizing 3D Stereo Networks"**. In IEEE International Conference on Image Processing (ICIP), September 2021.

   In this work, we conduct a thorough profiling analysis of stereo networks, revealing that three-dimensional ($3\mathcal{D}$) convolutions used in these networks contribute significantly to the overall computational load, accounting for up to 94% of the operations. To address this bottleneck, we propose a novel approach based on three-dimensional separable convolutions, which effectively reduce the number of parameters and operations required. By incorporating these separable convolutions into state-of-the-art stereo networks, we achieve promising results. Our approach leads to a reduction of up to $7\times$ in the number of operations and up to $3.5\times$ in the number of parameters, while maintaining the performance of the networks. This demonstrates the effectiveness of our proposed approach in enhancing the efficiency of stereo networks without compromising their performance.

2. Faranak Shamsafar*, Samuel Woerz*, Rafia Rahim, and Andreas Zell. **"MobileStereoNet: Towards Lightweight Deep Networks for Stereo Matching"**, In IEEE Winter Conference on Applications of Computer Vision (WACV), January 2022.

   In this work, we propose lightweight depth estimation models for stereo vision by adapting the MobileNet architecture. We extend MobileNet blocks to create efficient $2\mathcal{D}$ and $3\mathcal{D}$ models for stereo vision. Additionally, we introduce a novel cost volume approach to improve the accuracy of the $2\mathcal{D}$ model, bringing it closer to the performance of $3\mathcal{D}$ networks. Compared to the baseline $2\mathcal{D}$ and $3\mathcal{D}$ models, our proposed methods achieve a significant reduction of 27% (parameters) and

95% (operations) for $2\mathcal{D}$, and 72% (parameters) and 38% (operations) for $3\mathcal{D}$, while maintaining performance.

3. Rafia Rahim, Samuel Woerz, and Andreas Zell. **"LeanStereo: A Leaner Backbone based Stereo Network"**. In International Joint Conference on Neural Networks (IJCNN), June 2023.

   In this work, we present a fast end-to-end stereo matching method that achieves significant speedup by incorporating a leaner backbone. To compensate for the performance loss caused by the leaner backbone, we propose a learned attention weights-based cost volume combined with LogL1 loss for stereo matching. The inclusion of LogL1 loss not only improves the overall performance of our network but also enables faster convergence. Through a detailed empirical evaluation and comparison with state-of-the-art methods such as ACVNet (Xu *et al.*, 2022), LEAStereo (Cheng *et al.*, 2020) and CFNet (Shen *et al.*, 2021), we demonstrate that our optimized method requires $4\times$ fewer operations and is 9 to $14\times$ faster while delivering comparable performance.

4. Rafia Rahim, Samuel Woerz, and Andreas Zell. **"Distilling Stereo Networks for Performant and Efficient Leaner Networks"**. In International Joint Conference on Neural Networks (IJCNN), June 2023.

   Knowledge distillation has been widely used for tasks like classification and segmentation, but its application in distilling state-of-the-art stereo matching methods has been relatively limited. The complexity of stereo matching networks, which consist of multiple two- and three-dimensional modules, has hindered the use of knowledge distillation in this context.
   In this work, we address this gap by proposing a joint framework that combines insights from state-of-the-art stereo methods with general knowledge distillation techniques. Our approach enables the distillation of stereo networks while achieving competitive results and faster inference. We emphasize the importance of carefully designing the complete distillation pipeline, including the selection of the backbone, distillation points, and corresponding loss functions. Through extensive empirical analysis, we demonstrate that our distilled student networks are leaner and faster while maintaining excellent performance. For example, our student network outperforms performance-oriented methods like PSMNet (Chang and Chen, 2018), CFNet (Shen *et al.*, 2021), and LEAStereo (Cheng *et al.*, 2020) on benchmark SceneFlow dataset, while being $8\times$, $5\times$, and $8\times$ faster, respectively. Moreover, compared to speed-oriented methods with inference times less than 100*ms*, our optimized student networks achieve comparable performance.

The thesis outline is as follows:

Chapter 2 In this chapter, we provide an in-depth introduction to the problem of stereo depth estimation. We begin by discussing the significance of epipolar geometry, which plays a crucial role in constraining the search space during stereo matching. Subsequently, we explore rectification and triangulation, essential techniques employed to improve the accuracy of stereo depth estimation. Later, we discuss traditional and deep neural network-based stereo matching algorithms. Finally, we present well-known stereo matching datasets that are used for benchmarking and comparison in this thesis.

Chapter 3 In this chapter, we present our paper titled "Separable Convolutions for Optimizing Stereo Networks" by Rahim *et al* (2021). We begin by conducting a detailed profiling of $3\mathcal{D}$ stereo networks to identify the bottlenecks that hinder their performance. Building upon this analysis, we introduce our approach which leverages a set of separable $3\mathcal{D}$ convolutions specifically designed to address these bottlenecks and optimize the encoder-decoder component of end-to-end stereo networks. Finally, we provide a comprehensive comparative analysis to demonstrate the effectiveness of our proposed separable convolutions in reducing both the network size and computational requirements of these networks.

Chapter 4 This chapter highlights our work titled "MobileStereoNet: Towards Lightweight Deep Networks for Stereo Matching" by Shamsafar et al. (2022). Specifically we provides details on how we utilize the lightweight MobileNet blocks (Howard *et al.*, 2017) to reduce the computational requirements in the backbone and encoder-decoder modules of stereo networks. Furthermore, we introduce a novel approach for constructing the cost volume, tailored to ensure optimal performance of the MobileStereoNet models while maintaining their lightweight nature.

Chapter 5 This chapter provides detailed information about our work titled "LeanStereo: A Leaner Backbone-based Stereo Network" by Rahim *et al* (2023). We present our motivation and efforts in designing a lighter backbone for end-to-end stereo networks. Furthermore, we investigate the impacts of incorporating the LogL1 loss function and attention-based cost volume techniques to mitigate the performance loss resulting from integrating the leaner backbone in $3\mathcal{D}$ stereo networks. We conclude with a comprehensive empirical analysis and comparisons on benchmark datasets.

Chapter 6 In this chapter, we present our paper titled "Distilling Stereo Networks for Performant and Efficient Leaner Networks" by Rahim *et al* (2023). We discuss our approach of leveraging knowledge distillation techniques to design leaner $3\mathcal{D}$ stereo methods. We provide a detailed discussion of the various design choices we explored in developing lightweight stereo networks, referred

to as DSNet (s), through the use of knowledge distillation. These networks are specifically designed to be efficient and computationally lean. To validate our approach, we conduct a comprehensive empirical analysis, evaluating the design choices and measuring the performance of our distilled network(s) against other state-of-the-art methods on benchmark datasets.

Chapter 7 In this concluding chapter, we provide a summary of the key findings and contributions from the dissertation. We highlight the main conclusions derived from our work and discuss their significance in the broader context of the field. Additionally, we outline potential future directions for further exploration and development.

Chapter 2

Foundations

Humans possess a remarkable ability to perceive the three-dimensional ($3\mathcal{D}$) structure of a scene even from a single image. However, accurately estimating the $3\mathcal{D}$ position of objects algorithmically from a single image poses a challenge due to the loss of a dimension in the projection process. This is known as single-image depth estimation, and it is considered an ill-posed problem, because multiple three-dimensional surfaces can produce the same two-dimensional image. In contrast, stereo vision offers a solution by utilizing a pair of images captured from slightly different viewpoints to infer the $3\mathcal{D}$ position of objects.

Stereo vision is inspired by the natural systems found in many species, including humans, where binocular stereopsis is utilized for depth perception. Humans, for instance, have two eyes, each observing a slightly different view of the world. These two views are then fused by our brain to form a cohesive and accurate perception of the scene in three dimensions.

In this chapter, we provide a concise overview of the steps involved in algorithmically estimating depth from a pair of input images. We begin by discussing the generic steps involved and subsequently delve into the detailed process of stereo matching. Specifically, we describe the evolution of the field from traditional methods to deep learning-based stereo matching approaches. Finally, we conclude the chapter by examining the benchmarking datasets and metrics employed to quantify the effectiveness of stereo matching methods.

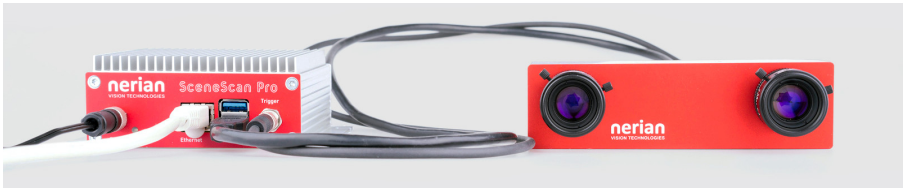


Figure 2.1: A stereo system comprising two cameras and a processing unit. `https://nerian.de/produkte/`

2.1 Stereo Vision

In stereo vision, a pair of cameras (see Fig. 2.1) is used to capture a scene from different viewpoints. The goal is to estimate the depth of the scene by analyzing the positional differences of corresponding objects or pixels in both camera views. These positional differences, called "*disparity*," occur between the same pixels in the left and right views. By *triangulating* these disparities, depth information can be obtained.

Figure 2.2: Steps involved in depth estimation using stereo vision.

The stereo vision pipeline, as depicted in Fig. 2.2, consists of several key steps:

- **Camera calibration:** This initial step involves determining the intrinsic parameters (e.g., focal length) and extrinsic parameters (e.g., baseline, relative rotation, and translation) of the camera pair. Calibration helps establish a precise relationship between the cameras and the captured images.
- **Rectification:** To simplify subsequent processing, the left and right images are rectified. This transformation aligns the images, ensuring that corresponding epipolar lines are parallel and appear on the same scanline in both views.
- **Stereo matching (disparity computation):** The stereo matching step involves finding corresponding points or features between the rectified left and right images. Disparity refers to the positional difference of these corresponding points, which is calculated by identifying pixel disparities along the epipolar lines.
- **Depth estimation:** Using the computed disparities, depth estimation is performed through triangulation. This process enables the creation of a depth map or a three-dimensional representation of the scene.

Each of these steps contributes to the overall stereo vision pipeline, allowing for accurate depth estimation and three-dimensional understanding of the captured scene.

This thesis primarily focuses on the *stereo matching* step, which is considered the most important and complex task within the overall pipeline of stereo vision. However, it is important to note that we also acknowledge the prerequisite that the cameras are calibrated and the images are rectified before performing stereo matching. These calibration and rectification steps are typically addressed by various publicly available stereo matching datasets (*c.f.* Sec. 2.5).

Nonetheless, we believe it is essential to thoroughly explain the process of image rectification and highlight how it simplifies the subsequent stereo matching process. By exploring the concepts and techniques related to image rectification, we aim to establish a solid foundation for the subsequent discussions and optimizations of the stereo matching step in this thesis.

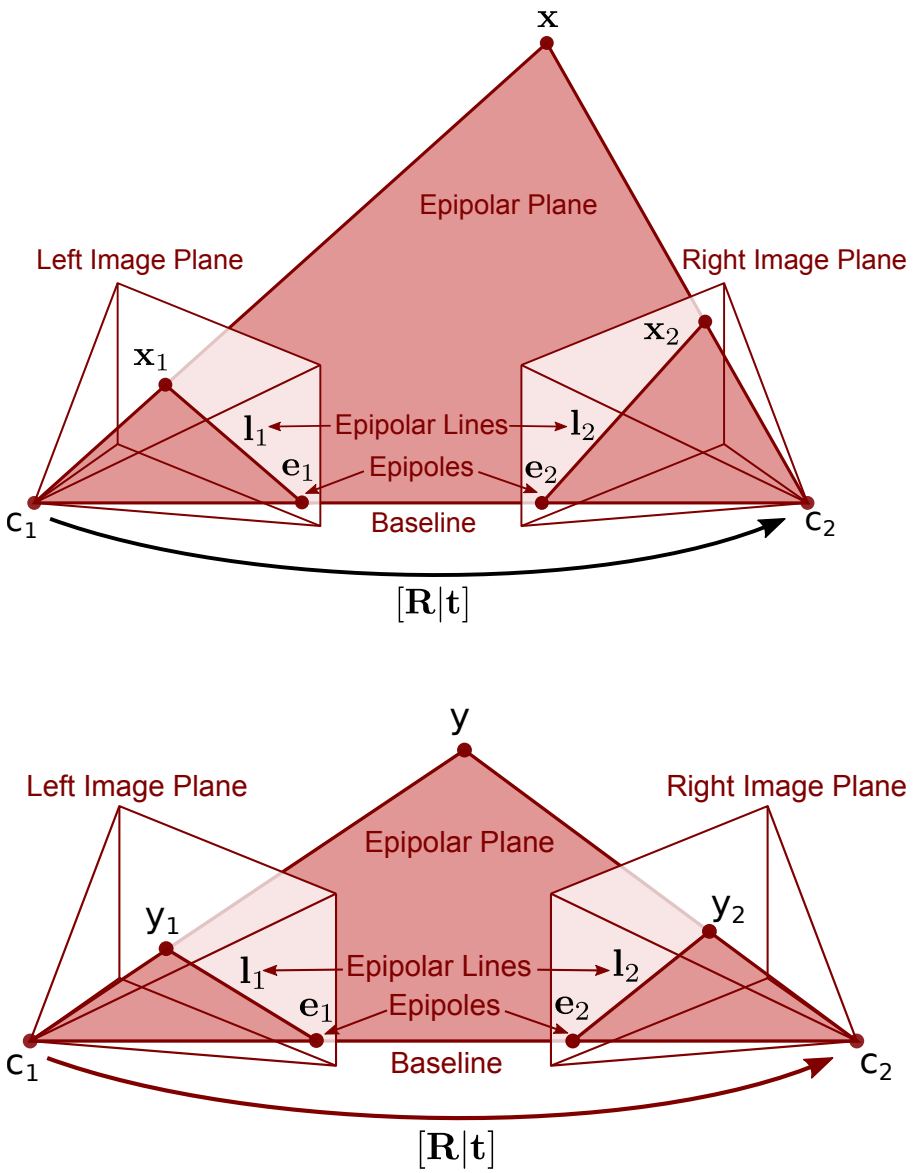


Figure 2.3: Example of stereo geometry for two different $3\mathcal{D}$ points. It is worth noting that even though points $\mathbf{x}$ and $\mathbf{y}$ are distinct and mapped to different locations on the image planes, for both of them the epipolar lines ($\mathbf{l}_1$ and $\mathbf{l}_2$) pass through the same epipolar points *i.e.* $\mathbf{e}_1$ and $\mathbf{e}_2$. Adapted and redrawn from Szeliski (2022).

2.2 Stereo Rectification

In stereo matching, the primary challenge is to determine the corresponding pixels between the left and right views. The question that arises is: How can we efficiently find these corresponding pixels? For instance, when we have a single pixel (x, y) in the left

view, should we search through all the pixels in the right view to find its match? The answer is no.

To overcome this challenge, we can utilize the concept of epipolar geometry, which enables us to narrow down the search space and efficiently determine the corresponding pixels. Epipolar geometry establishes a geometric relationship between the two views, providing constraints on the possible locations of corresponding pixels.

By applying the principles of epipolar geometry, we can rectify the images, transforming them such that corresponding pixels lie on the same horizontal lines. This rectification process simplifies the search for corresponding pixels, as we only need to search along the horizontal direction instead of the entire image. This reduction in search complexity from a two-dimensional search to a one-dimensional search greatly enhances the efficiency of the stereo matching process.

2.2.1 Epipolar Geometry

Consider the setup depicted in Fig. 2.3 (top), where two cameras capture a $3\mathcal{D}$ point, denoted as $\mathbf{x}$, from different views. The centers of the left and right cameras are labeled as $\mathbf{c}_1$ and $\mathbf{c}_2$, respectively. The $3\mathcal{D}$ point $\mathbf{x}$ is projected onto the left image plane as $\mathbf{x}_1$ and onto the right image plane as $\mathbf{x}_2$. The line connecting the camera centers is known as the *baseline*, which intersects the image planes at points $\mathbf{e}_1$ and $\mathbf{e}_2$, referred to as *epipoles*. Additionally, the *epipolar plane*, formed by points $\mathbf{x}$, $\mathbf{c}_1$, and $\mathbf{c}_2$, also intersects the image planes at points $\mathbf{x}_1$ and $\mathbf{x}_2$. The *epipolar line* $\mathbf{l}_1$, connects the point $\mathbf{x}_1$ to the epipole $\mathbf{e}_1$ on the left image plane. Similarly, $\mathbf{l}_2$ represents an epipolar line on the right image plane, connecting points $\mathbf{x}_2$ and $\mathbf{e}_2$. The extrinsic parameters of the cameras, denoted as $\mathbf{R}$ and $\mathbf{t}$, indicate their relative rotation and translation, respectively.

For a given point $\mathbf{x}_1$ in the left image, its corresponding point in the right image lies on the epipolar line $\mathbf{l}_2$. This implies that searching the entire right image is unnecessary. Instead, we can limit our search to a single line. Similarly, for the point $\mathbf{x}_2$ in the right image, its corresponding point lies on the epipolar line $\mathbf{l}_1$ in the left image. Therefore, by knowing the epipolar plane and the intrinsic and extrinsic parameters of the cameras such as the focal length $\mathbf{f}$, baseline $\mathbf{b}$, relative rotation $\mathbf{R}$, and translation $\mathbf{t}$ of the cameras, we can efficiently search along a one-dimensional line to find the corresponding points in an image pair.

In Fig. 2.3 (bottom), another example is presented where a different $3\mathcal{D}$ point $\mathbf{y}$ is projected onto the left and right image planes as $\mathbf{y}_1$ and $\mathbf{y}_2$, respectively. It is worth noting that although this point is projected to different locations on the image planes compared to Fig. 2.3 (top), the epipoles remain unchanged. This observation illustrates that, for a given image plane, all epipolar lines intersect at a common point known as *epipole*.

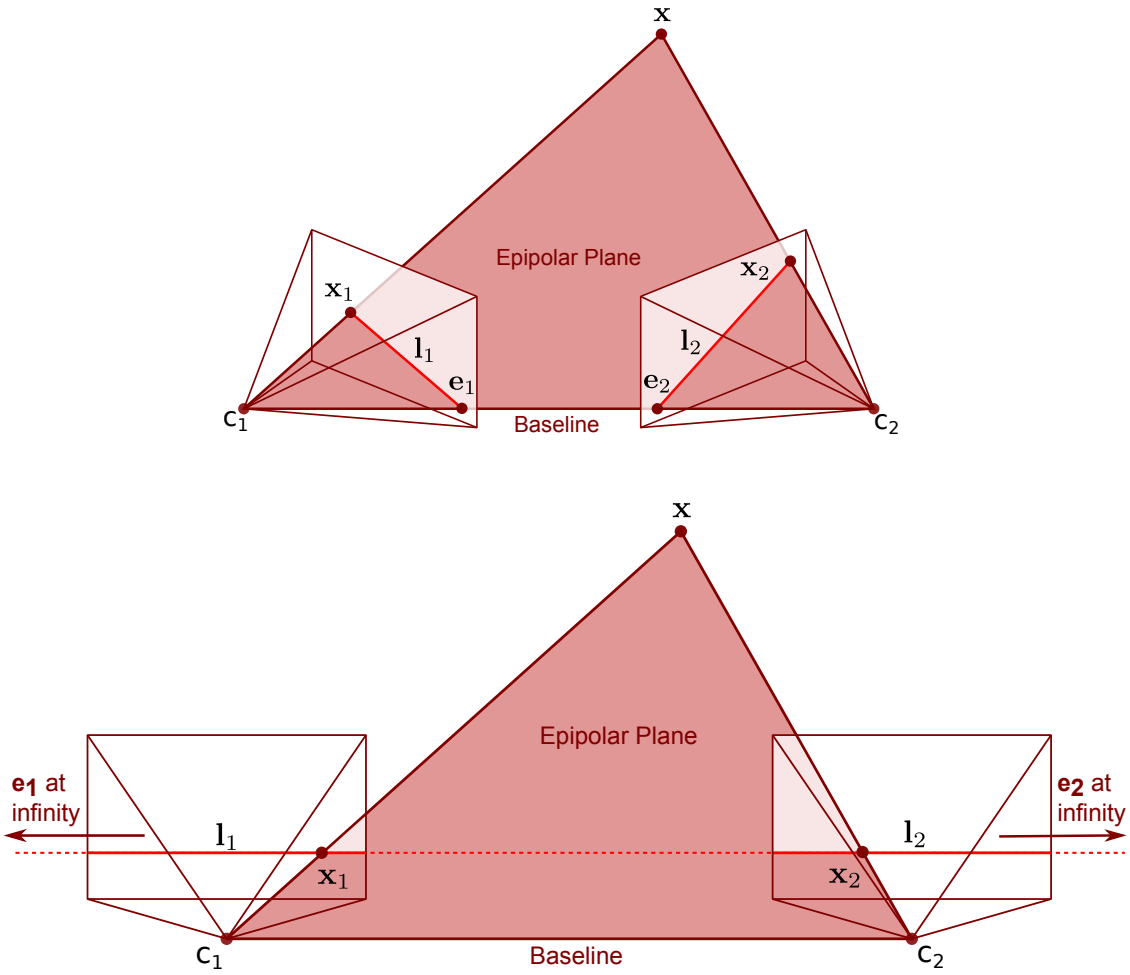


Figure 2.4: Comparison of epipoles ($\mathbf{e}_1$ and $\mathbf{e}_2$) and search lines ($\mathbf{l}_1$ and $\mathbf{l}_2$) when the cameras are not facing the same direction (top) and when they are facing the same direction (bottom). Adapted and redrawn from Szeliski (2022)

Coplanar Image Pairs

In Fig. 2.3, it can be observed that both cameras are not facing the same direction. Now, let's consider what would happen if both cameras were facing the same direction. In such a scenario:

- The captured images from both cameras would lie in the same plane and become parallel to the baseline.
- Consequently, the epipoles, which are the points where the baseline intersects the image planes, would move outside the image domain, approaching infinity.
- The epipolar lines, which indicate the correspondence between the images, would

become horizontal.

To illustrate this, refer to Fig. 2.4 (top). In this case, the epipoles lie within the image plane since the cameras are not facing the same direction and the image planes are not coplanar. However, in Fig. 2.4 (bottom), both epipoles have moved outside the image domain, the baseline is parallel to the image plane, and both epipolar lines are horizontal to the image planes.

Fig. 2.5 presents a practical example where both the left and right images are captured using cameras facing the same direction. In this case, the marked points (indicated by red circles) in the left image have corresponding points along the horizontal lines marked on the right image. This simplifies the process of finding corresponding points and is a commonly employed technique in modern stereo cameras. An illustration of such a stereo system is depicted in Fig. 2.1, where both cameras face the same direction and are positioned at a specific distance (baseline **b**) from each other.

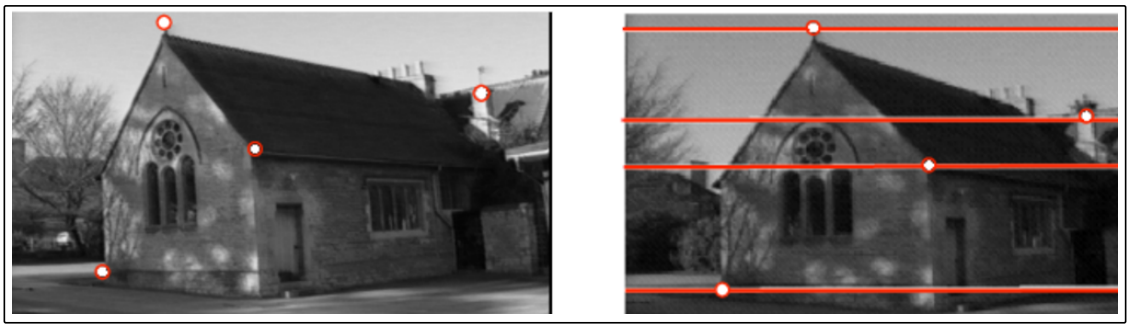


Figure 2.5: Example of rectified stereo pair images. Correspondence for each red point in the left image (shown as a red circle) is found by searching along the corresponding red horizontal line in the right image (Hartley and Zisserman, 2004).

In an ideal scenario, both cameras in a stereo system should be calibrated in such a way that we only need to search along the horizontal line for corresponding points. However, in real-life situations, cameras may not be perfectly calibrated. Therefore, it becomes necessary to explicitly reproject the image planes onto a common plane to facilitate correspondence search along the horizontal direction only. This overall process is known as *rectification*. Fig. 2.6 illustrates an example where a stereo pair was initially not coplanar (top) and then made coplanar after rectification (bottom). After rectification, it can be observed that the correspondence lines become horizontal with respect to the image planes.

2.3 Stereo Matching

Once the images have been rectified, the next step in stereo depth estimation is to perform stereo matching, which involves finding the positional displacement between pixels. Specifically, given a pixel p_l in the left image, the goal is to determine its horizontal

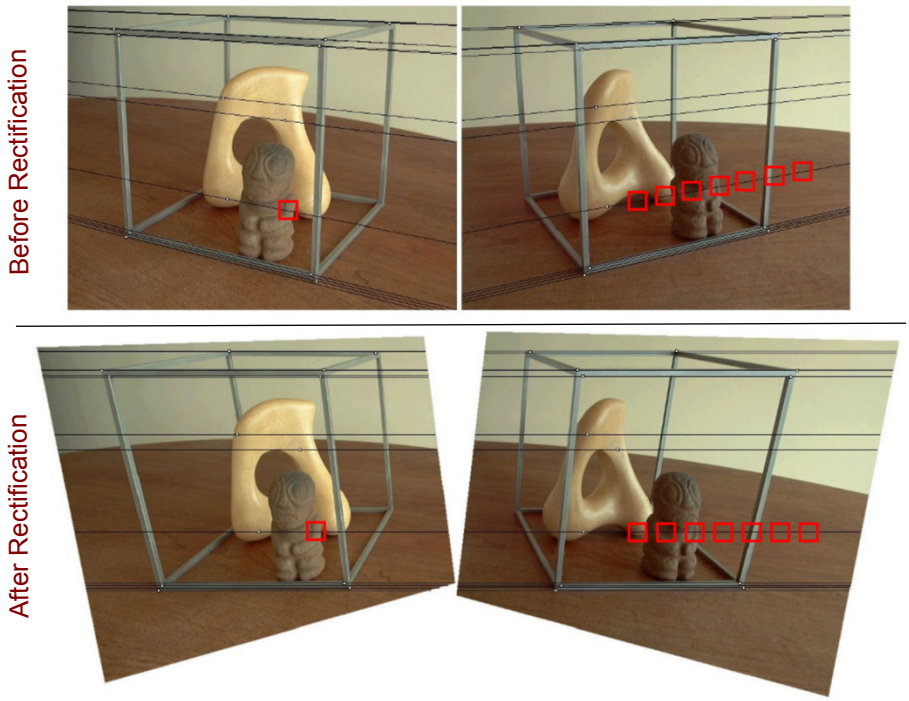


Figure 2.6: Stereo pair before and after rectification. In the top image, prior to rectification, the epipolar lines were not horizontal, requiring the search for corresponding points in both vertical and horizontal directions. However, in the bottom image, after rectification, the search line becomes purely horizontal. (Image Source: Loop and Zhang (1999))

displacement d in the right image. Stereo matching has been an actively researched topic in the computer vision community.

Modern stereo matching algorithms primarily rely on end-to-end deep neural networks. It is important to note that before the breakthrough of neural networks, stereo algorithms heavily depended on handcrafted features and optimization techniques, similar to other computer vision tasks. In the following sections, we will discuss in detail the building blocks of both traditional and modern stereo methods, highlighting their key components and steps involved.

2.3.1 Traditional Methods

Traditional stereo methods typically involve four main steps (Scharstein and Szeliski, 2002) (*c.f.* Fig. 2.7):

1. **Matching cost computation:** In the *matching cost computation* step, the goal is to compare the features of the left and right images in order to find the corresponding pixels. One common approach is to consider a patch (a small window) centered

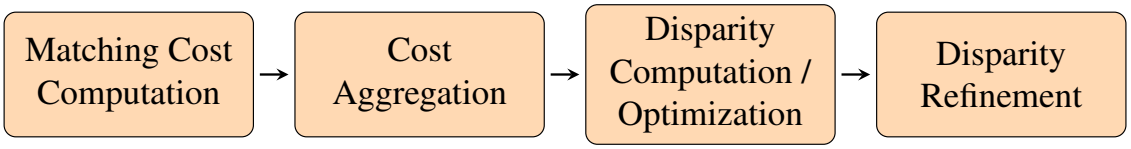


Figure 2.7: Steps used by traditional stereo matching methods.

around a pixel in the left image and compute its similarity to all the patches along the epipolar line (horizontal line) in the right image, as shown in Fig. 2.8.
Different matching criteria can be used to measure the similarity between patches. Traditional criteria include the absolute intensity difference (mean absolute error) (Anandan, 1989; Matthies *et al.*, 1989) and the squared intensity difference (mean-squared error) (Kanade *et al.*, 1995). However, these criteria are sensitive to radiometric differences between the images, which can affect matching accuracy.
To address this issue, more robust matching methods have been introduced. Techniques proposed by Hirschmuller (2007); Klaus *et al.* (2006) and Viola and Wells III (1997) aim to improve matching performance by considering factors beyond simple intensity differences. These methods incorporate additional information, such as local image gradients, texture patterns, or statistical measures, to enhance the matching process and handle challenging scenarios, such as occlusions and textureless regions. By using more sophisticated matching criteria, the accuracy and robustness of traditional stereo methods can be improved, albeit at extra computational cost.

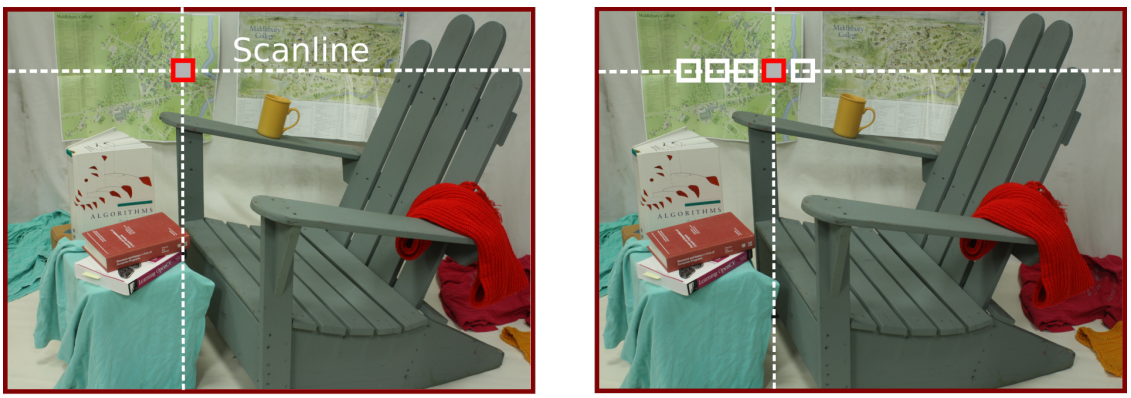


Figure 2.8: A visual representation of the feature matching process. Features from the left image are matched along the epipolar line in the right image to identify the corresponding patch. Images from Middlebury dataset (Scharstein *et al.*, 2014) are used to draw this figure.

2. **Cost aggregation:** In the *cost aggregation* step, the matching costs obtained in

Step 1 are accumulated over neighboring pixels. This is achieved by summing or averaging the costs over a small window. Aggregating the costs of neighboring pixels helps to reduce errors that may arise when considering the cost of a single pixel alone. Most methods calculate the cost of a pixel by treating all neighboring pixels equally, using a fixed-size window (Klaus *et al.*, 2006; Bleyer and Gelautz, 2005; Egnal, 2000). However, some methods (Yoon and Kweon, 2006; Yang *et al.*, 2008) assign different weights to the neighbors based on their color similarities and proximity to the central pixel. By considering color similarities and spatial proximity, these methods aim to improve the accuracy of cost aggregation.

3. **Disparity computation / optimization:** In local methods, the *disparity computation* step involves calculating the disparity at each pixel by selecting the disparity with the lowest aggregation cost among all the aggregated matching costs along the epipolar line. This approach is used in methods such as those proposed by Egnal (2000); Hirschmüller *et al.* (2002) and Yoon and Kweon (2006). On the other hand, global methods (Klaus *et al.*, 2006; Bleyer and Gelautz, 2005; Sun *et al.*, 2005) take a different approach. They typically skip the cost aggregation step (step 2) and instead define a global energy function. The primary focus of global methods is on the disparity computation step, where the objective is to find the disparity function that minimizes the overall energy function. This energy function consists of a data term that incorporates the pixelwise matching cost and a smoothness term that promotes smoothness in the selected disparities (Hirschmuller, 2007). Global methods allocate a significant portion of their computations to this disparity computation step.

4. **Disparity refinement:** In the *disparity refinement* step, the estimated disparities are improved through various techniques. This may involve removing disparity peaks, performing consistency checks to ensure the disparities are consistent across the left and right views, or interpolating gaps in the disparity map (Hirschmüller *et al.*, 2002; Yang *et al.*, 2008; Hirschmuller, 2003).

Traditional stereo methods, despite their widespread adoption, are not without limitations that affect their performance and adaptability. These methods face challenges in handling occlusions, textureless regions, and complex scenes, where accurate depth estimation becomes problematic. Furthermore, their reliance on handcrafted features and matching algorithms can lead to inaccuracies in depth estimation and limited robustness in various scenarios. Additionally, they are sensitive to parameter tuning and have difficulty handling large disparities and non-ideal imaging setups. The computational complexity of some traditional methods also poses challenges for real-time applications.

2.3.2 Modern Methods

Modern methods (Guo *et al.*, 2019b; Xu *et al.*, 2022; Tankovich *et al.*, 2021) are utilizing advancements in machine learning, specifically deep learning, to address the limitations of traditional stereo methods and improve their accuracy. By leveraging deep neural networks, these methods can learn intricate representations from data, enabling them to handle challenging scenarios such as occlusions, textureless regions, and complex scenes more effectively. The use of deep learning techniques allows for improved feature extraction and modeling, leading to more accurate and reliable stereo depth estimation. Ini-

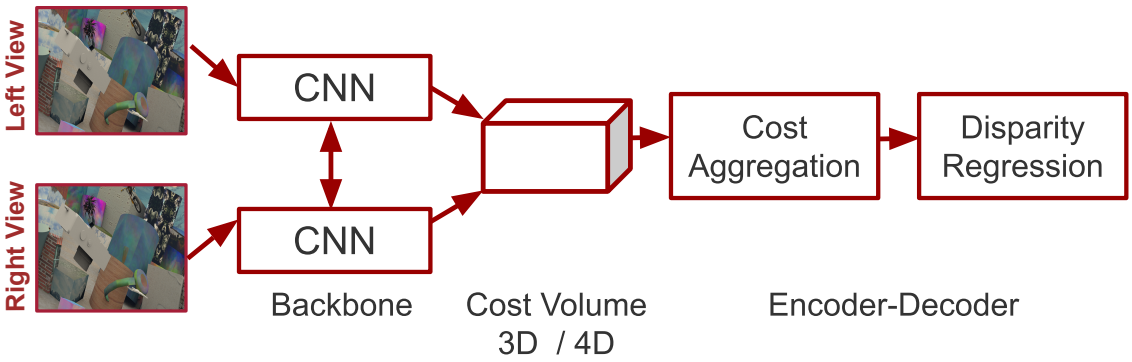


Figure 2.9: Main modules of end-to-end deep learning based stereo matching pipeline.

tially, attempts were made to train neural networks to learn the individual steps involved in the stereo matching pipeline (Luo *et al.*, 2016b; Batsos *et al.*, 2018; Schonberger *et al.*, 2018; Shaked and Wolf, 2017). Although these methods demonstrated the potential of using neural networks to solve the stereo matching problem, they were complex, difficult to train, and challenging to integrate into the entire matching pipeline.

However, the introduction of DispNet-C (Mayer *et al.*, 2016) marked a new era in stereo matching by introducing the first end-to-end learning-based stereo method. This breakthrough paved the way for the development of more advanced end-to-end deep learning-based stereo matching methods. The generic pipeline shown in Fig. 2.9 comprises different modules involved in end-to-end stereo methods, with each module representing the following:

- Backbone: This module takes a pair of input images and employs a shared backbone to extract visual features. It operates on $2\mathcal{D}$ inputs using $2\mathcal{D}$-dimensional convolution filters, effectively making it a $2\mathcal{D}$ CNN. The majority of recent methods, including (Mayer *et al.*, 2016; Gu *et al.*, 2020; Guo *et al.*, 2019a; Xu *et al.*, 2022), adopt an encoder-decoder style network architecture. This design allows the network to learn features by initially reducing the input resolution and subsequently upsampling it to a certain factor, typically $2\times$, of the original image resolution.

- Cost Volume: This module takes the features extracted from the left and right images by the shared backbone and combines them to form a volume known as the cost volume. The dimension of the cost volume can vary depending on the specific network, with options including $3\mathcal{D}$ or $4\mathcal{D}$ volumes.

- Cost-Aggregation: This module takes the cost volume as input and applies a series of convolutional, downsampling, and upsampling layers. Its purpose is to aggregate costs. The architecture of this module depends on the dimensionality of the cost volume used as input and can be either a $2\mathcal{D}$ CNN or a $3\mathcal{D}$ CNN, where the later uses $3\mathcal{D}$ convolutional kernels.

- Disparity Regression: This module takes inputs from multiple layers in the cost aggregation modules and combines them to generate output disparity maps. For example, in the case of a $3\mathcal{D}$ CNN, it often (Guo *et al.*, 2019a; Xu *et al.*, 2022) utilizes point-wise $3\mathcal{D}$ convolutions to convert $4\mathcal{D}$ inputs into $3\mathcal{D}$ outputs. These outputs are then further processed to regress a $2\mathcal{D}$ disparity map

As stated earlier, almost all recent deep learning-based methods use a similar architecture in the backbone module. The main differences among these methods arise from the approach used to combine the left and right features for constructing the cost volume and how this cost volume is subsequently utilized to regress the disparities. In this context, we explore two popular techniques for cost volume construction.

Correlation based $3\mathcal{D}$ Cost Volume:

The correlation-based cost volume was initially introduced by DispNet-C (Mayer et al., 2016), and it has since been adopted and utilized by many other methods. In this method, features from the left image ($\mathbf{f}_l$) and the right image ($\mathbf{f}_r$) are combined successively using a correlation operator to generate a cost volume with D disparity levels (see Figure 2.10). The computation is performed as follows:

$$\mathbf{CV}_{\text{correlate}}(d,x,y) = \langle \mathbf{f}_l(x,y), \mathbf{f}_r(x-d,y) \rangle \quad \text{for all } d \in D$$

In this equation, the output value at location (d,x,y) is computed by taking the dot product between the F-dimensional feature vectors $\mathbf{f}_l(x,y)$ and $\mathbf{f}_r(x-d,y)$. This cost volume captures the similarity between the left and right features at different disparities, providing valuable information for subsequent disparity estimation. Once the output $3\mathcal{D}$ cost volume is constructed, it is processed by $2\mathcal{D}$ encoder-decoder modules to regress the disparities.

Following the introduction of the correlation volume, several end-to-end methods have been proposed to enhance the overall performance of stereo networks (Pang *et al.*, 2017; Liang *et al.*, 2018; Song *et al.*, 2019; Yang *et al.*, 2018). However, these methods still face limitations in the way features are combined.

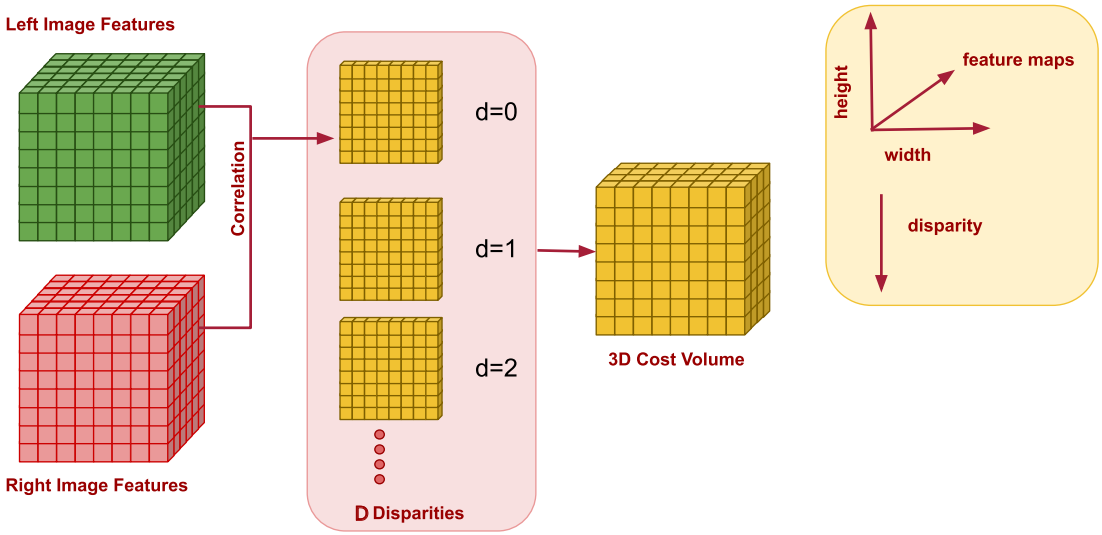


Figure 2.10: Visualization of $3\mathcal{D}$ correlation-based cost volume (with D disparity levels). Each output level d in this volume is constructed by taking the dot product between the left and the right image features shifted by d positions.

Concatenation based $4\mathcal{D}$ Cost Volume:

In comparison to the correlation-based cost volume, the concatenation-based cost volume, introduced by Kendall et al. (Kendall *et al.*, 2017), does not summarize the features via a correlation operator. Instead, it constructs a $4\mathcal{D}$ cost volume by simply stacking the left and right image features shifted by d positions (illustrated in Figure 2.11). Mathematically, the concatenation-based cost volume is defined as follows:

$$\mathbf{CV}_{\text{concatenate}}(d,x,y,.) = \text{Concatenate}\left(\mathbf{f}_l(x,y),\mathbf{f}_r(x-d,y)\right) \quad \text{for all } d \in D$$

The main motivation behind the concatenation-based approach is to allow the network to learn the appropriate summarization and combination of features across different disparity levels. By directly stacking the features, the network has the flexibility to capture and utilize the information it deems relevant for disparity estimation. The output of the concatenation-based approach is a $4\mathcal{D}$ cost volume, which is then processed by a cost-aggregation module using a series of $3\mathcal{D}$ convolutional filters. This module refines the information and summarizes it for disparity regression module, enabling the network to learn and generate more accurate and detailed depth estimations.

Since its introduction, the concatenation-based cost volume has gained popularity due to its improved performance in stereo matching. It has been widely adopted by state-of-the-art methods such as the Pyramid Stereo Matching Network (PSMNet) (Chang and Chen, 2018), GA-Net (Zhang *et al.*, 2019), and GC-Net (Kendall *et al.*, 2017). However,

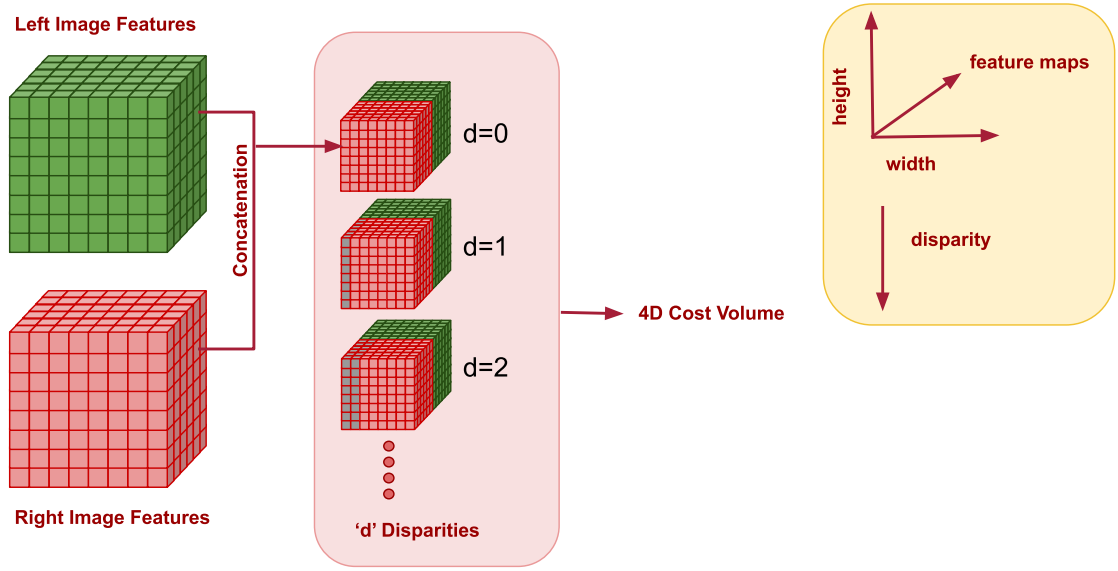


Figure 2.11: Visualization of $4\mathcal{D}$ concatenation-based cost volume. F-dimensional left and right image features are concatenated after horizontally shifting the right image features by d positions to construct one level in the $4\mathcal{D}$ cost volume.

it is important to consider that the use of the concatenation-based cost volume comes with increased computational requirements.

Unlike the correlation-based cost volume, which can be processed by a $2\mathcal{D}$ convolutional neural network (CNN), the concatenation-based approach requires a $3\mathcal{D}$ CNN to handle the additional dimension of the $4\mathcal{D}$ cost volume. This leads to higher computational complexity and memory usage.

The increased computational requirements present challenges both during training and inference. To address this issue, researchers (Gu *et al.*, 2020; Guo *et al.*, 2019a; Wang *et al.*, 2019) have proposed various techniques to optimize the computational efficiency of the $3\mathcal{D}$ CNNs or the cost volume itself. For instance, GwcNet (Guo *et al.*, 2019a) partitions the cost volume into groups and processes them independently to reduce the overall computational complexity. Cascade Cost Volume by Gu *et al.* (2020) gradually reduces the resolution of the cost volume while refining the disparity estimation in a cascaded manner.

Recent methods have explored alternative approaches to stereo matching by either utilizing both cost volumes or directly regressing the disparity maps without constructing a cost volume. For example, Xu *et al.* (2022) propose ACVNet, which uses a correlation cost volume to learn attention weights for refining and enhancing the concatenation cost volume. On the other hand, Xu and Zhang (2020); Tankovich *et al.* (2021) employ feature aggregation modules to aggregate features instead of relying on the cost volume. These methods then directly regress the disparity maps, bypassing the traditional cost

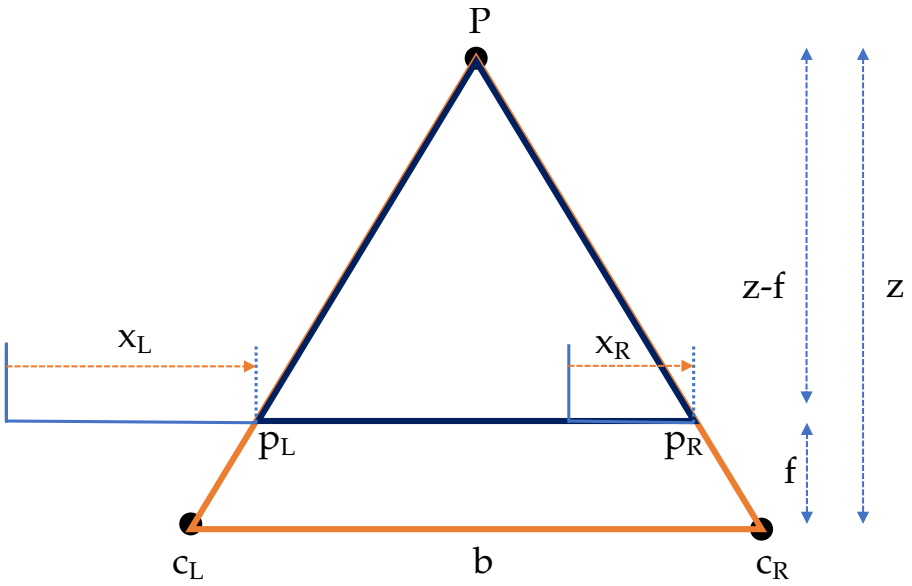


Figure 2.12: Recovering depth from disparity using triangulation. The two triangles are colored differently, and the principle of similar triangles is utilized to determine the depth. Adapted and redrawn from lecture notes by Dr. Christian Unger.

volume construction step.

In this thesis, our main objective was to optimize state-of-the-art networks that leverage correlation, concatenation, and attention-based cost volume modules. In line with existing research, our findings confirmed that both concatenation and attention-based cost volume modules consistently outperformed other methods.

2.4 Triangulation

Once we have computed the disparity estimates, the next step involves converting these estimates into depth values using the principle of triangulation.

Let's consider a bird's-eye view scenario with rectified cameras, as depicted in Figure 2.12. In this configuration, a three-dimensional point, denoted as P, is projected onto the left and right image planes, resulting in points p_L and p_R, respectively. The horizontal positions of the projected point P in the image plane are represented as x_L and x_R. The cameras, positioned at a baseline distance b apart, have a focal length of f. The distance between the 3D point P and the camera centers, denoted as z, represents the depth of the point.

To calculate the depth, we define the length of the base of the triangle $\triangle P p_L p_R$ as:

$$l = (b + x_R) - x_L$$

Given that the disparity is defined as the horizontal displacement $d = x_L - x_R$, we can rewrite l as:

$$l = b - d$$

By applying the principle of similar triangles to $\triangle Pp_Lp_R$ and $\triangle Pc_Lc_R$ in Figure 2.12, we can derive the value of z:

$$\begin{aligned} \frac{z-f}{b-d} &= \frac{z}{b} \\ zb - fb &= zb - zd \\ z &= \frac{fb}{d} \end{aligned}$$

This derivation shows that, given the camera parameters (f and b) and the disparity information, we can compute the depth of a given three-dimensional scene.

2.5 Stereo Matching Datasets and Evaluation Metrics

Stereo matching algorithms heavily rely on stereo matching datasets and evaluation metrics for their development and evaluation. These resources provide standardized benchmarks to assess the performance of different stereo matching techniques. Over the years, the stereo vision research community has introduced a variety of datasets to facilitate the design of more sophisticated stereo matching algorithms and evaluate their effectiveness.

Initially, stereo matching datasets primarily consisted of a limited amount of data captured in controlled indoor environments. However, as the computer vision community acknowledged the significance of large and diverse datasets in fostering the development of superior and more robust algorithms, stereo vision researchers responded by introducing large-scale datasets captured in real outdoor environments. These datasets serve as valuable assets for researchers to address the challenges encountered in real-world scenarios and improve the performance of stereo vision algorithms.

Now, let's delve into the details of some commonly used stereo matching datasets and evaluation metrics.

2.5.1 KITTI Dataset

The KITTI 2012 dataset (Geiger *et al.*, 2012) is the first large-scale outdoor stereo dataset. It includes 389 grayscale stereo pairs captured from a car equipped with two pairs of stereo cameras: one colored pair and one monochrome pair. The car is also equipped with LIDAR, GPS, and inertial sensors, as shown in Figure 2.13. The dataset is divided into 194 training pairs and 195 test pairs. After rectification, the images have a resolution of 1240×376 pixels.

In both the training and test sets, the dataset provides sparse labeling, where only approximately one-third of the pixels are labeled. The training pairs with ground-truth labels are available for download. However, to evaluate the performance on the test set, participants are required to upload their prediction results to the evaluation server*. The evaluation server offers a set of metrics to assess performance. Two commonly used metrics are:

(i) *k*-pixel Error: This metric measures the percentage of pixels whose disparity error is greater than *k* pixels. For instance, *3-px* error represents the percentage of pixels having a disparity error greater than 3 pixels.

(ii) End-Point-Error (EPE): EPE measures the average disparity error across all pixels.

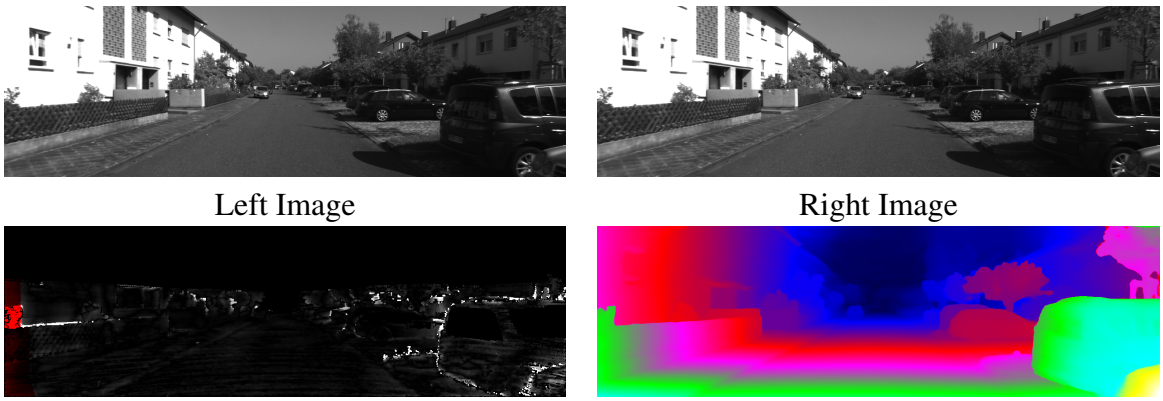
Figure 2.13: Recording platform used for capturing the KITTI Benchmark Suite dataset (Geiger *et al.*, 2012). The vehicle is equipped with diverse a set of sensors.

For the qualitative evaluation of the results, the KITTI 2012 benchmark provides disparity error maps and estimated disparity maps for each image pair. Figure 2.14 shows an example of an input image pair, a disparity error map, and a dense disparity map generated for one of the methods (Shen *et al.*, 2022) submitted to the benchmark server. The disparity error map visualizes the magnitude of disparity errors and is color-coded on a linear scale. It ranges from black (0 pixels of error) to brighter shades (5 pixels of error). Note that any occluded pixels falling outside the image boundaries are represented by the color red.

A few years later, an enhanced version of the KITTI 2012 dataset was introduced as part of the KITTI benchmark suite (Menze and Geiger, 2015). The KITTI 2012 dataset consisted of grayscale images depicting static scenes, while the new version, called KITTI 2015, introduced 400 stereo pairs (colored) capturing dynamic scenes with independently moving cars. These pairs were further split into 200 training pairs and 200 test pairs.

The primary evaluation metrics for this dataset include the percentage of pixels with an absolute disparity error greater than 3 pixels and the percentage of pixels with a relative

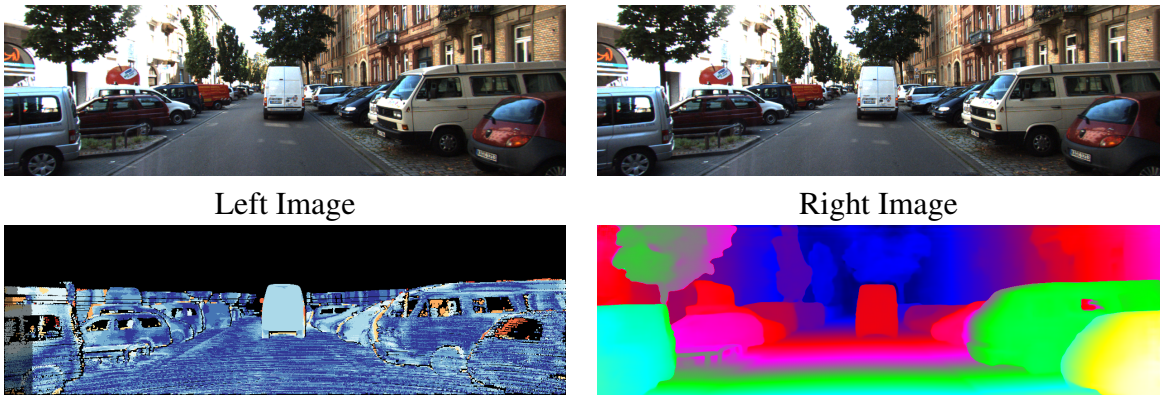
*https://www.cvlibs.net/datasets/kitti/eval_stereo.php



Left Image | Right Image

Disparity Error Map | Estimated Disparity Map

Figure 2.14: An example from the KITTI 2012 dataset showing a test image pair along with the corresponding disparity error map and estimated disparity map using a method proposed by Shen *et al.* (2022).



Left Image | Right Image

Disparity Error Map | Estimated Disparity Map

Figure 2.15: An example of a test image pair from the KITTI 2015 dataset, along with the disparity error map and disparity map estimated using Shen *et al.* (2022)'s method.

error larger than 5% (referred to as the D1 error). These error rates can be calculated for all pixels, background pixels, or only foreground pixels belonging to moving objects. In terms of qualitative evaluation, the benchmark provides a disparity map (colored) and disparity error map for each image pair. Figure 2.15 presents an example of estimated disparity and error maps obtained using one of the recent benchmark methods Shen *et al.* (2022)). In the error map, disparity estimates are color coded with the color changing from blue to red as the disparity error increases. Dark regions within the error images represent occluded pixels that fall outside the image boundaries.

2.5.2 Middlebury Dataset

The Middlebury dataset (Scharstein *et al.*, 2014) was the first benchmark that allowed authors to submit their results for online[†] evaluation. It consists of real-world stereo pairs with dense ground-truth annotations, focusing exclusively on indoor environments. The Middlebury 2014 dataset comprises a total of 33 stereo pairs, with 10 pairs designated for testing, 10 pairs for training, and an additional 13 pairs for supplemental use. Ground-truth labels are available for the training and additional images. This dataset is relatively more challenging compared to the KITTI dataset as it features high-resolution images, specifically 6 megapixels compared to the 0.3 megapixels resolution of the KITTI dataset. The images and labels are provided in full (F), half (H), and quarter (Q) resolutions.

The evaluation metrics employed for the Middlebury dataset include the percentage of pixels where the disparity error exceeds certain thresholds, namely $0.5, 1, 2$, and 4 pixels. These metrics serve as indicators of accuracy and precision in estimating disparities. Figure 2.16 shows an example image from the Middlebury dataset,

Left Image | Right Image | Ground-truth Disparity Map

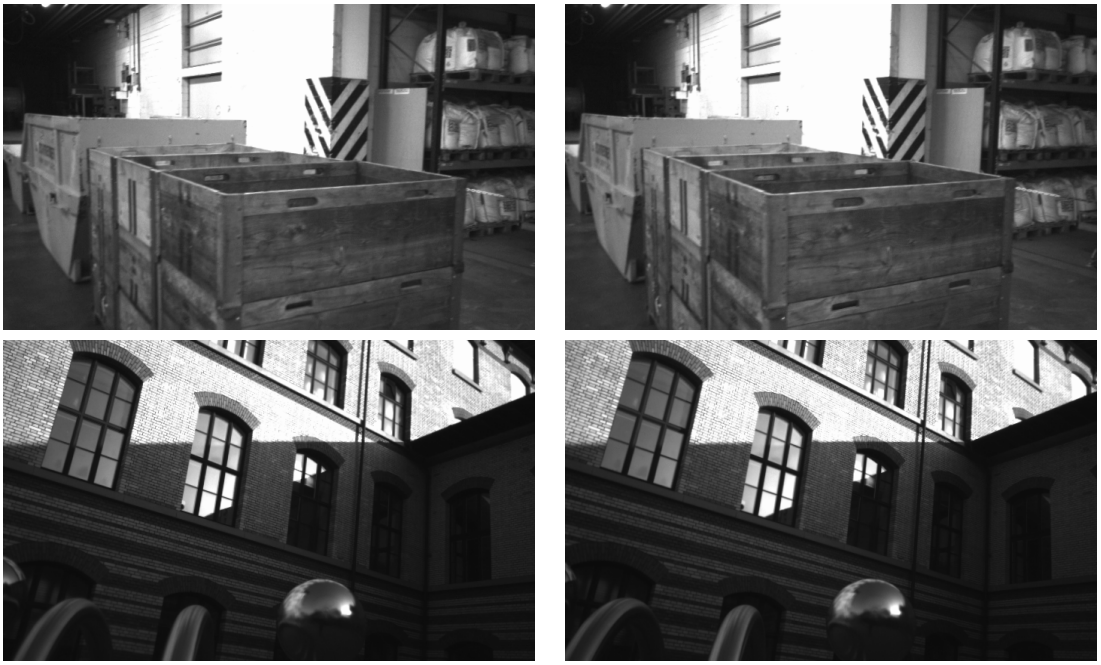
Figure 2.16: An example of a stereo pair and its corresponding ground-truth label from the Middlebury dataset (Scharstein *et al.*, 2014).

2.5.3 ETH3D Dataset

The ETH3D two-view stereo dataset (Schops *et al.*, 2017) consists of 47 grayscale stereo pairs captured in both indoor and outdoor environments. The dataset is divided into 27 training pairs and 20 test pairs. Notably, the ground truths for the test set are withheld, allowing for the evaluation of different methods using an online benchmark platform[‡] that employs the same evaluation metrics introduced by the Middlebury dataset. Fig. 2.17 presents sample images from the ETH3D dataset, illustrating scenes captured in both indoor and outdoor environments.

[†] https://vision.middlebury.edu/stereo/

[‡] `https://www.eth3d.net/overview`



Left Image Right Image

Figure 2.17: Sample stereo pairs from the ETH3D dataset (Schops *et al.*, 2017) from both the indoor (top) and outdoor (bottom) subsets.

2.5.4 SceneFlow Dataset

With the increasing popularity and success of deep neural network-based methods in computer vision, including stereo vision, the need for a large-scale stereo dataset became evident to enhance the performance of these methods. The SceneFlow (Mayer *et al.*, 2016) is a groundbreaking dataset in the field of stereo matching and has been widely utilized by most methods to date. Due to the data-hungry nature of neural networks, many stereo methods are initially pre-trained on the large scale SceneFlow dataset and subsequently fine-tuned using smaller domain-specific datasets such as KITTI and Middlebury.

The SceneFlow dataset is a synthetic dataset that provides dense ground truth annotations. It consists of 39,824 stereo pairs with a resolution of 540×960 pixels, which are split into 35,454 training pairs and 4,370 test pairs. The dataset comprises three subcategories: *FlyingThings3D, Monkaa,* and *Driving*. The *FlyingThings3D* subset contains images with dynamic foreground objects overlaid on a structured background consisting of various geometric shapes. The *Monkaa* subset is generated from an animated movie, while the *Driving* subset contains scenes depicting driving scenarios. Fig. 2.18 shows sample pairs from these subsets. The quantitative performance on the SceneFlow dataset is typically evaluated by measuring the average disparity error.

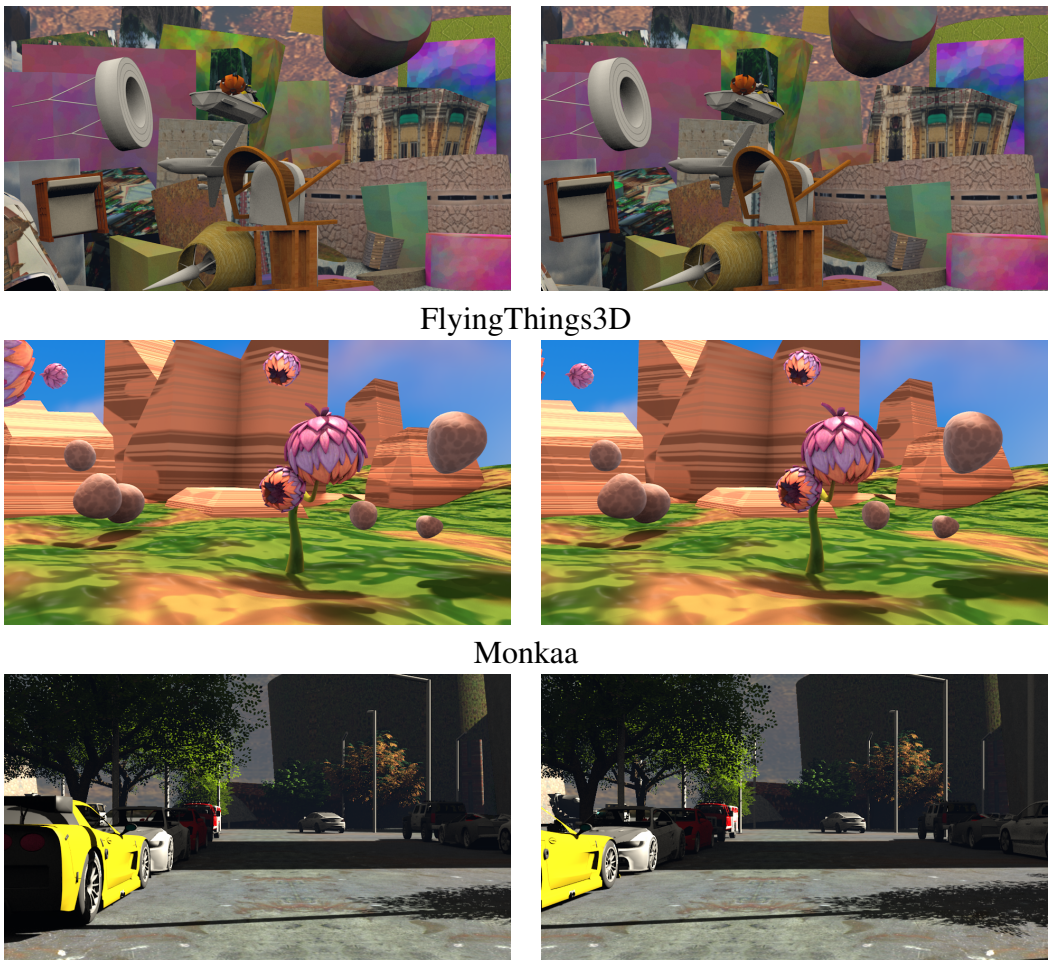


Figure 2.18: Sample images from the SceneFlow dataset (Mayer *et al.*, 2016). Each row shows a corresponding pair of left and right images from different subsets.

2.5.5 Other Datasets

In addition to the datasets presented, there are other less commonly used stereo datasets that offer valuable resources for developing and evaluating depth estimation algorithms. While these datasets may not be as widely utilized, they provide opportunities for the research community to address challenges related to volume, complexity, diversity, and data quality in stereo vision. Here, we provide a summary of these datasets:

- **MPI Sintel:** It is a synthetic dataset (Wulff *et al.*, 2012) extracted from animated movies, providing dense annotations. It consists of 1,064 training frames and 564 test frames with a resolution of 1024x436 pixels.

- **CARLA:** CARLA is an open-source simulator (Dosovitskiy *et al.*, 2017) used to generate a synthetic dataset for the urban driving environment. It aims to support research in autonomous driving by providing realistic scenarios and ground truth annotations.

- **Oxford Robotcar:** This dataset (Maddern *et al.*, 2017) contains real-world data collected for driving scenes using a trinocular camera setup. It captures stereo pairs with both wide and narrow baselines, covering over 100 km of navigation.

- **Apolloscape:** Apolloscape is a high-resolution dataset (Huang *et al.*, 2019) with 5,165 stereo pairs (4,156 training and 1,009 test) for driving scenes. It provides dense ground truth annotations and has a resolution of 3 megapixels.

- **DrivingStereo:** This dataset (Yang *et al.*, 2019) consists of 180,000 stereo pairs capturing various weather conditions and driving scenarios. It covers urban, sub-urban, highway, elevated, and country roads. The dataset provides semi-dense ground truth annotations with 174,437 train pairs and 7,751 test pairs.

- **Holopix50k:** Holopix50k (Hua *et al.*, 2020) is a large-scale stereo dataset captured from mobile phone cameras in the wild. It contains 50,000 rectified stereo pairs contributed by users from the Holopix mobile-based social platform. The dataset covers diverse categories such as fire, landscape, light, selfies, sunset, and water. Note that there are no ground-truth annotations available for this dataset.

Chapter 3

Separable Convolutions for Optimizing $3\mathcal{D}$ Stereo Networks

As discussed in Chapter 2, end-to-end deep learning methods, in general, consist of four main steps. In the first step, a *'backbone network'* is used to compute features from both the input images. In the second step, extracted features from both the images are fused to form a *'cost volume'* – *c.f*. Fig. 3.1. In the third step, cost volume is further processed via a sequence of convolutional layers to do *'cost aggregation'* and as a final step we *'regress the disparities'*. Broadly speaking, these methods can be broadly classified into two categories: two-dimensional ($2\mathcal{D}$) or three-dimensional ($3\mathcal{D}$) methods. This classification depends on the methodology used to fuse features, the types of layers employed, and the convolutions applied in the cost aggregation and disparity regression steps.

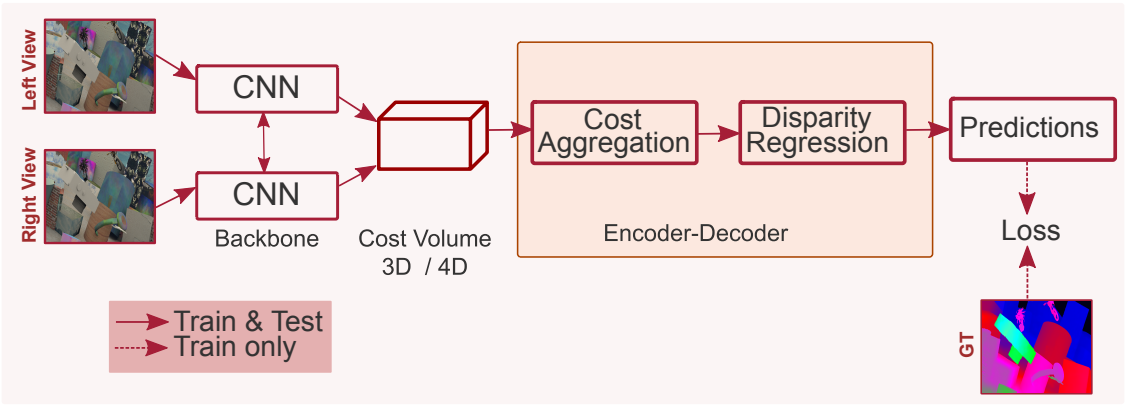


Figure 3.1: This chapter specifically addresses the major computation bottleneck in state-of-the-art $3\mathcal{D}$ stereo networks, namely the encoder-decoder step (shaded in the figure). Our approach involves designing specially optimized operations to effectively reduce the overall computational load.

In this chapter, we present our work on "Separable Convolutions for Optimizing $3\mathcal{D}$ Stereo Networks" by Rahim *et al* (2021). We start by conducting a thorough profiling of three-dimensional ($3\mathcal{D}$) stereo methods. Based on the insights gained from this analysis, we focus specifically on the encoder-decoder component of the network to reduce the

overall computational demands, including parameters and operations, required by state-of-the-art (SOTA) stereo networks.

3.1 Introduction

Even though 3$\mathcal{D}$ methods give far superior performance compared to 2$\mathcal{D}$ methods, this performance improvement comes at the cost of extra computational budget. In fact, these 3$\mathcal{D}$ methods are orders of magnitude slower than 2$\mathcal{D}$ ones. A lot of effort in the community has been recently targeted to make these methods computationally efficient (Wang *et al.*, 2019; Tulyakov *et al.*, 2018; Duggal *et al.*, 2019). Although the majority of these methods target optimization of the cost-volume construction step, they still use the costly 3$\mathcal{D}$ convolutions for disparity regression. In this work, we aim to further reduce the overall computational footprint of the 3$\mathcal{D}$ methods by optimizing 3$\mathcal{D}$ convolutions.

For this, we first do a thorough empirical investigation of the state of the art 3$\mathcal{D}$ stereo methods (Zhang *et al.*, 2019; Chang and Chen, 2018) to identify major contributing factors to the overall computational cost of these methods. As expected, our investigations conclude that 3$\mathcal{D}$ convolutions are by far the most costly operations in these methods. For instance, in GANetdeep, 3$\mathcal{D}$ convolutions consume around 94% of operations – *c.f*. Table 3.1. Next, we propose methods for optimizing 3$\mathcal{D}$ convolutions, *i.e*. how we can optimize 3$\mathcal{D}$ convolutions without sacrificing the performance of the 3$\mathcal{D}$ stereo networks.

To this end, motivated by the recent performance of light-weight 3$\mathcal{D}$ networks for visual recognition (Qiu *et al.*, 2017; Ye *et al.*, 2019), we propose to replace costly 3$\mathcal{D}$ convolution operations in 3$\mathcal{D}$ stereo networks with their light-weight counterparts (*i.e*. separable 3$\mathcal{D}$ convolutions) to build efficient stereo networks. For this we design and explore three different versions of separable 3$\mathcal{D}$ convolutions: (i) Feature-wise Separable Convolutions (FwSCs); (ii) Disparity-wise Separable Convolutions (DwSCs); and (iii) Feature-&-Disparity-wise Separable Convolutions (FDwSCs). We empirically evaluate these separable 3$\mathcal{D}$ convolution versions as "plug-&-run" replacement for existing 3$\mathcal{D}$ convolutions.

Overall, we make the following contributions:

i. we present an in-depth cost analysis of major building blocks of state of the art 3$\mathcal{D}$ stereo networks;
ii. we design and propose separable 3$\mathcal{D}$ convolutions for stereo networks;
iii. we empirically evaluate and show the "plug-&-run" characteristic of proposed convolutions for state-of-the-art methods on standard datasets.

Our results show that FwSCs and FDwSCs can be reliably used as a replacement for 3$\mathcal{D}$ convolutions in stereo networks without sacrificing their performance while reducing the number of operations (up to 7.2×) and number of parameters (up to 3.5×) – *c.f*. Fig. 3.2 & Table 3.3. To the best of our knowledge, this is the first work to explicitly optimize 3$\mathcal{D}$ convolutions involved in 3$\mathcal{D}$ stereo networks.

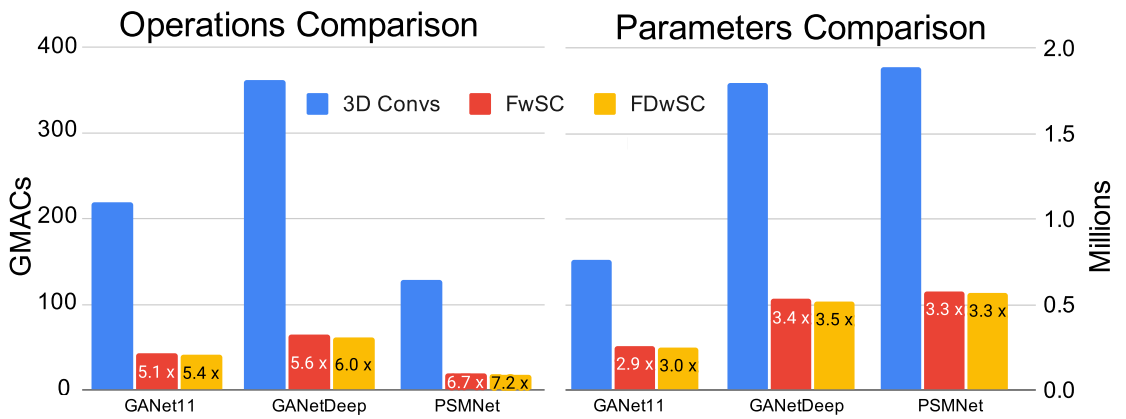


Figure 3.2: Relative comparison between $3\mathcal{D}$ convolutions, Feature-wise Separable Convolutions (FwSCs) and Feature-&-Disparity-wise Separable Convolutions (FDwSCs). We compare them *w.r.t.* number of operations and parameters when applied to different state-of-the-art methods including PSMNet (Chang and Chen, 2018) and GANet (Zhang *et al.*, 2019). Here $\times$ represents the reduction factor and GMACs represents Giga-MACs (1 MAC is one addition and one multiplication operation).

3.2 Related Work

In general, deep learning stereo methods can be broadly categorized into $2\mathcal{D}$ or $3\mathcal{D}$ learning methods (Poggi *et al.*, 2021) based on how they merge information across views and then process it. For instance, $2\mathcal{D}$ methods merge features from the left and right images into a $3\mathcal{D}$ cost volume and then apply $2\mathcal{D}$ convolutions. DispNet-C (Mayer *et al.*, 2016), iResNet (Liang *et al.*, 2018), MADNet (Tonioni *et al.*, 2019), SegStereo (Yang *et al.*, 2018), EdgeStereo (Song *et al.*, 2019) are some examples of $2\mathcal{D}$ methods. On the other hand, $3\mathcal{D}$ methods first generate a $4\mathcal{D}$ cost volume by merging features from the left and right images (under stereo constraints) and use $3\mathcal{D}$ convolutional layers to refine and regress dense disparities. GC-Net (Kendall *et al.*, 2017), PSMNet (Chang and Chen, 2018), GANet (Zhang *et al.*, 2019), AnyNet (Wang *et al.*, 2019) and DeepPruner (Duggal *et al.*, 2019) are some examples of $3\mathcal{D}$ methods. Overall $3\mathcal{D}$ methods produce better results but at the expense of extra computational cost. Although recent methods propose to optimize the $3\mathcal{D}$ cost volume construction step, they still use costly $3\mathcal{D}$ convolutions for cost aggregations, refinement and disparity regression. For instance, Wang *et al.* (2019) use hierarchical $3\mathcal{D}$ cost volumes to reduce $4\mathcal{D}$ volume construction cost while Tulyakov *et al.* (2018) introduce bottleneck matching modules to efficiently match features across disparities. In another work, Duggal *et al.* (2019) employs patch-matching to sample a range of disparities during volume construction.

Model	Metric	2$\mathcal{D}$ CNNs		3$\mathcal{D}$ CNNs		Total
			%		%	
GANet	# Params (M)	2.4	53.3%	0.8	17.7%	4.5
	# Operations (GMACs)	4.8	2.0%	217.4	91.9%	236.6
GANetdeep	# Params (M)	2.4	36.3%	1.8	27.2%	6.6
	# Operations (GMACs)	4.8	1.3%	358.9	93.5%	383.8
PSMNet	# Params (M)	3.3	63.5%	1.9	36.5%	5.2
	# Operations (GMACs)	58.2	31.4%	127.1	68.6%	185.3

Table 3.1: Detailed computational cost profiling of 3$\mathcal{D}$ stereo networks. Here % represents the fraction of parameters (Params in millions) and operations (GMACs) relative to complete network (image size for GANet models is 240×528 and PSMNet is 256×512 pixels.). Note that % might not add up to 100% in this table because we have not reported computations involved in cost volume module as they are insignificant as compared to the 2$\mathcal{D}$ and the 3$\mathcal{D}$ CNNs. We can see that although 3$\mathcal{D}$ CNNs have fewer parameters compared to 2$\mathcal{D}$ CNNs, they still consume the majority of operations.

3.3 Methodology

An end-to-end 3$\mathcal{D}$ stereo pipeline consists of these modules:

1. Feature extraction: a 2$\mathcal{D}$ shared backbone-network to extract features from left and right images of a rectified stereo pair.
2. Volume construction: a 4$\mathcal{D}$ cost volume constructed by merging left and right images features maps.
3. Cost aggregation & disparity regression: a 3$\mathcal{D}$ network to aggregate cost volume and then regress and refine disparities from aggregated cost volume.

In this section, we present the detailed module-wise profiling of the 3$\mathcal{D}$ stereo networks and our approach of seprarable convolutions to optimize the computational burden of these network to design light-weight stereo networks.

3.3.1 Profiling 3$\mathcal{D}$ Stereo Networks

As the first step, we profile the overall computational requirements of the state-of-the-art 3$\mathcal{D}$ stereo networks (Zhang *et al.*, 2019; Chang and Chen, 2018). Specifically, we record the number of parameters and operations involved in feature extraction and cost aggregation & disparity regression modules – *c.f*. Table 3.1. We report the number of parameters in millions (M) and number of operations/MACs (Multiply-ACcumulate operations) in Giga-MACs, where 1MAC = 1 multiplication + 1 addition operations. We can observe

from Table 3.1 that despite having fewer parameters than $2\mathcal{D}$ networks, $3\mathcal{D}$ networks consume more operations in all the networks and thus represent clear bottlenecks. For instance, in GANet11 (Zhang *et al.*, 2019) $3\mathcal{D}$ CNN contains 0.8×10^6 parameters (17% relative to complete network) compared to $2\mathcal{D}$ feature extraction module that requires 2.4×10^6 parameters (53.3%). However, here $3\mathcal{D}$ CNN consumes 217 GMACs operations (91.9% of overall operations) compared to 4.8 GMACs (2 %) of the $2\mathcal{D}$ network. Similar observations can be made about other networks.

Once we have built a detailed insight into the reasons behind these networks computational overhead, we set out to network design choices to reduce the computational overhead without compromising their performance.

3.3.2 Separable $3\mathcal{D}$ Stereo Networks

Light-weight convolutional networks like MobileNet, ShuffleNet, *etc*. (Sandler *et al.*, 2018; Zhang *et al.*, 2018) have been proposed to improve the computational efficiency of $2\mathcal{D}$ visual recognition convolutional networks. These methods propose to use depth-wise separable variants of $2\mathcal{D}$ convolutions to reduce the number of operations. However, in comparison to a $2\mathcal{D}$ convolution kernel, a $3\mathcal{D}$ convolution kernel works on a $4\mathcal{D}$ volume and aggregates information across spatial, depth and channels dimensions, and thus there are different possible ways a $3\mathcal{D}$ convolution can be made separable. For instance, a $3\mathcal{D}$ kernel ($k \times k \times k$, *e.g.* $k = 5$) working on an input $4\mathcal{D}$ cost volume of size $h_i \times w_i \times d_i \times c_i$, where h_i is the height, w_i is the width, d_i is the depth (disparity), and c_i are the channels of the input, aggregates information from a window of size $k \times k \times k \times c_i$ in each of its applications. Furthermore, as a $3\mathcal{D}$ convolutional layer contains many $3\mathcal{D}$ kernels, each producing a new feature map, so overall there are $(h_i \times w_i \times d_i \times (k \times k \times k \times c_i)) \times c_o$ operations performed in each layer – here c_o is number of output channels (number of kernels) in a layer. From this, we can see that these $3\mathcal{D}$ convolutions can be made separable across either d_i or c_i dimension or even simultaneously both across d_i and c_i. We intend to reduce computational requirement by making $3\mathcal{D}$ convolutions separable in one or more of the given dimensions.

Feature-wise Separable Convolutions (FwSCs):

As a first replacement for $3\mathcal{D}$ convolutions in stereo networks, we propose to split the convolutions across the feature (or channel) dimension. FwSCs works in two steps: in the first step, the $4\mathcal{D}$ cost volume of size $h_i \times w_i \times d_i \times c_i$ is split into c_i cubes, and c_i kernels of size $k \times k \times k$ are applied to generate c_i output cubes – *c.f*. Fig. 3.3.

In the second step, c_o point-wise kernels of size $1 \times 1 \times 1 \times c_i$ are applied to aggregate information across the c_i cubes. These steps lead to a significant reduction in the number of operations, as the total number of operations reduces to $(h_i \times w_i \times d_i \times k \times k \times k) \times c_i + (h_i \times w_i \times d_i \times 1 \times 1 \times 1 \times c_i) \times c_o$ compared to $3\mathcal{D}$ convolutions. When applied in the state of the art stereo networks, FwSCs lead up to 6.7× reduction in GMACs

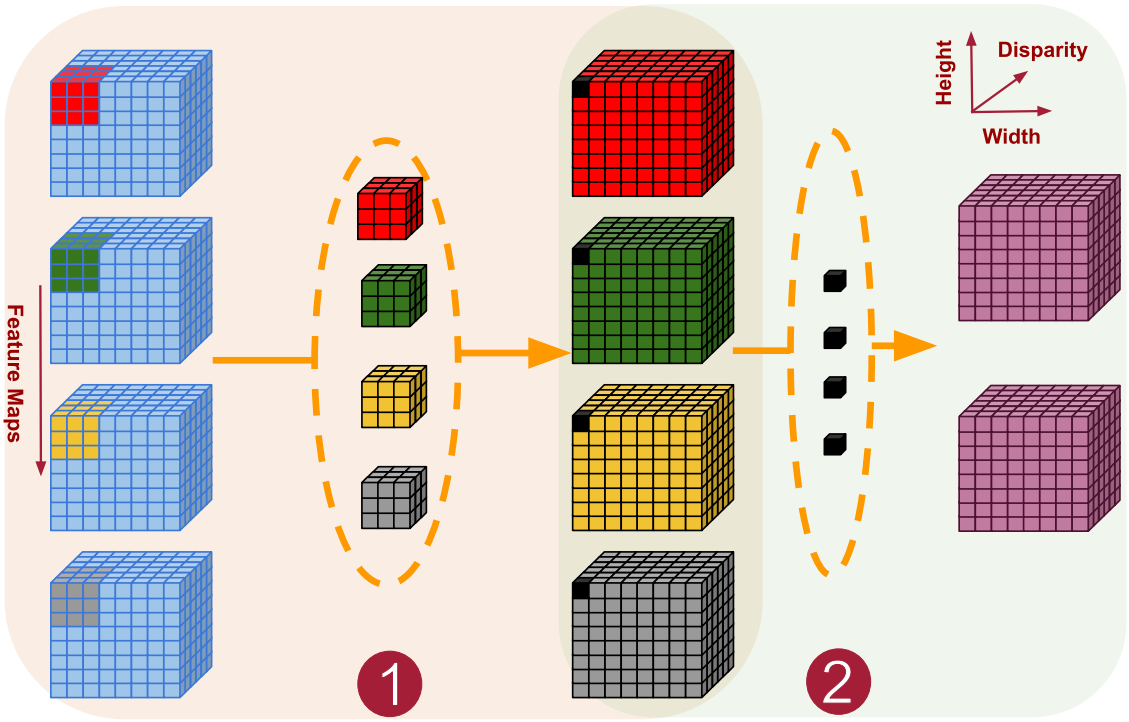


Figure 3.3: Feature-wise Separable Convolutions (FwSCs).

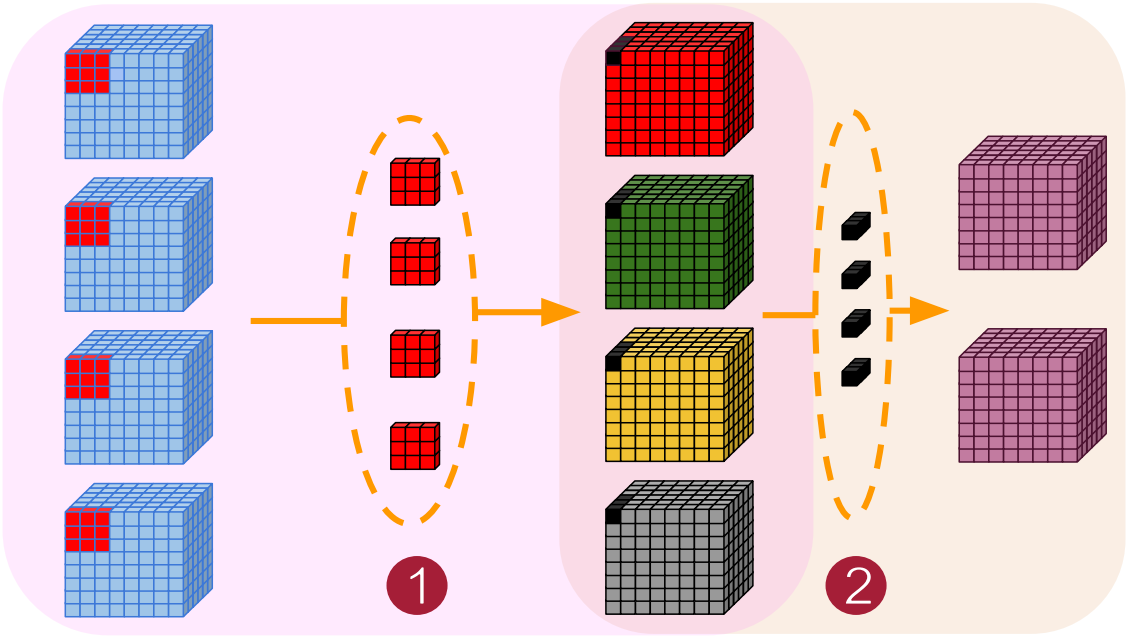


Figure 3.4: Disparity-wise Separable Convolutions (DwSCs).

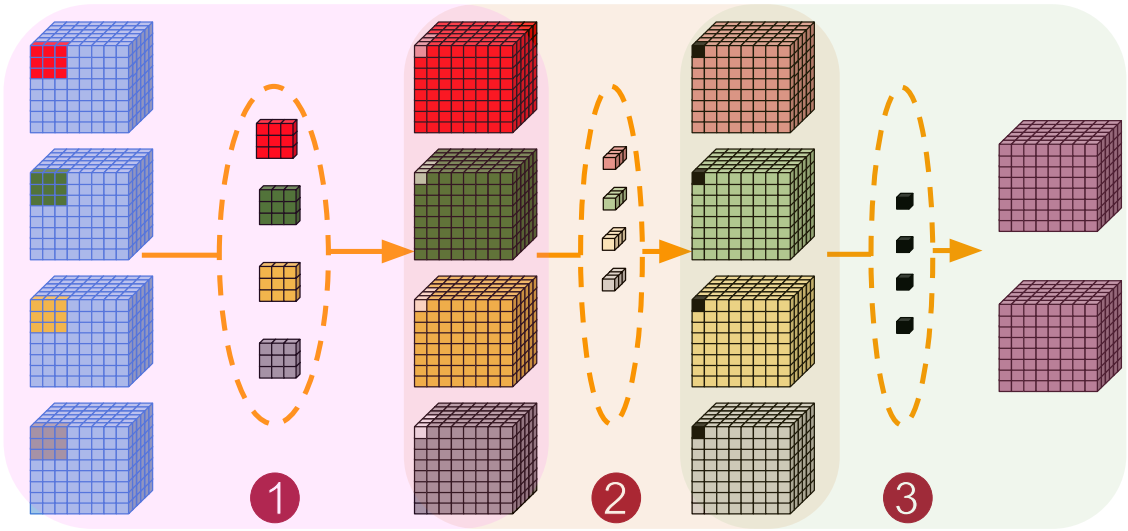


Figure 3.5: Feature-Disparity-wise Separable Convolutions (FDwSCs).

Model	#3$\mathcal{D}$ Convs	Metric	3$\mathcal{D}$ Convs	FwSCs		FDwSCs	
					Red.		Red.
GANet11	11	# Params (M)	0.76	0.26	2.9×	0.25	3.0×
		# Operations (GMACs)	217.44	43.05	5.1×	40.60	5.4×
GANetdeep	15	# Params (M)	1.80	0.53	3.4×	0.52	3.5×
		# Operations (GMACs)	358.85	64.03	5.6×	60.06	6.0×
PSMNet	22	# Params (M)	1.89	0.58	3.3×	0.57	3.3×
		# Operations (GMACs)	127.12	19.06	6.7×	17.70	7.2×

Table 3.2: Comparison of parameters and operations utilized by 3$\mathcal{D}$ convolutions vs separable 3$\mathcal{D}$ convolutions *i.e.* FwSCs and FDwSCs. Red. is the reduction factor.

(see Table 3.2)* with better performance than 3$\mathcal{D}$ convolutions in the majority of cases (Sec. 3.4).

Disparity-wise Separable Convolutions (DwSCs):

DwSCs are identical to FwSCs, except the separable operation is performed in the disparity dimension. Precisely, in the first step the 4$\mathcal{D}$ cost volume of size $h_i \times w_i \times d_i \times c_i$ is split into d_i cubes (after permuting $h_i \times w_i \times d_i \times c_i$ to $h_i \times w_i \times c_i \times d_i$) and d_i kernels of size $k \times k \times k$ are applied to produce d_i output cubes.

In the second step, c_o kernels of size $1 \times 1 \times 1 \times d_i$ are applied to aggregate information across the d_i cubes – *c.f.* Fig. 3.4. Finally, the output volume is permuted back. Although

*Please note that these calculations also include MACs operations for biases and batch-normalization.

DwSCs lead to a reduction in the number of operations, the overall reduction is not as significant as in FwSCs, because usually in stereo networks the size of the disparity dimension is greater than the size of feature dimension. For instance, in GANetdeep the size of the disparity dimension is 48, whereas in almost all the layers the size of the feature dimension is 32.

Moreover, for a given pixel, first aggregating the information across the feature dimension and then across the disparity dimension does not appear to be the optimal choice. This was confirmed by our initial experiments, which returned much worse results, thus we do not report further experimental results on the DwSCs.

Feature-Disparity-wise Separable Convolutions (FDwSCs)

FDwSCs are extremely separable variants of $3\mathcal{D}$ convolutions and built by adding an extra layer of separability on FwSC. These convolutions are built in three steps (see Fig. 3.5). In the first step, similar to FwSCs, the input cost volume is split into c_i cubes, and c_i disparity-wise separable kernels of size $k \times k \times 1$ are applied to each cube to aggregate spatial information and generate c_i output cubes. In the second step, c_i point-wise kernels of size $1 \times 1 \times k$ are applied to each cube independently to aggregate disparity information. As final step, c_o kernels of size $1 \times 1 \times 1 \times c_i$ are applied to aggregate information across the feature dimension. The total number of operations involved in FDwSCs is equal to $(h_i \times w_i \times d_i \times (k \times k \times 1)) \times c_i + (h_i \times w_i \times d_i \times (1 \times 1 \times k)) \times c_i + (h_i \times w_i \times d_i \times 1 \times 1 \times 1 \times c_i) \times c_o$.

As a result, FDwSCs lead to a further reduction in the number of parameters and operations than all other types of convolutions discussed. Precisely, they lead to a reduction of 3.3× in parameters and 7.2× in GMACs for PSMNet and 3.5× in parameters and 6.0× in GMACs for GANetdeep – *c.f*. Table 3.2. Surprisingly, this reduction in the number of parameters and operations does not significantly impact the performance – *c.f*. Table 3.3.

3.4 Experimental Results

We trained and evaluated our networks on two popular stereo benchmark datasets.

SceneFlow (Mayer *et al.*, 2016) is a large scale synthetic dataset and contains $35,454$ training images and $4,370$ test images of size 540×960 with dense annotations. For our experiments, we further divide the training dataset to $32,454$ training and 3000 validation pairs following (Zhang *et al.*, 2019) .

KITTI 2015 (Menze and Geiger, 2015) is a real dataset of driving scenes. It only contains 200 training and 200 test image pairs of size 376×1240 pixels with sparse annotations. For our experiments, we further divide the training dataset into 150 training and 50 validation pairs as in (Zhang *et al.*, 2019).

3.4.1 Training Details

For our baseline networks, we use the publicly released code of GANetdeep and PSMNet. For fair comparison and to show the "plug-&-run" capabilities of separable $3\mathcal{D}$ convolutions we use identical training settings for both baseline networks (*i.e.* networks with $3\mathcal{D}$ convolutions) and optimized networks (*i.e.* networks with separable $3\mathcal{D}$ convolutions). Furthermore, it is important to mention that we do not perform any type of hyperparameters tuning and train all the networks from scratch (given in Table 3.3) with default values as in base networks.

GANet: All the input images are cropped to the size of 240×528 while the maximum disparity is set to 192. Furthermore, all the images are standard normalized using the mean and standard deviation of the training set. For training, we use the Adam Optimizer (λ = 1e-3, $\beta_1 = 0.9$ and $\beta_2 = 0.999$) with a training batch size of 4. For KITTI 2015, we transfer learned the models trained on SceneFlow for 650 epochs with the learning rate reduced to 1e-4 after 300 epochs.

PSMNet: The majority of settings are identical to the baseline except that the images are randomly cropped to 256×512 during training, and we use a batch size of 8. Moreover, for transfer learning on KITTI 2015 the learning rate is reduced after 200 epochs.

Dataset	Model	Metric	$3\mathcal{D}$ Convs	FwSCs	FDwSCs
SceneFlow	GANet11 (Zhang *et al.*, 2019)	3-*px* (%)	4.21	4.02	**3.94**
		D1 error (%)	3.49	3.36	**3.25**
		EPE	0.99	**0.92**	0.94
	GANetdeep (Zhang *et al.*, 2019)	3-*px* (%)	4.01	**3.74**	4.13
		D1 error (%)	3.29	**3.03**	3.45
		EPE	1.01	**0.90**	0.99
	PSMNet (Chang and Chen, 2018)	3-*px* (%)	4.20	**3.72**	3.76
		D1 error (%)	3.48	**3.08**	3.10
		EPE	0.99	**0.90**	**0.90**
KITTI	GANet11 (Zhang *et al.*, 2019)	3-*px* (%)	2.01	**1.94**	2.07
		D1 error (%)	1.92	**1.85**	1.95
		EPE	**0.67**	**0.67**	0.68
	GANetdeep (Zhang *et al.*, 2019)	3-*px* (%)	**1.67**	**1.67**	1.73
		D1 error (%)	1.61	**1.57**	1.65
		EPE	**0.63**	0.64	0.71
	PSMNet (Chang and Chen, 2018)	3-*px* (%)	**2.10**	2.57	2.18
		D1 error (%)	**2.00**	2.50	2.08
		EPE	**0.88**	0.92	0.90

Table 3.3: Quantitative results of networks trained using $3\mathcal{D}$ convolutions, FwSCs and FDwSCs on benchmark datasets.

3.4.2 Results

For quantitative comparison, following the SceneFlow and KITTI 2015 evaluation protocols, we report three evaluation metrics, including 3-pixels, D1 and End-Point-Errors (EPE) (Mayer *et al.*, 2016; Menze and Geiger, 2015) for all the networks . On the SceneFlow dataset (Table 3.3 (top)) we notice that for most of the cases, networks with FwSCs outperform baseline networks with 3$\mathcal{D}$ convolutions despite having around 6.7$\times$ less operations and 3.3$\times$ fewer parameters. The better performance of FwSCs, compared to 3$\mathcal{D}$ convolutions, can be attributed to the fact that a cost volume contains a lot of redundant information (the majority of an image neighbouring disparities do not change) and these separable 3$\mathcal{D}$ convolutions contain enough representation power to model disparity variations. Surprisingly, even extremely separable FDwSCs perform better than regular 3$\mathcal{D}$ convolutions in most of the cases. However, their performance is a little inferior to FwSC, which is somewhat expected, due to the loss of their correlation modeling capabilities in disparity dimension.

By looking at the quantitative results from the KITTI 2015 dataset from Table 3.3 (bottom) we can draw similar conclusions as on the SceneFlow dataset. FwSCs still outperform 3$\mathcal{D}$ convolutional networks with a relatively smaller number of 3$\mathcal{D}$ convolutional layers, *e.g*. GANetdeep has 15 convolutional layers. However, for the networks with a far higher number of 3$\mathcal{D}$ convolutional layers, *e.g*. PSMNet (22 convolutional layers) we see a slight deterioration in their performance – this might be because KITTI 2015 is a small and sparse dataset. FDwSCs also give comparable performance to FwSCs and 3$\mathcal{D}$ convolutions.

Figures 3.6 and 3.7 show qualitative results on sample examples from the KITTI 2015 validation and SceneFlow test set. Here once again we can verify that separable 3$\mathcal{D}$ convolutions are giving comparable results to their 3$\mathcal{D}$ convolutions counterparts.

3.5 Conclusions

In this chapter, we have shown that 3$\mathcal{D}$ convolutional layers are the major bottleneck in overall execution of 3$\mathcal{D}$ stereo networks. To circumvent this we have proposed and empirically shown that separable 3$\mathcal{D}$ convolutions can considerably reduce the number of operations and parameters for stereo matching in state-of-the-art networks. These proposed convolutions not only lead to leaner networks but they also consistently lead to better performance of these networks. We believe that these convolutions "plug-&-run" nature can lead to their integration into many other 3$\mathcal{D}$ stereo networks.

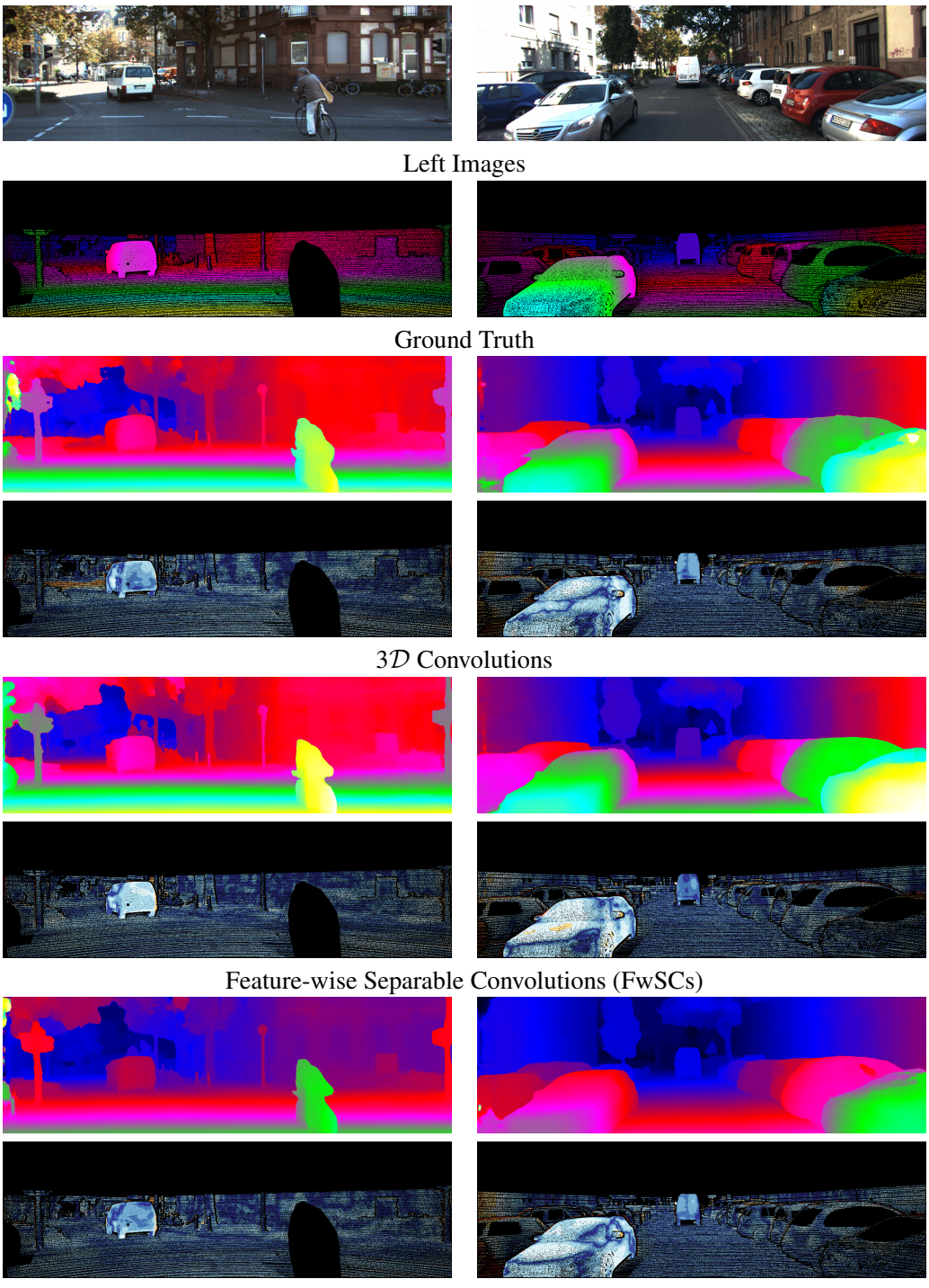


Figure 3.6: Qualitative results of GANet (Zhang *et al.*, 2019) on the KITTI dataset using different types of convolution operations. For each operation, we provide both an estimated disparity map and an error map (to quantify the quality of the estimated disparities). For example, the third and fourth rows show disparity maps and error maps of estimations using 3$\mathcal{D}$ convolution operation. In error maps, darker red and blue colors represent higher and lower disparity errors, respectively.

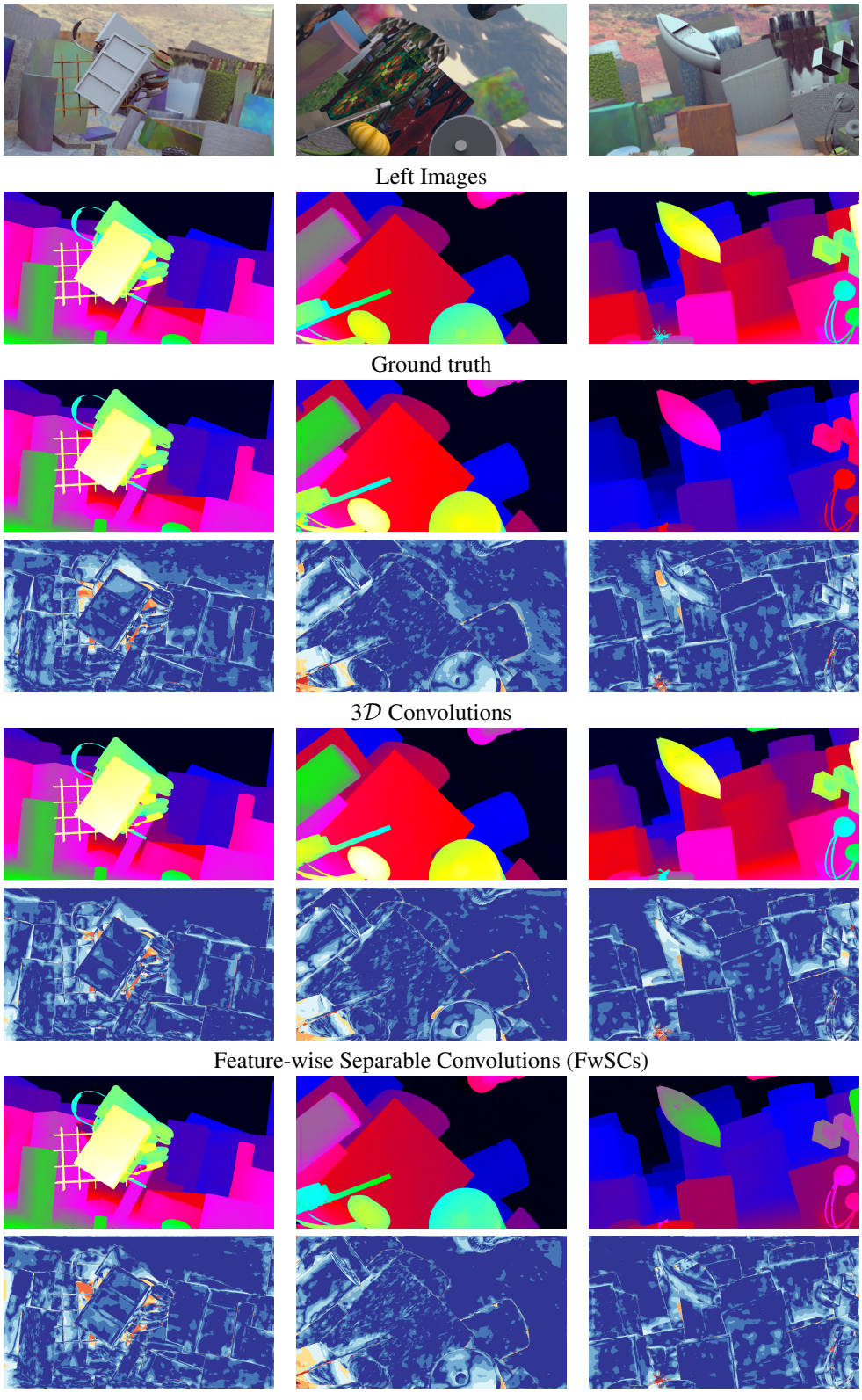


Figure 3.7: Qualitative results on the KITTI dataset of GANet (Zhang *et al.*, 2019) using different types of convolution operations.

Chapter 4

MobileStereoNet: Towards Lightweight Deep Networks for Stereo Matching

4.1 Introduction

In this chapter, we delve further deeper to optimize the individual components of the stereo networks, capitalizing on the advancements made in accelerating state-of-the-art neural networks, as well as the insights gained from designing $3\mathcal{D}$ separable convolutions in Chapter 3.5. Specifically, we present our work titled "MobileStereoNet: Towards Lightweight Deep Networks for Stereo Matching" by Shamsafar *et al* (2022). This work introduces MobileStereoNets, which effectively reduce the computational requirements in the backbone and encoder-decoder parts (module 1 and 3 in Fig. 4.1) of end-to-end stereo networks by utilizing the concept of MobileNet blocks (Howard *et al.*, 2017). In addition to optimizing the backbone and encoder-decoder parts, we introduce a novel cost volume construction method (module 2 in Fig. 4.1) for $2\mathcal{D}$ stereo methods. This new method is designed to improve the overall performance of the network.

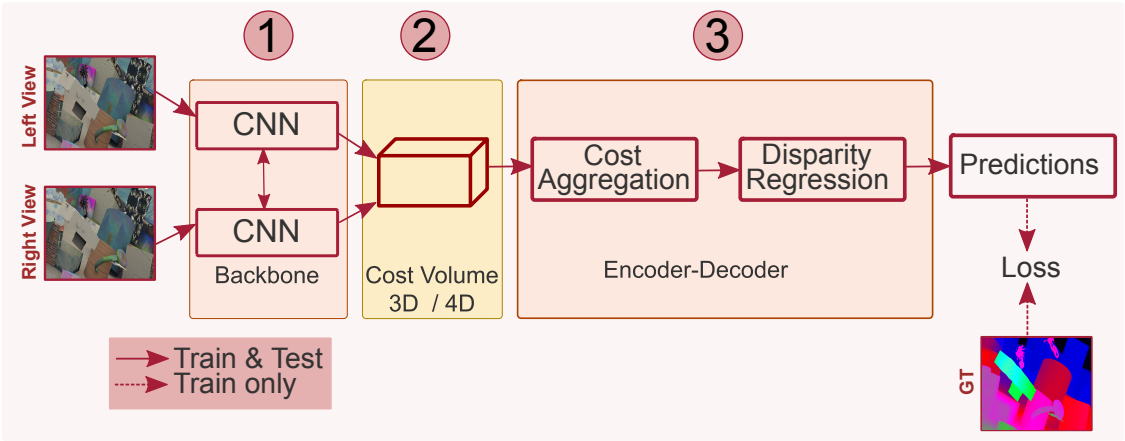


Figure 4.1: In this chapter, mainly, we tailor efficient blocks from faster state-of-the-art neural networks (Howard *et al.*, 2017) to optimize the backbone and encoder-decoder parts of both $2\mathcal{D}$ and $3\mathcal{D}$ stereo networks.

Specifically, we design and build two end-to-end stereo matching networks, which exploit MobileNet-V1 (Howard *et al.*, 2017) and MobileNet-V2 (Sandler *et al.*, 2018) 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.* $3\mathcal{D}$ or $4\mathcal{D}$, we propose a $2\mathcal{D}$ and a $3\mathcal{D}$ network. Moreover, for the $3\mathcal{D}$ 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. 4.2).

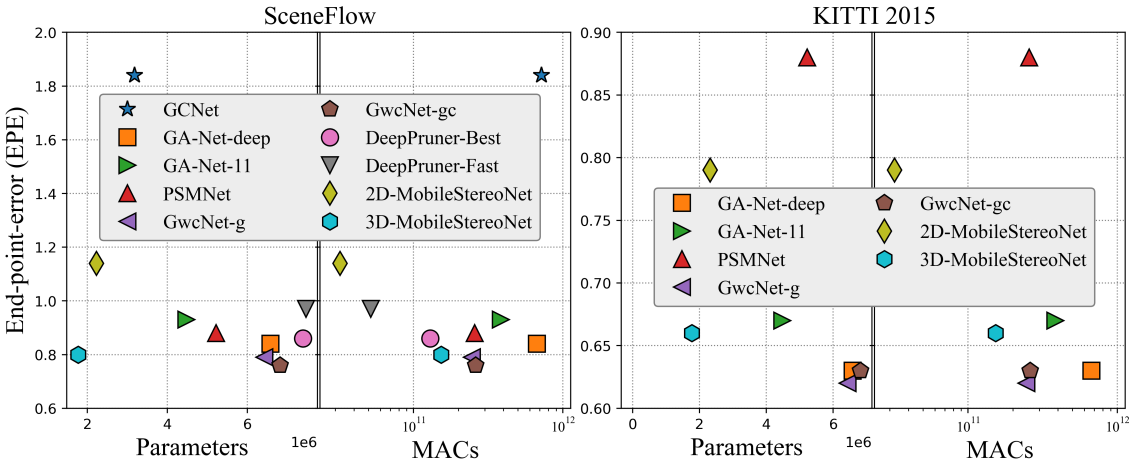


Figure 4.2: Performance vs computation cost on the SceneFlow test set (*Left*) and the KITTI 2015 validation set (*Right*): Metrics are EPE, number of parameters ($\times 10^6$), and number of operations (MACs in log scale). For all, the lower is better. By using a new parameterized cost volume, $2\mathcal{D}$-MobileStereoNet shows closer performance to $3\mathcal{D}$ models with the least MACs. $3\mathcal{D}$-MobileStereoNet obtains competitive accuracy with the least number of parameters.

Our main contributions can be summarized as follows:

i. We propose two lightweight models ($2\mathcal{D}$ and $3\mathcal{D}$) for stereo matching, employing MobileNet blocks while maintaining accuracy.
ii. We extend MobileNet blocks from their original $2\mathcal{D}$ convolutions to $3\mathcal{D}$, demonstrating their effectiveness in reducing computational load when processing $3\mathcal{D}$ data.
iii. We introduce a learnable cost volume module for the $2\mathcal{D}$ model, ensuring comparable accuracy with over-parameterized $3\mathcal{D}$ models.
iv. We conduct extensive experiments to analyze the accuracy/complexity trade-off across various design choices. The insights gained from these experiments can be applied to similar $2\mathcal{D}$/$3\mathcal{D}$ networks to reduce complexity.

4.2 Related work

Stereo vision is one of the most 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 (Žbontar and LeCun, 2016; Seki and Pollefeys, 2017; Batsos *et al.*, 2018), and approaches that formulate the whole process in an end-to-end scheme (Mayer *et al.*, 2016; Kendall *et al.*, 2017; Chang and Chen, 2018; Guo *et al.*, 2019a; Zhang *et al.*, 2019). Most of the end-to-end methods are developed based on $3\mathcal{D}$ 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 $3\mathcal{D}$ architectures. In (Yee and Chakrabarti, 2020), 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 (Duggal *et al.*, 2019) mitigates the computational complexity of $3\mathcal{D}$ convolutional layers by calculating the matching cost for a subset of possible disparity values. Recently, (Rahim *et al.*, 2021) proposed to use feature-wise and feature-disparity-wise separable convolutions for optimizing the $3\mathcal{D}$ stereo models. On the whole, according to the results in (Yee and Chakrabarti, 2020), the $2\mathcal{D}$ architectures can make a better trade-off between accuracy and speed.

Other works mainly focus on reducing the complexity of $3\mathcal{D}$ convolutional layers for other $3\mathcal{D}$ vision tasks. Qiu *et al* (Qiu *et al.*, 2017) developed pseudo-convolutions that decouple a $3\mathcal{D}$ convolution into $3\mathcal{D}$ convolutions equivalent to a $2\mathcal{D}$ convolution and a light $3\mathcal{D}$ convolution, which aggregates information only across the third domain. The network is demonstrated on video classification. In (Ye *et al.*, 2019), authors made use of $3\mathcal{D}$ depth-wise convolutions for $3\mathcal{D}$ reconstruction. Recently, (Feichtenhofer, 2020) proposed progressively forward expanding a $2\mathcal{D}$ 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 $3\mathcal{D}$ cost volume (Dosovitskiy *et al.*, 2015; Luo *et al.*, 2016b; Mayer *et al.*, 2016; Žbontar and LeCun, 2016; Ilg *et al.*, 2018). Later, Kendall *et al* (Kendall *et al.*, 2017) proposed to concatenate unary features to make a $4\mathcal{D}$ cost, requiring $3\mathcal{D}$ convolutions in the following. After that, this approach was mainly adapted for $3\mathcal{D}$ models with some modifications to enhance the accuracy, *e.g*. variance-based (Rao *et al.*, 2020), group-wise correlation (Guo *et al.*, 2019a) and pyramid (Wu *et al.*, 2019) cost volume.

4.3 Methodology

As a first step, we reformulate two common light blocks (Howard *et al.*, 2017; Sandler *et al.*, 2018) to raise them from $2\mathcal{D}$ convolutions to $3\mathcal{D}$ for the application of stereo matching. We also analyze their computation cost *w.r.t*. the standard $2\mathcal{D}$/$3\mathcal{D}$ convolution counterparts. The computation 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.

4.3.1 Light Blocks Replacing $2\mathcal{D}$/$3\mathcal{D}$ Convolutions

As a pioneer work, MobileNet-V1 (Howard *et al.*, 2017) employs depth-wise 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 (Sandler *et al.*, 2018) 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. 4.3, MobileNet-V1 and MobileNet-V2 blocks (shortened to $v1$ and $v2$ hereon, for simplicity) are shown.

Raising MobileNet Blocks to $3\mathcal{D}$ Stereo Network

Originally, $v1$ and $v2$ blocks are designed to replace $2\mathcal{D}$ convolutions and are proved mainly for sparse prediction tasks, like image classification. Still, many other computer vision problems require $3\mathcal{D}$ convolutions to operate on $4\mathcal{D}$ data, *e.g*. tasks with temporal input besides the spatial data. Likewise, dense disparity estimation is such a topic, which

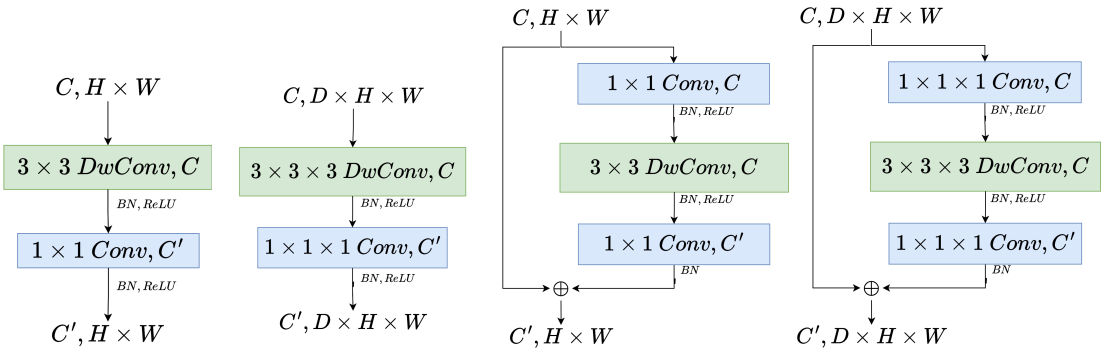


Figure 4.3: *Left*: MobileNet-V1 block and its extension to $3\mathcal{D}$, *Right*: MobileNet-V2 block and its extension to $3\mathcal{D}$.

explores the space for the 3rd dimension of the scene. This outlook of using $3\mathcal{D}$ convolutions for stereo matching has emerged recently, where they are utilized for processing $4\mathcal{D}$ cost data. Hence, we take $v1$ and $v2$ blocks to their $3\mathcal{D}$ counterparts for stereo vision application and show their necessity for light stereo vision models.

For this, just as in $2\mathcal{D}$ convolutions, we commit the depth-wise and point-wise convolutions in the channel (feature) dimension in $3\mathcal{D}$ convolutions. To be more precise, to raise the convolutions from $2\mathcal{D}$ to $3\mathcal{D}$, 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 $4\mathcal{D}$ data of cost volume (Kendall *et al.*, 2017; Wu *et al.*, 2019; Guo *et al.*, 2019a) 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 $2\mathcal{D}$ to $3\mathcal{D}$, considering the same size for the added dimension, *i.e.* if the $2\mathcal{D}$ kernel is 3×3, the $3\mathcal{D}$ kernel is $3 \times 3 \times 3$. Therefore, taking $v1$ and $v2$ blocks to $3\mathcal{D}$ is straightforward by applying depth-wise and point-wise convolutions in the channel dimension. Figure 4.3 displays the new $v1$ and $v2$ blocks raised to $3\mathcal{D}$.

In Tab. 4.1, we compute the cost of the standard $2\mathcal{D}$/$3\mathcal{D}$ convolutions and $2\mathcal{D}$/$3\mathcal{D}$ $v1$ 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 $2\mathcal{D}$ and $3\mathcal{D}$ versions, as expected. Moreover, exploiting $v1$ and $v2$ blocks in $3\mathcal{D}$ 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. 4.4. We consider the common cases in an hourglass module (*e.g.* in (Chang and Chen, 2018; Guo *et al.*, 2019a)) 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 $2\mathcal{D}$ 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 $3\mathcal{D}$ convolutional layers of $3\mathcal{D}$ networks.

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

Table 4.1: Computation cost of the standard convolutions vs MobileNet-V1 (v_1) and MobileNet-V2 (v_2) blocks in $2\mathcal{D}$ and $3\mathcal{D}$ counterparts. In MACs, $(D\times)$ belongs to $3\mathcal{D}$ types. We have ignored the computation cost of batch normalization, ReLU and residual connection of v_2. The example for the reduction factor (*w.r.t.* the standard convolution counterparts) is computed for $k = 3$, $C_{in} = 32$, $C_{out} = 64$, and $t = 2$.

4.3.2 Proposed Models

Here, we describe two end-to-end baselines ($2\mathcal{D}$ and $3\mathcal{D}$) and design MobileStereoNets in reliance on these two models and $2\mathcal{D}/3\mathcal{D}$ *v*1 and *v*2. 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 encoder-decoder (hourglass) is applied on top of the cost volume to estimate the disparity map.

$3\mathcal{D}$ Baseline

For this case, we adopt GwcNet-g (Guo *et al.*, 2019a) with only one hourglass (Fig. 4.5) as it is performing superior to other similar designs, *e.g.* GCNet (Kendall *et al.*,

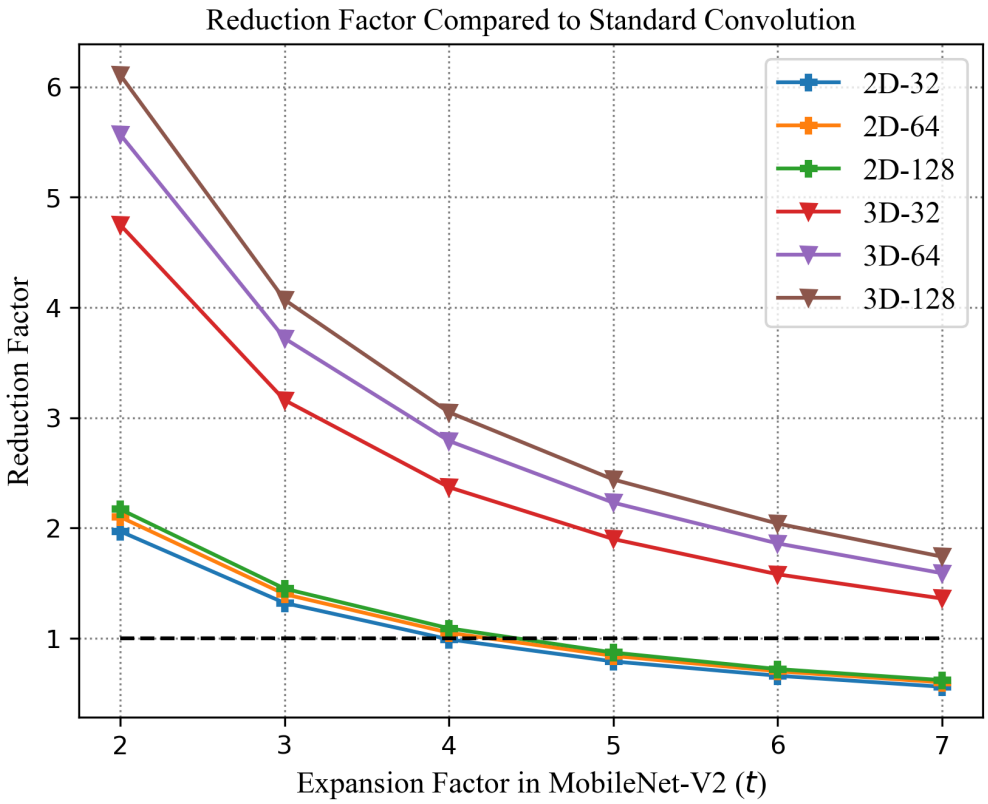


Figure 4.4: Reduction factor of $2\mathcal{D}/3\mathcal{D}$ *v*2 blocks with varying expansion factor (t) *w.r.t.* standard convolution counterparts. The right hand numbers indicate the channel numbers ($C_{in} = C_{out}$ and $k = 3$).

2017), PSMNet (Chang and Chen, 2018), and GA-Net (Zhang *et al.*, 2019). This baseline utilizes a ResNet-like backbone and an encoder-decoder with $3\mathcal{D}$ convolutions. Namely, a $4\mathcal{D}$ cost volume is constructed by group-wise correlation of unary features, requiring $3\mathcal{D}$ convolutions afterward. The encoder-decoder consists of an hourglass (Chang and Chen, 2018; Guo *et al.*, 2019a) outputting a downsampled disparity map, which after upsampling is compared against the ground-truth with a smooth-$L1$ loss function.

$2\mathcal{D}$ Baseline

In order to develop a much lighter stereo network, we modify the $3\mathcal{D}$ baseline such that it uses an encoder-decoder with $2\mathcal{D}$ convolutions (Fig. 4.5). This approach contrasts with the recent trend, where $3\mathcal{D}$ convolutions are deployed to add a feature dimension for disparity via a $4\mathcal{D}$ cost volume. With a ResNet-like backbone similar to the $3\mathcal{D}$ baseline, an input image with resolution $H \times W$ is turned into a feature of size $320 \times (H/4) \times$

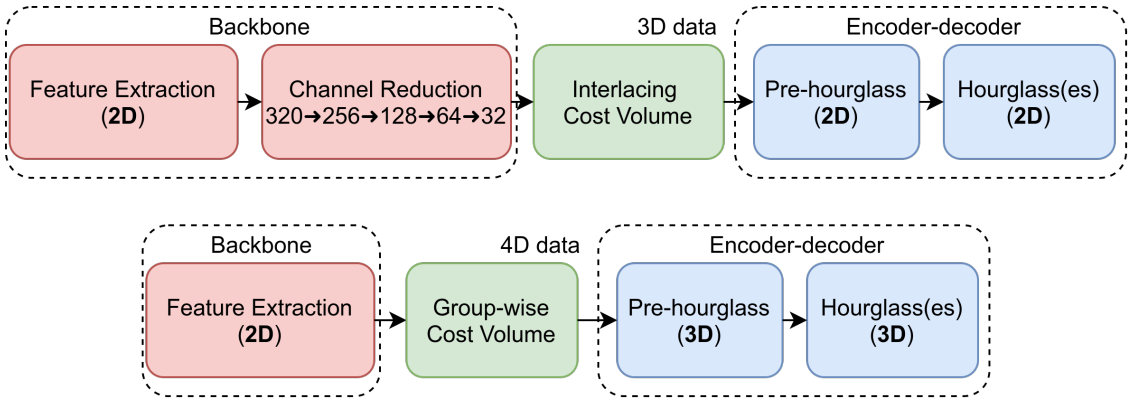


Figure 4.5: *Top* ($2\mathcal{D}$ Baseline): After feature extraction, data is reduced to 32 channels. A $3\mathcal{D}$ cost volume, $(d_{max}/4) \times (H/4) \times (W/4)$, is generated using the proposed interlacing cost volume construction and it is processed by $2\mathcal{D}$ convolutions. *Bottom* ($3\mathcal{D}$ Baseline): A $4\mathcal{D}$ cost volume, $40 \times (d_{max}/4) \times (H/4) \times (W/4)$, is computed using Gwc40 (Guo *et al.*, 2019a) and it is processed with $3\mathcal{D}$ convolutions.

$(W/4)$. 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 $3\mathcal{D}$ cost volume to retain the encoder-decoder with $2\mathcal{D}$ convolutions. Ignoring the feature dimension for disparity, the $3\mathcal{D}$ cost volume is of size $(d_{max}/4) \times (H/4) \times (W/4)$, where d_{max} is the maximum disparity level. Finally, the $3\mathcal{D}$ cost volume is taken into the encoder-decoder module after passing through two convolutions as a pre-hourglass module.

The detailed architectures of the $2\mathcal{D}$ and $3\mathcal{D}$ baseline models are displayed in Fig. 4.6. The numbers in the blocks indicate the output size of each particular layer/module. The feature extraction step is the same for the two models. The architecture of hourglass and its intraconnections are also similar, except that in the $2\mathcal{D}$ baseline, the convolutions are all in $2\mathcal{D}$ type, while there are $3\mathcal{D}$ convolutions in hourglass of the $3\mathcal{D}$ baseline. These two models differ in the cost volume construction and the channel reduction module as well.

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:

$$C_{3\mathcal{D}}(d,x,y) = G(f_L(x,y), f_R(x-d,y)), \tag{4.1}$$

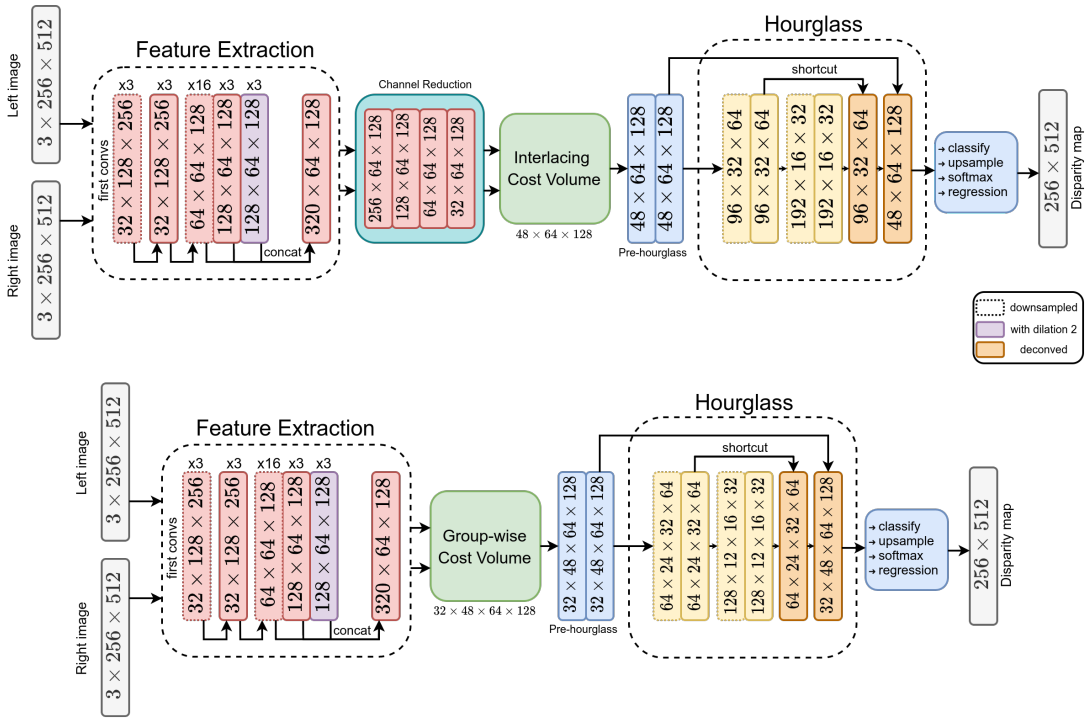


Figure 4.6: *Top*: $2\mathcal{D}$ baseline, *Bottom*: $3\mathcal{D}$ baseline. The numbers in the blocks indicate the output size of each particular layer/module.

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 $3\mathcal{D}$ data, *i.e.* $f(x,y) \rightarrow f(.,x,y)$, DispNetC (Mayer *et al.*, 2016) proposed to use a correlation layer (dot product) to merge these data by $f_L(.,x,y) \odot f_R(.,x-d,y)$. Later, Kendall *et al.* (2017) introduced a cost volume as *4D data*, $C_{4\mathcal{D}}(d,.,x,y)$, by concatenating the unary features.

These $3\mathcal{D}$ or $4\mathcal{D}$ 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 $2\mathcal{D}$ convolutions (and not $3\mathcal{D}$, 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

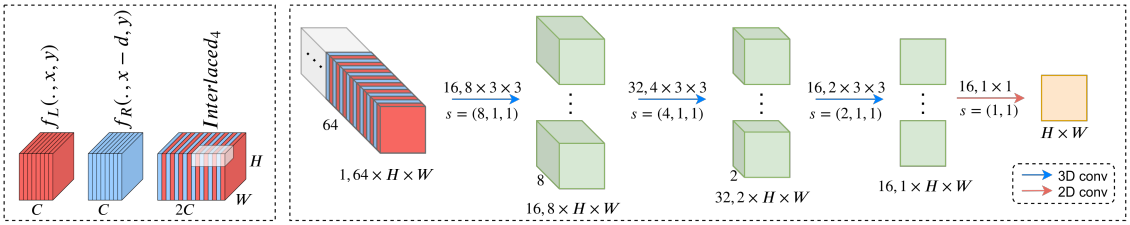


Figure 4.7: *Left*: Interlacing cost volume construction at a particular disparity from the left (red) and right (blue) unary features. The kernels of the first layer take a group of non-overlapping interlaced features (here, four channels from each unary feature, $Interlaced_4$). *Right*: Processing 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.

channels, $2C \times H \times W$ (Fig. 4.7). After unsqueezing the output and raising it to $4\mathcal{D}$ data ($1 \times 2C \times H \times W$), a $3\mathcal{D}$ convolutional layer is applied such that the $3\mathcal{D}$ kernels cover a specific number of left/right feature pairs. That is, a kernel of size $2i \times 3 \times 3$ covers i channels 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. 4.7) are also $3\mathcal{D}$ convolutions 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 $m \times 1 \times 1$, showing that they are covering non-overlapping channels. Finally, we convert the data to a single channel and pass through one $2\mathcal{D}$ convolutional layer. Note that similar to other methods for the cost volume, the spatial resolution is unchanged. We can write the general formula for a certain disparity level as Eg. 4.2. In Sec. 4.4, we show that the inclusion of learnable weights in stereo network contributes to better aggregation of the left/right features.

$$C_{3\mathcal{D}}(d,x,y) = \text{Interlace}\{f_L(.,x,y), f_R(.,x-d,y)\} \tag{4.2}$$

MobileStereoNets

In our baselines, the feature extraction and channel reduction are essentially processed with $2\mathcal{D}$ convolutions. On the other hand, the hourglass employs $2\mathcal{D}$ or $3\mathcal{D}$ convolutions depending on the constructed cost volume. In order to obtain lighter networks, we replace these components with $2\mathcal{D}$ and $3\mathcal{D}$ 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:

- Can we replace different modules in the $2\mathcal{D}$/$3\mathcal{D}$ baselines with $2\mathcal{D}$/$3\mathcal{D}$ $v1$ and $v2$ to achieve lighter stereo networks and keep the error rates low?
- If so, which modules should be replaced with them for a better compromise?
- Which block type performs better in terms of accuracy and computational load?

Our experiments (*c.f*. Sec. 4.4) lead to our MobileStereoNets by modifying the baselines as follows:

- **First Convolutions:** Each of the three initial 3×3 convolutions are replaced with one *v*2. 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 *v*1 keeps performance competitive while reducing the computational complexity significantly.
- **Channel Reduction:** In the $2\mathcal{D}$ baseline, we keep this module, *i.e*. four 1×1 convolutions, unchanged with standard convolutions as replacing them with lighter blocks deteriorates the performance.
- **Pre-hourglass:** In the $2\mathcal{D}$ baseline, we replace the two 3×3 convolutions in both of the blocks with *v*2. For $3\mathcal{D}$-MobileStereoNet, we use our extension of *v*2 to $3\mathcal{D}$ instead of $3\mathcal{D}$ convolutions. In both models, we choose the expansion factor as $t = 3$.
- **Hourglass:** We employ a stack of three hourglasses for both $2\mathcal{D}$ and $3\mathcal{D}$ models. While the $3\mathcal{D}$ network uses the same channel dimension of 32 as GwcNet (Guo *et al.*, 2019a), the hourglass width is increased to 48 in the $2\mathcal{D}$ model. $2\mathcal{D}$-MobileStereoNet uses *v*2 instead of $2\mathcal{D}$ convolutions. In $3\mathcal{D}$-MobileStereoNet, we once again swap the $3\mathcal{D}$ convolutions for our extension of *v*2 to $3\mathcal{D}$. For both models, the expansion factor is $t = 2$.

4.4 Experimental Results and Discussions

Here, we first present the implementation details and evaluate the performance of the proposed interlaced cost volume. Then, extensive experiments for taking the $2\mathcal{D}$/$3\mathcal{D}$ *v*1 and *v*2 blocks into the baselines are elaborated to show the path taken to reach the final architectures. We also present the results of incorporating light blocks in other modules of the network and modular complexity analysis. Finally, we compare our developed networks with other methods. Note that d_{max} is 192 in all cases.

To compare different design approaches, we use the SceneFlow "final pass" dataset (Mayer *et al.*, 2016), which offers 35,454 training and 4,370 test samples at a 540x960 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*. 3-*px* and D1, which are percentages of the outliers with disparity errors larger than 3-*px* and $\max(3 - px, 0.05 \times \text{ground-truth})$, respectively.

4.4.1 Implementation Details

We used PyTorch for implementation and conducting experiments. All the trainings were executed on 4 × NVIDIA GeForce GTX 1080 Ti. We adapted the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. On the SceneFlow dataset, the networks were trained for 20 epochs, starting with a learning rate of 0.001. The learning rate was halved after epoch 10, 12, 14, and 16. The best model was selected based on the least EPE value. In the experiments on the KITTI 2015 validation set, we finetuned the best SceneFlow model for 400 epochs, reducing the initial learning rate 0.001 by a factor of 10 after 200 epochs. To submit the results to the KITTI 2015 benchmark, we finetuned starting from a SceneFlow checkpoint showing the best generalization performance from the SceneFlow to the KITTI 2015 images. For the $3\mathcal{D}$-MobileStereoNet, we used a batch size of 4, and for $2\mathcal{D}$-MobileStereoNet, the batch size was 8.

4.4.2 Cost Volume Construction

To verify the performance of the interlacing cost volume, we replaced the corresponding module in $2\mathcal{D}$ baseline (Fig. 4.5) with correlation, which is adopted by $2\mathcal{D}$ networks (Dosovitskiy *et al.*, 2015; Mayer *et al.*, 2016; Ilg *et al.*, 2018). Table 4.2 shows the evaluation of this baseline 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. 4.7), and hence, we consider this case for further experiments in the $2\mathcal{D}$ baseline.

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

4.4.3 Effect of Incorporating MobileNet Blocks

In this subsection, we incorporate v_1 and v_2 blocks (either $2\mathcal{D}$ or $3\mathcal{D}$, depending on the type of convolutional module) in various components of the $2\mathcal{D}$ and $3\mathcal{D}$ 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 $2\mathcal{D}$ and $3\mathcal{D}$ models are shown in Table 4.3. The best model is selected according to the least EPE obtained in 20 epochs. Also, the input

Method	EPE(px) ↓	D1(%) ↓	3-px (%) ↓
$Concatenation$	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 4.2: Performance evaluation on SceneFlow test set for the proposed $3\mathcal{D}$ cost volume: The costs created by $Interlaced_i$ contribute to lower error rates.

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 $2\mathcal{D}$ and $3\mathcal{D}$ 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 $2\mathcal{D}$ baseline, this combination results in the least EPE. We consider this combination for both $2\mathcal{D}$ and $3\mathcal{D}$ 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 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 $2\mathcal{D}$ baseline, both the complexity and the error are reduced. We also tried replacing other modules with MobileNet blocks, *i.e.* the channel reduction module and the convolutions in interlacing cost volume construction in the $2\mathcal{D}$ baseline. However, since these replacements deteriorate the learning capability of the network, they are kept unchanged with standard convolutions.

4.4.4 Incorporating Light Blocks in other Modules

In order to further reduce the complexity, the first convolutions in the feature extraction and the pre-hourglass convolutions (*c.f.* Fig. 4.6) are replaced with MobileNet-V2 (v_2). The experimental results are reported in Tables 4.4 and 4.5. Note that the first convolutions are of the $2\mathcal{D}$ type for both $2\mathcal{D}$ and $3\mathcal{D}$ baselines; however, the pre-hourglass comes in $2\mathcal{D}$ or $3\mathcal{D}$ convolutions depending on the baseline. We can observe that in $2\mathcal{D}$-MobileStereoNet, when the two convolution based modules are replaced with $v1$ and $v2$ variants, the network obtains the least EPE. In $3\mathcal{D}$-MobileStereoNet, this combination yields slightly higher EPE. However, due to the nice reduction in the computation cost,

$FE_{2\mathcal{D}}$	$HG_{2\mathcal{D}}$	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) $2\mathcal{D}$ models: $3\mathcal{D}$ cost volume using $Interlaced_4$ method

$FE_{2\mathcal{D}}$	$HG_{3\mathcal{D}}$	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) $3\mathcal{D}$ models: $4\mathcal{D}$ cost volume using Gwc40 (Guo *et al.*, 2019a)

Table 4.3: Performance evaluation on SceneFlow test set for variants of (a) $2\mathcal{D}$ and (b) $3\mathcal{D}$ baselines with $1 \times HG$. "FE" and "HG" stand for feature extraction and hourglass.

we consider the same design choice for the $3\mathcal{D}$ network. It is noteworthy that we have examined MobileNet-V1 (v_1) for these modules as well. However, as it deteriorates the performance, we ignore v_1 for these modules, albeit it shows much decrease in the cost.

4.4.5 Complexity Analysis

Table 4.6 shows the computation cost of the main modules, *i.e*. feature extraction and encoder-decoder, in baselines (with standard convolutions) and in MobileStereoNets. Note that feature extraction is the same in $2\mathcal{D}$ and $3\mathcal{D}$ models. We see our design choice for feature extraction is significantly reducing the complexity both in operation (from

first-conv$_{2\mathcal{D}}$	pre-HG$_{2\mathcal{D}}$	EPE(px) ↓	MACs(G) ↓	Params(M) ↓
conv.	conv.	1.50	30.33	1.21
conv.	v_2	1.41	30.00	1.16
v_2	conv.	1.54	29.75	1.20
v_2	v_2	**1.40**	**29.42**	**1.15**

Table 4.4: Performance evaluation for the selected variant of $2\mathcal{D}$ baseline (FE$_{2\mathcal{D}}$:v_1, HG$_{2\mathcal{D}}$:v_2) from Table 4.3a, when replacing other components with v_2 block ($t = 2$).

first-conv$_{2\mathcal{D}}$	pre-HG$_{3\mathcal{D}}$	EPE(px) ↓	MACs(G) ↓	Params(M) ↓
conv.	conv.	**0.99**	105.01	0.98
conv.	v_2	1.01	69.44	0.89
v_2	conv.	**0.99**	104.44	0.97
v_2	v_2	1.01	**68.86**	**0.88**

Table 4.5: Performance evaluation for the selected variant of $3\mathcal{D}$ baseline (FE$_{2\mathcal{D}}$:v_1, HG$_{3\mathcal{D}}$:v_2) from Table 4.3b, when replacing other components with v_2 block ($t = 2$).

52.07 to 7.84 GigaMACs) and in parameters (from 7.84 to only 0.39 million). We also observe that the cost of the encoder-decoder modules, either in $2\mathcal{D}$ or $3\mathcal{D}$, is reduced in lighter networks in both number of operations and parameters. Evidently, the major bottleneck for the $3\mathcal{D}$ models is the encoder-decoder with $3\mathcal{D}$ convolutions.

	Baselines		MobileStereoNets	
	MACs(G)	Params(M)	MACs(G)	Params(M)
Feature Extraction	52.07	2.95	7.84	0.39
Encoder-decoder$_{2\mathcal{D}}$ in $2\mathcal{D}$-MobileStereoNet	4.38	2.61	3.92	1.64
Encoder-decoder$_{3\mathcal{D}}$ in $3\mathcal{D}$-MobileStereoNet	167.51	3.45	128.73	1.34

Table 4.6: Analyzing the computation cost in terms of MACs and number of parameters for the main modules.

4.4.6 Quantitative and Qualitative Results

The discussed experiments support our design choice for the final $2\mathcal{D}$ and $3\mathcal{D}$ models, *i.e.* $2\mathcal{D}$-MobileStereoNet and $3\mathcal{D}$-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.

Method	EPE(px)	Params(M)	Red. Params
DispNet-C (Ilg *et al.*, 2018)	1.67	38.00	17.0x
CRL (Pang *et al.*, 2017)	1.60	78.77	35.3x
AutoDispNet-C (Saikia *et al.*, 2019)	1.51	37.00	16.6x
iResNet (Liang *et al.*, 2018)	1.40	43.11	19.3x
$2\mathcal{D}$-MobileStereoNet	**1.14**	**2.23**	-

(a) $2\mathcal{D}$ models

Method	EPE(px)	MACs(G)	Params(M)	Red. MACs	Red. Params
GCNet (Kendall *et al.*, 2017)	1.84	718.01	3.18	4.7x	1.8x
PSMNet (Chang and Chen, 2018)	0.88	256.66	5.22	1.7x	2.9x
GA-Net-deep (Zhang *et al.*, 2019)	0.84	670.25	6.58	4.4x	3.7x
GA-Net-11 (Zhang *et al.*, 2019)	0.93	383.42	4.48	2.5x	2.5x
Gwc40-Cat24-Base (Guo *et al.*, 2019a)	1.12	169.42	4.60	1.1x	2.6x
GwcNet-gc (Guo *et al.*, 2019a)	**0.76**	260.49	6.82	1.7x	3.9x
GwcNet-g (Guo *et al.*, 2019a)	0.79	246.27	6.43	1.6x	3.6x
DeepPruner-Best (Duggal *et al.*, 2019)	0.86	129.23	7.39	0.8x	4.2x
DeepPruner-Fast (Duggal *et al.*, 2019)	0.97	**51.83**	7.47	0.3x	4.2x
$3\mathcal{D}$-MobileStereoNet	0.80	153.14	**1.77**	-	-

(b) $3\mathcal{D}$ models

Table 4.7: Comparison on SceneFlow test set for (a) $2\mathcal{D}$ and (b) $3\mathcal{D}$ models. "Red." indicates the reduction factor of our models compared to other methods.

SceneFlow Dataset

The evaluation of the proposed models on SceneFlow are presented in Tables 4.7a and 4.7b. Note that $2\mathcal{D}$-MobileStereoNet has more parameters than $3\mathcal{D}$-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 $2\mathcal{D}$ models, a lower error is achieved by $2\mathcal{D}$-MobileStereoNet with 16.6x fewer parameters. Moreover, we observe that in $3\mathcal{D}$-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, $3\mathcal{D}$-MobileStereoNet uses 1.7x fewer parameters (in millions) and 3.9x fewer GigaMACs. Note that although DeepPruner-Fast (Duggal *et al.*, 2019) obtains the least number of operations, it is still over-parametrized and this causes issues when finetuning on smaller datasets like KITTI (*c.f.* Table 4.9). Fig. 4.8 depicts qualitative results on the SceneFlow dataset.

KITTI 2015 Dataset

This dataset consists of images of real-world driving scenarios (Menze and Geiger, 2015), 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 160/40 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. 4.8, $2\mathcal{D}$-MobileStereoNet attains comparable results to PSMNet, which is a $3\mathcal{D}$ model, with much less computational load (2.3x/8x fewer parameters/operations). Also, $3\mathcal{D}$-MobileStereoNet is outperforming PSMNet and GA-Net-11, and is competitive with GA-Net-deep. Compared to GwcNet-g, $3\mathcal{D}$-MobileStereoNet is lighter with 3.6x/1.6x fewer parameters/operations.

A qualitative comparison on KITTI 2015 validation set is shown in Fig. 4.9.

Method	EPE(px)	D1(%)	3-px (%)	MACs(G)	Params(M)
PSMNet (Chang and Chen, 2018)	0.88	2.00	2.10	256.66	5.22
GA-Net-deep (Zhang *et al.*, 2019)	0.63	1.61	1.67	670.25	6.58
GA-Net-11 (Zhang *et al.*, 2019)	0.67	1.92	2.01	383.42	4.48
GwcNet-gc (Guo *et al.*, 2019a)	0.63	1.55	1.60	260.49	6.82
GwcNet-g (Guo *et al.*, 2019a)	**0.62**	**1.49**	**1.53**	246.27	6.43
$2\mathcal{D}$-MobileStereoNet	0.79	2.53	2.67	**32.2**	2.32
$3\mathcal{D}$-MobileStereoNet	0.66	1.59	1.69	153.14	**1.77**

Table 4.8: Comparison on KITTI 2015 validation set.

From Fig. 4.9, once again, we can verify that $2\mathcal{D}$-MobileStereoNet shows close performance to $3\mathcal{D}$ models with the least number of operations. Also, $3\mathcal{D}$-MobileStereoNet obtains competitive or better accuracy with the least number of parameters among other methods.

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 4.9 shows that $2\mathcal{D}$-MobileStereoNet is surpassing GCNet (a $3\mathcal{D}$ model) with 27%/95% fewer parameters/operations. Also, we can verify that $3\mathcal{D}$-MobileStereoNet shows superior performance when compared to

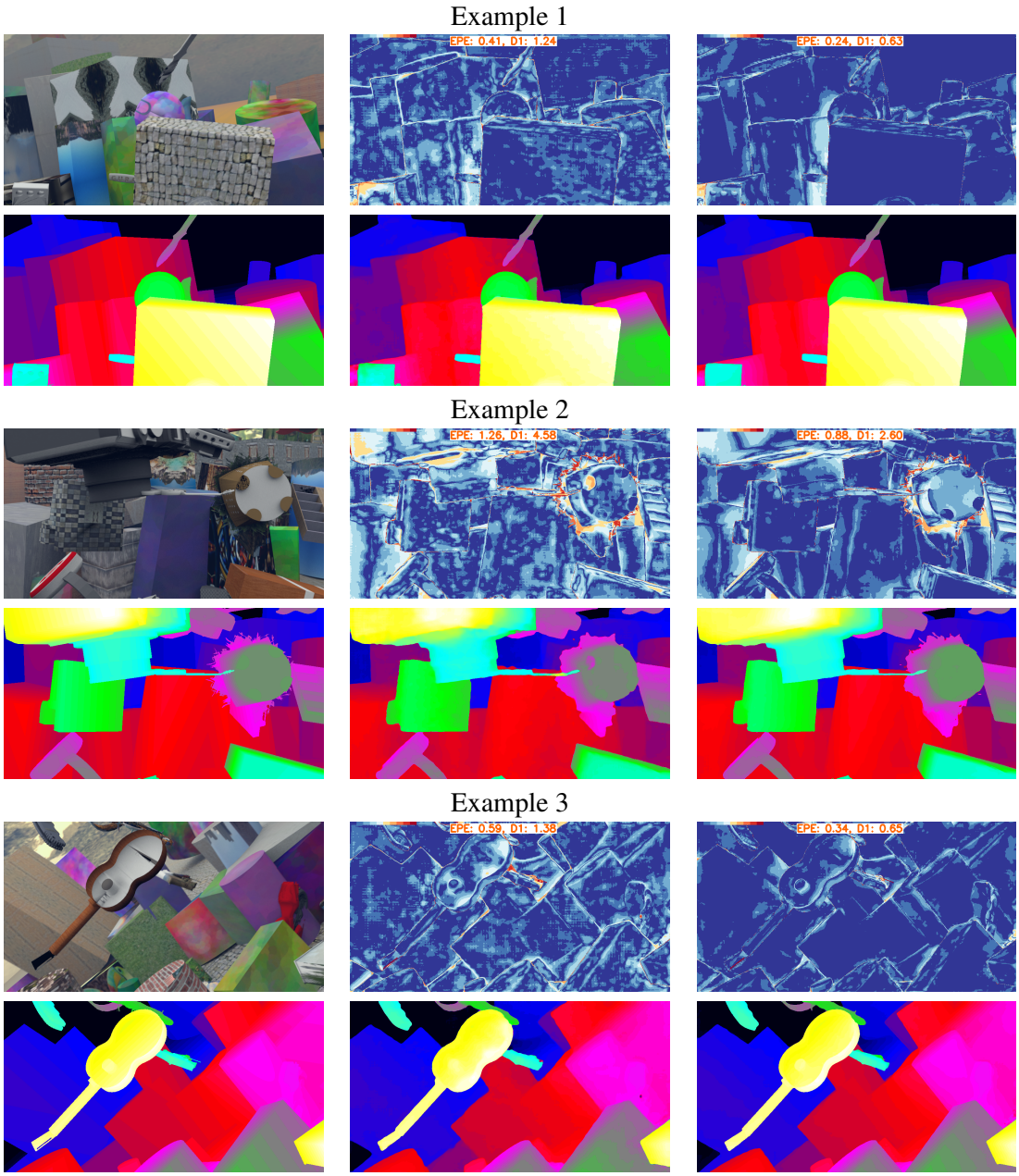


Figure 4.8: Qualitative performance on SceneFlow: Every two rows correspond to a test sample. In the left-most column, the samples and the ground-truth disparity maps are illustrated. The following two columns show the disparity and error maps (embedded with error values) estimated by 2$\mathcal{D}$-MobileStereoNet and 3$\mathcal{D}$-MobileStereoNet. Warmer colors in error maps denote higher errors.

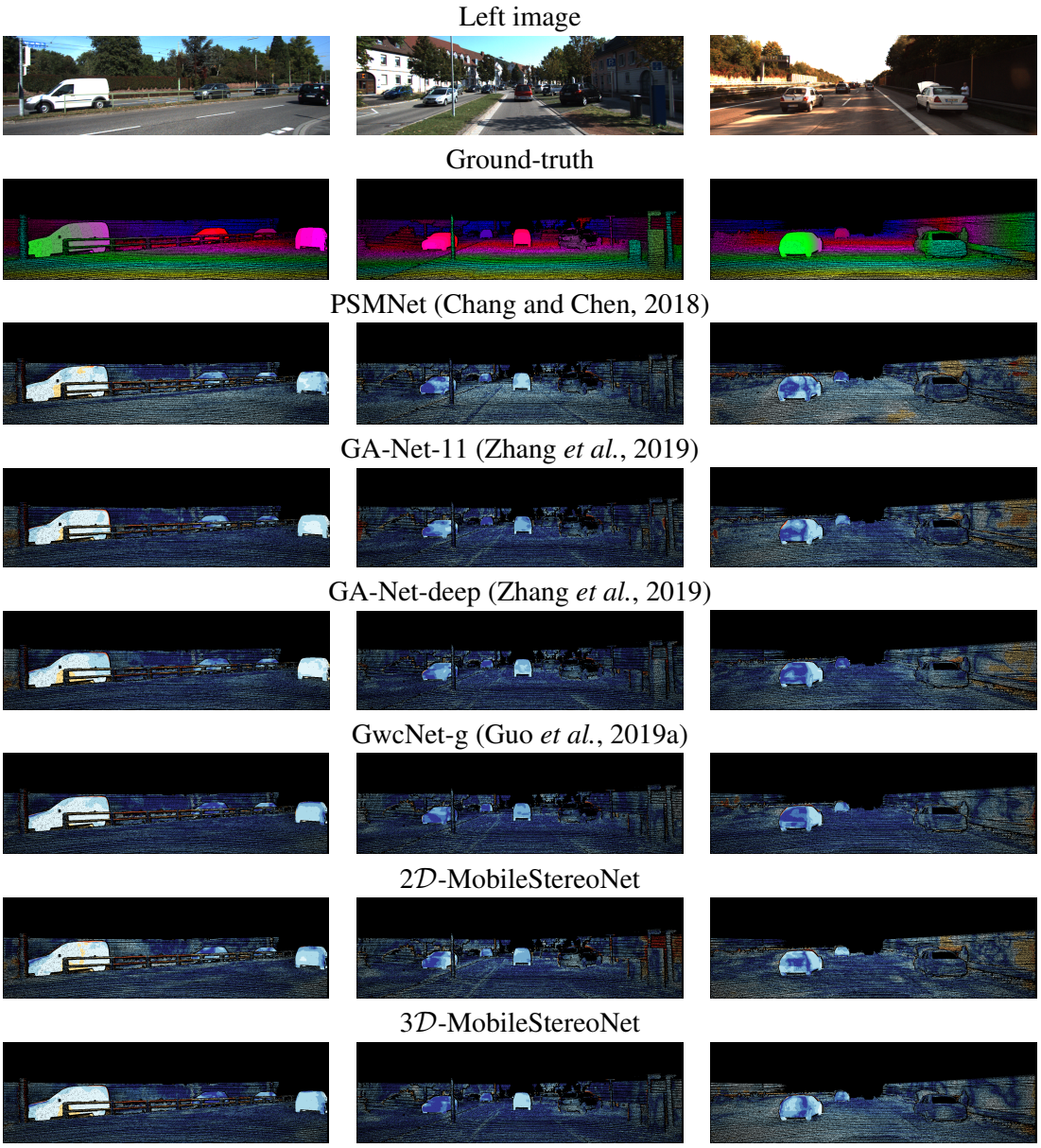


Figure 4.9: Qualitative performance on KITTI 2015 validation set: From top to bottom, the left image, the ground-truth disparity map and the estimated disparity maps by PSMNet (Chang and Chen, 2018), GA-Net-11 (Zhang *et al.*, 2019), GA-Net-deep (Zhang *et al.*, 2019), GwcNet-g (Guo *et al.*, 2019a), 2$\mathcal{D}$-MobileStereoNet and 3$\mathcal{D}$-MobileStereoNet are illustrated. For a fair comparison, we trained all the models with a 160 training and 40 test images of the KITTI 2015 training set. Warmer colors in error maps denote higher errors.

Methods	All(%)			Noc(%)		
	$D1_{bg}$	$D1_{fg}$	$D1_{all}$	$D1_{bg}$	$D1_{fg}$	$D1_{all}$
MC-CNN (Žbontar and LeCun, 2016)	2.89	8.88	3.89	2.48	7.64	3.33
Fast DS-CS (Yee and Chakrabarti, 2020)	2.83	4.31	3.08	2.53	3.74	2.73
GCNet (Kendall *et al.*, 2017)	2.21	6.16	2.87	2.02	5.58	2.61
DeepPruner-Fast (Duggal *et al.*, 2019)	2.32	3.91	2.59	2.13	3.43	2.35
PSMNet (Chang and Chen, 2018)	1.86	4.62	2.32	1.71	4.31	2.14
AutoDispNet-CSS (Saikia *et al.*, 2019)	1.94	**3.37**	2.18	1.80	**2.98**	2.00
DeepPruner-Best (Duggal *et al.*, 2019)	1.87	3.56	2.15	1.71	3.18	1.95
GwcNet-g (Guo *et al.*, 2019a)	**1.74**	3.93	2.11	**1.61**	3.49	**1.92**
$2\mathcal{D}$-MobileStereoNet	2.49	4.53	2.83	2.29	3.81	2.54
$3\mathcal{D}$-MobileStereoNet	1.75	3.87	**2.10**	**1.61**	3.50	**1.92**

Table 4.9: Comparison on KITTI 2015 benchmark. $3\mathcal{D}$-MobileStereoNet requires 98% and 72% fewer parameters compared to AutoDispNet-CSS and GwcNet-g, respectively.

Method	Memory (MB)
GA-Net-deep (Zhang *et al.*, 2019)	26.4
GA-Net-11 (Zhang *et al.*, 2019)	18.0
GwcNet-gc (Guo *et al.*, 2019a)	27.9
GwcNet-g (Guo *et al.*, 2019a)	26.3
PSMNet (Chang and Chen, 2018)	21.1
$2\mathcal{D}$-MobileStereoNet	10.0
$3\mathcal{D}$-MobileStereoNet	8.0

Table 4.10: Comparison of model size. Our two proposed methods yield more compact models compared to other works.

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

$3\mathcal{D}$-MobileStereoNet visually achieves competitive or better results compared to other methods.

Model size

We also report the memory requirement of our models in Table 4.10. Both of the proposed methods show smaller memory sizes, which is promising for memory-constrained chips and their power consumption.

4.5 Conclusions

In this chapter we presented our efforts to tackle the challenge of high memory usage in stereo networks, specifically for embedded or mobile devices. We introduced two lightweight models, MobileStereoNet-$2\mathcal{D}$ and MobileStereoNet-$3\mathcal{D}$, with the primary goal of reducing the computational cost associated with these networks. By utilizing MobileNet blocks, we effectively reduced the parameters, operations, and overall model size. Furthermore, we addressed the accuracy aspect of the lightweight $2\mathcal{D}$ models by designing a novel cost volume construction method for the MobileStereoNet-$2\mathcal{D}$ model.

Overall, the results and insights presented in this chapter demonstrate the potential of MobileStereoNets in addressing the computational challenges of stereo networks. The advancements made in optimizing the computational requirements and balancing accuracy and complexity contribute to the development of efficient stereo matching solutions for embedded and mobile devices.

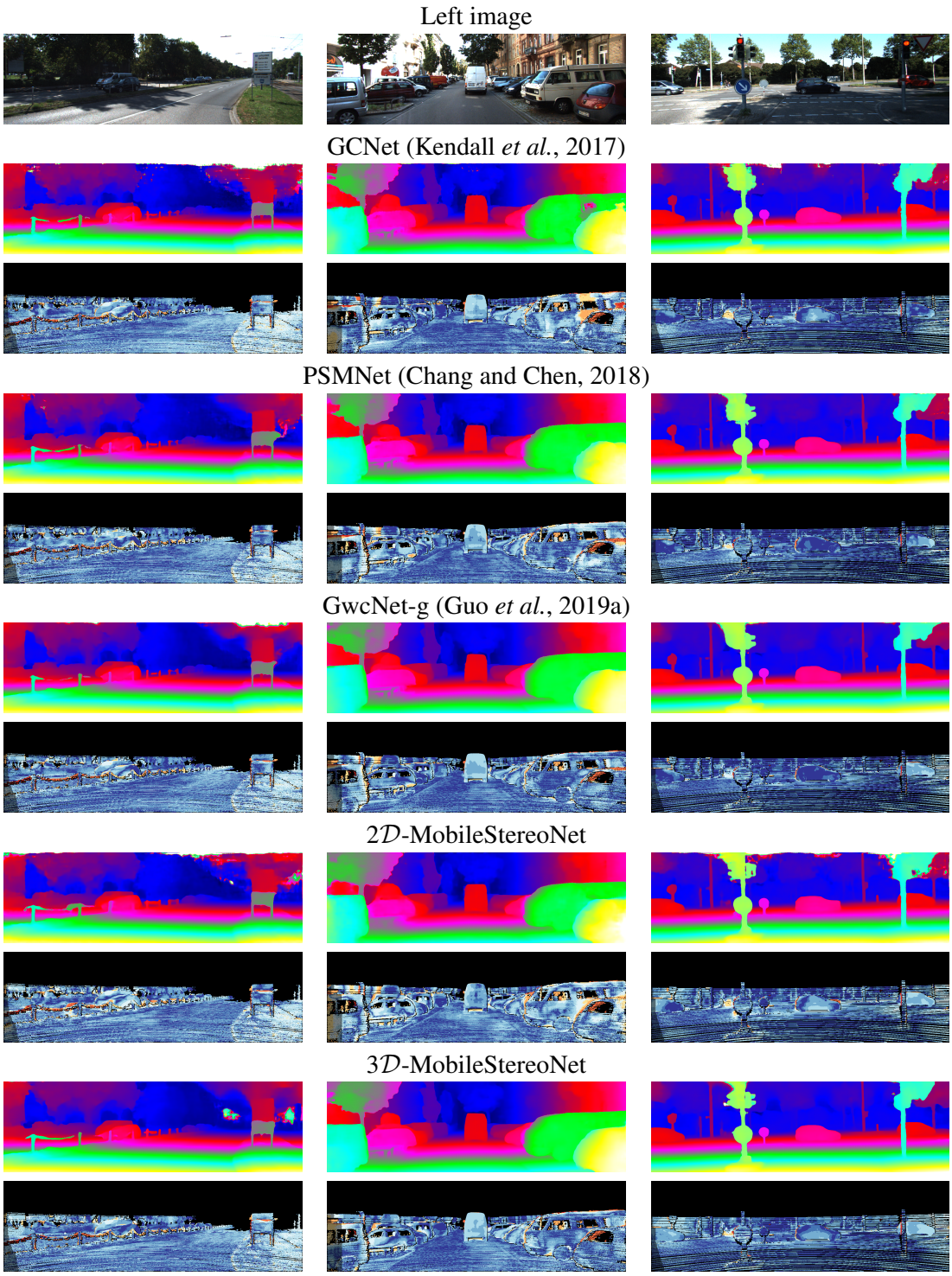


Figure 4.10: Qualitative performance (disparity images together with error maps) from KITTI 2015 benchmark. Warmer colors in error maps denote larger values.

Chapter 5

LeanStereo: A Leaner Backbone based Stereo Network

5.1 Introduction

In the pursuit of optimizing stereo networks, this chapter delves into the design of leaner backbones and explores different configurations of architectural modules to bridge the performance gap resulting from the integration of leaner backbones. Specifically, this chapter presents our work titled "LeanStereo: A Leaner Backbone based Stereo Network" by Rahim et al. (2023).

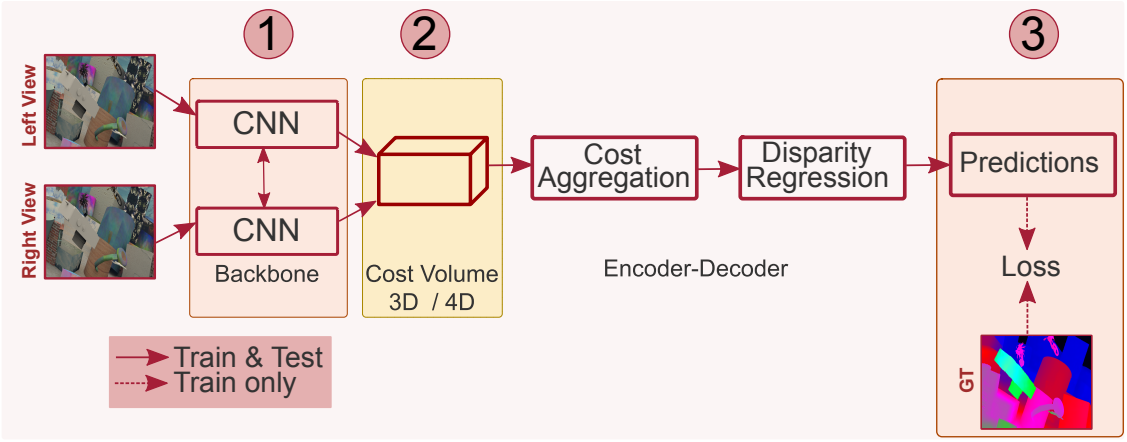


Figure 5.1: This chapter focuses on optimizing the backbone (module 1) of the network to reduce computational requirements. Additionally, we investigate the effectiveness of incorporating an attention-based cost volume (module 2) and a specialized loss function (LogL1 loss) (module 3) to enhance the overall performance of the network.

The key contribution of this work is the introduction of a leaner backbone for end-to-end stereo networks. This lean backbone, represented by module 1 in Figure 5.1, is designed to capture and leverage both structural and semantic information in input images. By specifically focusing on these aspects, the proposed lean backbone achieves a balance between computational efficiency and high-performance accuracy.

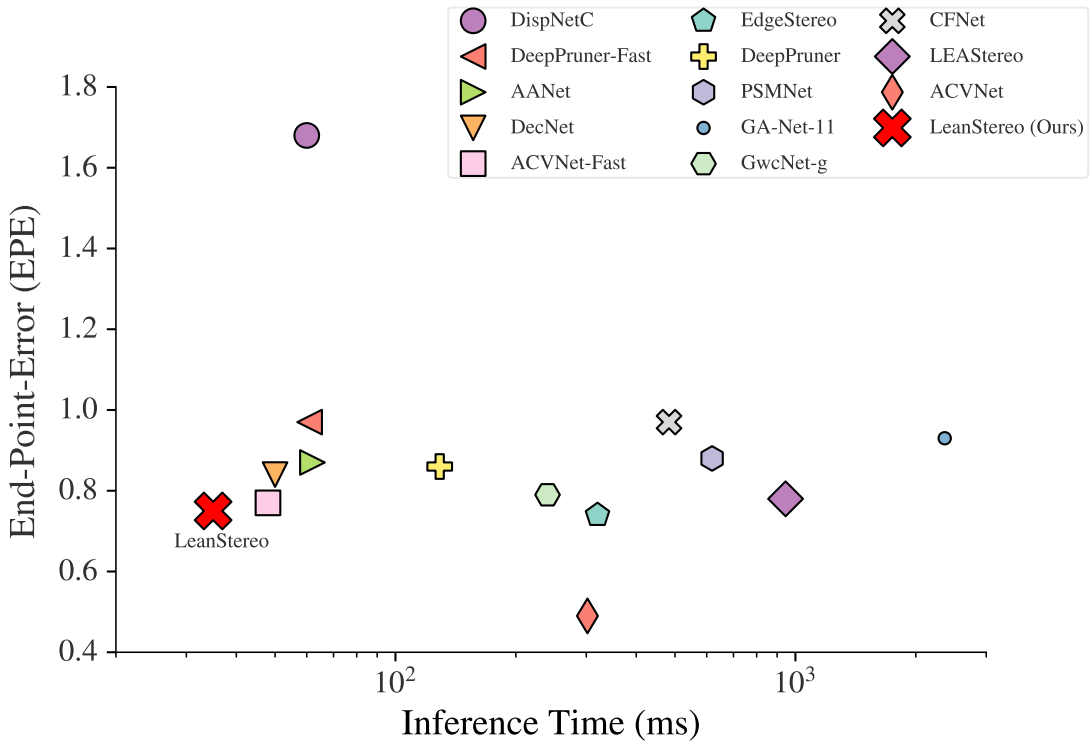


Figure 5.2: Comparison of LeanStereo with other state of the art methods. Our proposed method is real-time and have comparable performance with state of the art $3\mathcal{D}$ methods.

The inspiration for the lean backbone design comes from real-time backbones used in object detection (Sandler *et al.*, 2018; Wang *et al.*, 2018) and semantic segmentation (Yu *et al.*, 2021). The proposed backbone consists of two branches: a pixel-level detail capturing branch (Shallow Branch) and a semantic information capturing branch (Deep Branch). This two-branch architecture allows for separate optimization of low-level and high-level features, ensuring computational efficiency without sacrificing performance.

To further enhance the network's performance, attention-based cost volume (module 2) and a specialized loss function (LogL1 loss) (module 3) are incorporated. The attention mechanism allows the network to prioritize relevant features in the cost volume, resulting in improved disparity estimation. The specialized loss function effectively weighs the importance of different regions in the disparity map during training, promoting accurate depth estimation.

Overall, we make the following contributions:

i. We propose a two-branch-based lean backbone for $3\mathcal{D}$ stereo methods, explicitly capturing pixel-level and semantic-level information from stereo images.
ii. To address the performance gap resulting from a weaker backbone, we propose to use LogL1 loss, a specialized loss function that specifically targets the improve-

ment of smaller disparity errors. When combined with an attention-based cost volume, this approach not only reduces training times but also achieves similar performance compared to using a stronger backbone.

iii. We conduct a detailed empirical evaluation of various design choices, showcasing the significantly improved performance and computational efficiency of our network. For instance, our optimized network achieves inference times that are approximately 17×, 9×, 27×, and 14× faster than the PSMNet (Chang and Chen, 2018), ACVNet (Xu *et al.*, 2022), LEAStereo (Cheng *et al.*, 2020), and CFNet (Shen *et al.*, 2021) methods, respectively (refer to Figure 5.2). Notably, these metrics were obtained using identical settings and hardware for most of the open-source methods (refer to Section 5.4.5).

5.2 Related Work

Traditionally in the computer vision domain, improving the performance of the methods had been a pivot point; nowadays, the focus has started to towards real-time methods (Sandler *et al.*, 2018; Tan and Le, 2019; Yu *et al.*, 2021) as deep neural networks are becoming more and more resource/data hungry. Given this shift towards designing lightweight networks for other computer vision tasks, there is a need to explore the mechanisms to make the stereo network lightweight and real-time.

As discussed earlier, stereo methods can be categorized in two categories, *i.e.* $2\mathcal{D}$ and $3\mathcal{D}$ methods. In $2\mathcal{D}$ methods, we have a $3\mathcal{D}$ cost volume with dimensions $H \times W \times C$, where H is the height, W is the width and C is the number of features/channels of the volume. Given a $3\mathcal{D}$ cost volume, we need a cost aggregation and disparity regression network that contains $2\mathcal{D}$ convolution operations. In contrast, $3\mathcal{D}$ stereo methods have a $4\mathcal{D}$ cost volume of dimension $H \times W \times D \times C$, where D denotes the disparity dimension; therefore, we need $3\mathcal{D}$ convolution operations to aggregate the feature maps and regress disparities.

Although $2\mathcal{D}$ stereo methods (Mayer *et al.*, 2016; Liang *et al.*, 2018; Tonioni *et al.*, 2019; Yang *et al.*, 2018; Song *et al.*, 2019) are faster in comparison with $3\mathcal{D}$ stereo methods, they lack in performance. Therefore there is a need to see how can we benefit from performance of the $3\mathcal{D}$ (Kendall *et al.*, 2017; Chang and Chen, 2018; Zhang *et al.*, 2019; Wang *et al.*, 2019; Duggal *et al.*, 2019; Xu *et al.*, 2022; Guo *et al.*, 2019a) stereo methods whiling keep them light-weight. $3\mathcal{D}$ stereo methods became popular since GC-Net (Kendall *et al.*, 2017) exploited the geometrical information of stereo matching to build a costly $4\mathcal{D}$ volume that improved the overall performance of the stereo networks drastically.

Since then, a number of $3\mathcal{D}$ stereo methods have been proposed to improve the performance, speed or both for the $3\mathcal{D}$ stereo networks. GwcNet (Guo *et al.*, 2019a) builds on the work of (Kendall *et al.*, 2017) and introduces the group-wise correlation to build the cost volume that benefits from cost volume designs of both $2\mathcal{D}$ architectures (cor-

relation) and $3\mathcal{D}$ architectures (concatenation). PSMNet (Chang and Chen, 2018) uses spatial pyramid pooling layers to increase the receptive field for better performance.

AnyNet (Wang *et al.*, 2019) exploits the coarse-to-fine resolution strategies that had been shown successful for $2\mathcal{D}$ networks to reduce the computational complexity albeit at the cost of performance. DeepPruner (Duggal *et al.*, 2019) reduces the dimension of thecost volume step by narrowing down disparities range via a disparity estimation network. MobileStereoNet (Shamsafar *et al.*, 2022) and network with separable convolutions (Rahim *et al.*, 2021) have focused on reducing the computational requirements of the stereo networks by exploiting the MobileNet V1 and V2 modules of (Sandler *et al.*, 2018) . On the other hand, some recent methods have also focused solely on improving the overall performance of the stereo networks. GANet (Zhang *et al.*, 2019) incorporates the successful traditional method (Hirschmuller, 2007) in the form of a layer in an end-to-end network that improved the performance but at the cost of high inference time. ACVNet (Xu *et al.*, 2022) builts on GwcNet and improves its performance by deploying an attention-based cost volume. CFNet (Shen *et al.*, 2021) uses cascade and fusion based cost volume to improve the performance. Given these advancements on both performance and speed ends, we need to combine the benefits of these methods to get the best of both worlds.

5.3 Methodology

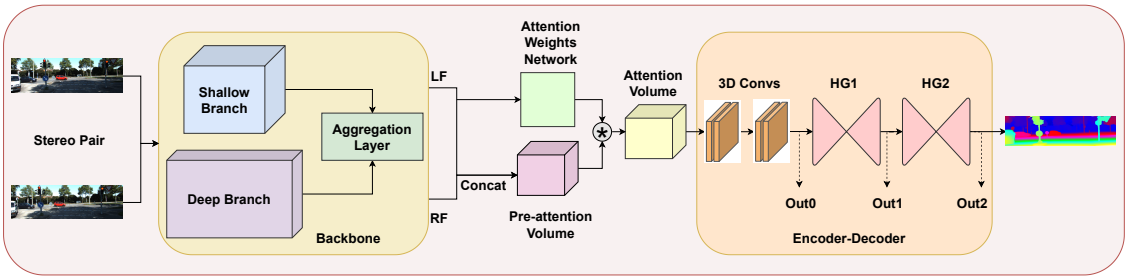


Figure 5.3: Proposed architecture. Here LF and RF represent the left and right image features extracted via the backbone network. 'Concat' means concatenating LF and LR. HG1 and HG2 represent the first and second hour-glass of the cost aggregation. Out0, Out1 and Out2 are the outputs from cost aggregation and used to regress the disparities.

We build our network based on the recent advancements and contributions in real-time image classification (Sandler *et al.*, 2018), object detection (Sandler *et al.*, 2018; Wang *et al.*, 2018), segmentation (Yu *et al.*, 2021), and performant stereo networks (Guo *et al.*, 2019a; Xu *et al.*, 2022) . Specifically, our architecture (*c.f*. Fig. 5.3) consists of a shared backbone that comprises of two branches – *c.f*. Sec. 5.3.1, an attention based cost volume and an hour-glass based encoder-decoder network to do cost aggregation and disparity regression – *c.f*. Sec. 5.3.2.

5.3.1 Backbone Network

Our backbone network has been inspired by MobileNet (Sandler *et al.*, 2018), PeleeNet (Wang *et al.*, 2018), and BiSeNet (Yu *et al.*, 2021), and aims at capturing both low-level features and semantic features present in the input image explicitly with limited computations. More precisely, the backbone network consists of two branches, named 'Shallow Branch'and 'Deep Branch'.

Shallow Branch:

This branch is responsible for capturing low-level image details. It consists of three levels and contains fewer layers compared to deep branch. The exact design for shallow branch is presented in Table 5.1. Levels 1, 2, and 3 are built by downscaling the features by factors of 1/2, 1/4, and 1/8, respectively, compared to the original input height (H) and width (W). Because this branch is responsible for capturing only fine details and does not require larger receptive fields, it is composed of only 8 convolutional layers.

Level	Input Size	Operation	Down Sample	Output Features	Output Size
1	$3 \times H \times W$	3x3 Conv	yes	64	$64 \times H/2 \times W/2$
1	$64 \times H/2 \times W/2$	3x3 Conv	no	64	$64 \times H/2 \times W/2$
2	$64 \times H/2 \times W/2$	3x3 Conv	yes	64	$64 \times H/4 \times W/4$
2	$64 \times H/4 \times W/4$	3x3 Conv	no	64	$64 \times H/4 \times W/4$
2	$64 \times H/4 \times W/4$	3x3 Conv	no	64	$64 \times H/4 \times W/4$
3	$64 \times H/4 \times W/4$	3x3 Conv	yes	128	$128 \times H/8 \times W/8$
3	$128 \times H/8 \times W/8$	3x3 Conv	no	128	$128 \times H/8 \times W/8$
3	$128 \times H/8 \times W/8$	3x3 Conv	no	128	$128 \times H/8 \times W/8$

Table 5.1: Architecture of the shallow branch. H and W represent the height and width of the original input. Level correlates with the dimensions of the feature maps *w.r.t.* the original input size. For example, for Level=2, the height of the feature maps is H/(2^{Level}).

Deep Branch:

This branch is responsible for capturing higher-level information from the input images. To achieve this, the deep branch uses far more layers compared to the shallow branch and thus has a higher receptive field. With more layers, the backbone gets heavier. This is avoided by using fewer feature maps per layer. Inside each layer, the MobileNet (Sandler *et al.*, 2018) module is used to reduce the number of computations and hence keep the deep branch light-weight. More precisely, separable convolutions (*i.e.* replacing one 3×3 convolution with a series of depth-wise separable and point-wise convolutions) are used to make the backbone deeper and increase the receptive field while limiting the

amount of computations. The exact design of this branch can be found in Table 5.2. Overall, this branch is built by adding a series of Gather-and-Expansion (GE) (Yu *et al.*, 2021) layers on top of a stem operation (Wang *et al.*, 2018). The stem operation reduces the input size by a factor of 4, whereas the series of GE further reduces the size of feature maps by a factor of 32. Finally, a global average pooling layer at the end is used to capture the overall global context of the input for better semantic learning. The final feature maps from this branch are at level 5 *i.e.* $H/32 \times W/32$.

Level	Input Size	Operation	Down Sample	Output Features	Output Size
1 & 2	$3 \times H \times W$	Stem	yes	64	$16 \times H/4 \times W/4$
3	$16 \times H/4 \times W/4$	GELayer2	yes	32	$32 \times H/8 \times W/8$
3	$32 \times H/8 \times W/8$	GELayer1	no	32	$32 \times H/8 \times W/8$
4	$32 \times H/8 \times W/8$	GELayer2	yes	64	$64 \times H/16 \times W/16$
4	$64 \times H/16 \times W/16$	GELayer1	no	64	$64 \times H/16 \times W/16$
5	$64 \times H/16 \times W/16$	GELayer2	yes	128	$128 \times H/32 \times W/32$
5	$128 \times H/32 \times W/32$	GELayer1	no	128	$128 \times H/32 \times W/32$
5	$128 \times H/32 \times W/32$	GELayer1	no	128	$128 \times H/32 \times W/32$
5	$128 \times H/32 \times W/32$	GELayer1	no	128	$128 \times H/32 \times W/32$
5	$128 \times H/32 \times W/32$	Global Avg Pooling	no	128	$128 \times H/32 \times W/32$
5	$128 \times H/32 \times W/32$	1 x 1 Conv	no	128	$128 \times H/32 \times W/32$
5	$128 \times H/32 \times W/32$	3 x 3 Conv	no	128	$128 \times H/32 \times W/32$

Table 5.2: Architecture of the deep branch.

Aggregation Layer:

The aggregation layer is responsible for fusing the information from both branches in such a way that both low- and high-level information extracted is utilized maximally. A straightforward method, like summation or a concatenation of the obtained feature maps cannot be used since the feature maps from both branches are on different scales. Therefore, like (Yu *et al.*, 2021), in aggregation layer features of one branch are used as attention weights to select the important features of the other branch. Selection of the features from the shallow branch is done in three steps. First a 3×3 convolution and an upsampling operation on the output of the deep branch is applied to increase the spatial resolution of feature maps. Next these upsampled feature maps are converted to attention weights by applying a sigmoid. Finally, these weights are element-wise multiplied by shallow branch features to select relevant features. Similarly, for selecting deep branch features, feature maps from shallow branch are first converted to attention weights after downsampling them to the same resolution of the deep branch. Next they are element-wise multiplied with deep branch output feature maps to select the impor-

tant semantic information. Finally, selected shallow and deep branch features are added element-wise together after downsampling shallow branch features. The output resolution of aggregation layer is at level 5, *i.e*. the input resolution is reduced by factor of 2^5. This helps to keep the later computations fast.

5.3.2 Cost Volume, Cost Aggregation and Disparity Regression

Cost Volume:

We construct an attention-based cost volume in the network, following the recent advancements (Xu *et al.*, 2022). Precisely, first a cost volume called 'pre-attention volume' is constructed by concatenating left and right image features extracted via the backbone network. In parallel, another cost volume is constructed by taking group-wise correlations between left and right features. This correlation volume is then split into three sub-groups and a series of convolution and merging operations are then applied to obtain attention weights. Finally the computed 'attention weights' are applied on 'pre-attention volume' to filter the important disparities and give an 'attention volume' as output for further processing – *c.f*. Fig. 5.3.

Cost Aggregation:

Cost aggregation network consists of four $3\mathcal{D}$ convolution layers and two hour-glass networks (Guo *et al.*, 2019a) stacked after one another. Each of the hour-glass networks has an encoder-decoder architecture with four $3\mathcal{D}$ convolutions (layers that reduce the input size by a factor of 2) and two $3\mathcal{D}$ transpose convolutions (to up-sample the feature maps).

Disparity Regression:

Finally outputs from the three different points in the cost-aggregation network are used to regress disparities. These output points are: (i) the output of $3\mathcal{D}$ convolutions (Out0); (ii) the output of the first hour-glass network (Out1); and the output (Out2) of the second hour glass network – *c.f*. Fig. 5.3. Each of these outputs is passed through a pair of $3\mathcal{D}$ convolutions (to obtain a volume with single feature map *i.e*. $1 \times D/4 \times W/8 \times W/8$) and an upsampling layer (to achieve $1 \times D \times H \times W$ output). Next a softmax in the disparity dimension is applied to convert these outputs to probabilities. Finally, *argsoftmax* in the disparity dimension is applied to get disparity values. Note that during training all the three different disparity values computed are used in the loss function calculation, whereas in inference only the Out2 is used to regress the final disparities – *c.f*. Fig. 5.3.

5.3.3 LogLoss

Stereo methods typically use the L2, L1 or its variants like SmoothL1 as loss function. Generally speaking, *i.e.*

$$\mathscr{L} = \frac{\sum_{i=1}^{H}\sum_{j=1}^{W} E(p_{ij}, q_{ij})}{H \times W} \tag{5.1}$$

where p is the predicted disparity, q is the ground-truth disparity, ij represents a pixel and E defines how the error is computed between the predicted and ground-truth disparities, *e.g.* in case of L1 loss $E(x,y) = |x-y|$.

If we consider the partial derivatives of the loss function with respect to network parameters θ, then we see that for L1 it is

$$\frac{\partial \mathscr{L}}{\partial \theta} = \frac{\partial E}{\partial x}\frac{\partial x}{\partial \theta}, \text{where} \tag{5.2}$$

$$\frac{\partial E}{\partial x} = \begin{cases} 1 & \text{if } x-y>0 \\ -1 & \text{else} \end{cases}$$

and for L2 it is

$$\frac{\partial E}{\partial x} = 2(x-y)$$

Thus these error functions either have constant influence on the learning (in case of L1) or proportional to the error (in case of L2) without considering whether the error computed involves outliers or not. Alternatively, for small error differences either there are very small adjustments or constant adjustments to learning. Here our goal is to have a loss function that gives more importance to the small errors, thus following the works in mono-disparity estimation (Hu *et al.*, 2019; Watson *et al.*, 2019; Lee *et al.*, 2018) we also propose to use log ($\frac{\partial E}{\partial x} = \frac{1}{x}$, where $\frac{\partial E}{\partial x} \to 0, x \to \text{inf}$) as squeezing function that penalizes small disparity errors more in proportion to large disparity errors. Precisely we define,

$$E = log(|p_{ij} - q_{ij}| + \varepsilon) \tag{5.3}$$

We set $\varepsilon = 1$ to keep the loss positive. In our experiments, we found that not only this loss function lead to better performance but is more stable during training.

5.4 Experiments and Results

In this section, we first introduce the datasets used for the training and evaluation of our experiments. We then present the implementation details, followed by different design choices and results.

5.4.1 Datasets

We train and evaluate our models on two well-known stereo benchmarks, namely SceneFlow (Mayer *et al.*, 2016) and KITTI 2015 (Menze and Geiger, 2015).

KITTI:

KITTI datasets introduced as a benchmark are real world datasets of driving scenes (Geiger *et al.*, 2012; Menze and Geiger, 2015). KITTI 2012 contains 389 image pairs (194 training and 195 test image pairs) while KITTI 2015 contains 400 (200 training and 200 test image pairs) image pairs of resolution 376×1240 pixels with sparse ground-truth disparities for learning and evaluating stereo algorithms.

SceneFlow:

SceneFlow is a large scale synthetic dataset introduced by Mayer *et al* (Mayer *et al.*, 2016). It contains $35,454$ training images and $4,370$ test images of size 540×960 with dense ground truths.

Evaluation Metrics:

We report evaluation metrics including 3-*px*, 2-*px*, 1-*px*, D1 and End-Point-Errors (EPE) (Mayer *et al.*, 2016; Menze and Geiger, 2015). EPE (reported in *px* units) is the average disparity error, whereas 3-*px* represents the percentage of pixels whose error is greater than 3 pixels. Similarly, 2-*px* and 1-*px*, are percentage of pixels with errors greater then 2 and 1 pixels, respectively. D1 is the percentage of pixels where the 3-*px* error is greater than $0.05 \times$ ground-truth. We also report the number of parameters in millions (M) and number of MACs (Multiply-ACcumulate operations) in Giga-MACs (GMACs) for comparison. Note that 1MAC = 1 multiplication + 1 addition operation.

5.4.2 Implementation Details

We chose GwcNet (Guo *et al.*, 2019a) as our baseline method. For training, we use the Adam optimizer ($\beta_1 = 0.9$ and $\beta_2 = 0.999$). All the training images are cropped to the size of 256×512 with the maximum disparity set to 192. Furthermore, all the images are standard normalized using the mean and standard deviation of the ImageNet dataset. Our final model is trained with a batch size of 8 on 4 NVIDIA 2080 Ti GPUs. For the SceneFlow dataset, we train our model for 900K iterations with initial learning rate of *1E-3* which is reduced to half at fixed iterations steps, *i.e.*, at 600K, 700K, and 800K. For the KITTI 2015 dataset, we fine-tuned our pre-trained sceneflow model on KITTI datsets for 11K iterations with batch size of 16. The initial learning rate was set to *1E-3* which was later decreased to *1E-4* after 7K iterations.

Loss function	EPE (*px*)	D1 (%)	3-*px* (%)	2-*px* (%)	1-*px* (%)
L2	1.30	6.52	7.89	12.03	24.28
SmoothL1	0.83	2.86	3.58	5.24	10.80
LogL1Loss	**0.75**	**2.47**	**3.02**	**3.89**	**6.33**

Table 5.3: Performance evaluation of different loss functions.

5.4.3 Evaluation of Design Choices

Effect of Light-Weight Backbones:

During the evolution of the backbone we trained and experimented with different other light-weight backbones like ResNet18 (He *et al.*, 2016), ResNet34 (He *et al.*, 2016), MobileNet (Sandler *et al.*, 2018) and BiSeNet (Yu *et al.*, 2021) and its variants. Although ResNet18, ResNet34, MobileNet backbones require significantly less number of operations than BiSeNet (*i.e.* $5.1\times, 2.9\times$ and $13.6\times$ less operations respectively) however their performance was also far worse, *e.g.* ResNet34 had an EPE of 2 as compared to an EPE of 1.3 of our BiSeNet influenced model. Therefore we chose BiSeNet building blocks as our backbone, as this seems a reasonable choice *w.r.t.* to performance and speed. This is somewhat understandable as well since BiSeNet is optimized specially for image segmentation, which is itself a pixel-level task and thus has inherent capabilities to capture the semantic, textual and structural information at the pixel-level. For BiSeNet we also experimented with different sizes of the shallow branch and the deep branch, with different number of filters, number of layers and finally we settled with the architecture reported in the Tables 5.1 and 5.2.

Effect of the Loss Function:

In our initial experiments we did a comparison between L2, L1 and SmoothL1 loss, where SmoothL1 turned out to be consistently better. However the difference in performance gap to other competing methods was still large, especially the 1-*px* error – *c.f.* Table 5.3. We observed that despite using the SmoothL1 loss, the magnitude of 2-*px* and 1-*px* errors is high. As soon as we replaced the loss function with the LogL1 loss, we observed a noticeable performance improvement – *c.f.* Table 5.3. Here we can observe that the LogL1 loss not only improves EPE but also improves all others errors significantly, especially 1-*px* error. In addition we also observed that the LogL1 loss leads to faster convergence and more stable training of models.

Cost Volume	EPE (*px*)	D1 (%)	3-*px* (%)	2-*px* (%)	1-*px* (%)
w/o Attention	0.86	2.94	3.55	4.57	7.41
w Attention	**0.75**	**2.47**	**3.02**	**3.89**	**6.33**

Table 5.4: Effect of using learned attention weights for refining cost volume.

Encoder-decoder	EPE (*px*)	D1 (%)	3-*px* (%)	2-*px* (%)	1-*px* (%)	MACs (*G*)	Params (*M*)
$3\mathcal{D}$ Separable Conv	0.92	3.04	3.65	4.73	7.85	**4.95**	**36.51**
$3\mathcal{D}$ Conv	**0.75**	**2.47**	**3.02**	**3.89**	**6.33**	6.60	66.03

Table 5.5: $3\mathcal{D}$ Convolutions vs Depth-wise Separable $3\mathcal{D}$ Convolutions in encoder-decoder.

Effect of Attention in Cost Volume:

Table 5.4 compares the performance of the model trained with a simple group-wise correlation volume and a model where additional attention weights are learned and used to filter the important disparities. We can observe that the model with attention weights improves the performance for all the metrics. Overall, adding attention leads to a slight increased in MACs but the performance gain far outweights the increase in MACs here.

Effect of Separable Convolutions in the Encoder-Decoder:

To further reduce the computational requirements of our network we considered replacing costly $3\mathcal{D}$ convolutions in the encoder-decoder with light-weight depth-wise separable convolutions as suggested in (Rahim *et al.*, 2021). Table 5.5 shows that although $3\mathcal{D}$ separable convolutions reduce the number of parameters by $2\times$, we also witness a noticeable performance drop in all metrics. Furthermore, since $3\mathcal{D}$ separable convolutions do not have optimized kernel operations, for our network we decided not to use $3\mathcal{D}$ convolutions in the encoder-decoder part.

Effect of the Optimizer:

We evaluated the impact of different optimizers on our network performance. Specifically, we trained our models with Adam (Kingma and Ba, 2014), AdamW (Loshchilov *et al.*, 2017) and SGD. For our experiment, Adam and AdamW showed similar results (EPE 0.75 and 0.76 respectively), while SGD gave slightly inferior performance (0.80

Method	EPE (px)	D1 (%)	MACs (G)	Params (M)
GCNet (Kendall *et al.*, 2017)	1.84	-	718.01	3.18
PSMNet (Chang and Chen, 2018)	0.88	3.48	256.66	5.22
DeepPruner (Duggal *et al.*, 2019)	0.86	-	129.23	7.39
GwcNet-g (Guo *et al.*, 2019a)	0.79	2.77	246.27	6.43
CFNet (Shen *et al.*, 2021)	0.97	4.50	177.50	23.05
LEAStereo (Cheng *et al.*, 2020)	0.78	-	156.50	1.81
GANet (Zhang *et al.*, 2019)	0.93	3.49	383.42	4.48
GANetdeep (Zhang *et al.*, 2019)	0.84	3.29	670.25	6.58
HITNet (Tankovich *et al.*, 2021)	**0.43**	-	146.00	**0.97**
ACVNet (Xu *et al.*, 2022)	0.49	**1.65**	240.05	6.23
LeanStereo (Ours)	0.75	2.47	**66.03**	6.6

Table 5.6: Comparison on SceneFlow test set.

EPE). Therefore all the later networks were trained using teh Adam optimizer.

Based on conclusions drawn from the experiments on different design choices, we train our leaner backbone network with attention-based cost volume using LogL1 loss. This network is optimized using the Adam optimizer with a starting learning rate of *1E-3*.

5.4.4 Results

This section presents our quantitative and qualitative results on test and validation sets of KITTI 2015 and SceneFlow.

SceneFlow:

Table 5.6 shows our quantitative results on the SceneFlow test set, compared to existing stereo methods. We can observe that our network has superior EPE and D1 error than GwcNet and LEAStereo while having 4× and 2.5 × less operations, respectively. In addition, our network also contains 3.6× fewer operations when compared with ACVNet (Xu *et al.*, 2022) – for timing comparison *c.f.* Sec. 5.4.5. Fig. 5.4 shows qualitative results of our method in comparison with other existing methods. From the disparity

Method	EPE (px)	D1 (%)	3-px (%)	MACs (G)	Params (M)
PSMNet (Chang and Chen, 2018)	0.88	2.00	2.10	256.66	5.22
GANetdeep (Zhang *et al.*, 2019)	0.63	1.61	1.67	670.25	6.58
GANet (Zhang *et al.*, 2019)	0.67	1.92	2.01	383.42	**4.48**
GwcNet-g (Guo *et al.*, 2019a)	**0.62**	**1.49**	**1.53**	246.27	6.43
LeanStereo (Ours)	**0.62**	1.78	1.82	**60.41**	6.60

Table 5.7: Comparison on the KITTI 2015 validation set. All the methods are trained by us, with the same train/validation split for a fair comparison.

errors we can see that our method gives reliable disparity estimates that appears better than GwcNet while are comparable with ACVNet ones.

KITTI 2015:

For the KITTI dataset, we report both validation and test set results. All of the methods reported in Table 5.7 are trained by us, with the same train/validation split for a fair comparison. We can observe from the reported results that the performance of our model is on par with other methods, while having significantly fewer computations; for example, GwcNet and GANet require $4\times$ and $11\times$ more operations than our model, leading to their increased inference time in comparison. We also submitted the results of our model on the KITTI benchmark. Table 5.8 shows the quantitative comparison with other benchmark methods. We report the D1 error for foreground *(fg)*, background *(bg)* and all pixels following the benchmark. Qualitative results on KITTI benchmark are shown in Fig. 5.5.

5.4.5 Inference Time Comparison

Table 5.9 presents a comparison of the inference time for our final optimized method with $2\mathcal{D}$, $3\mathcal{D}$, and other real-time methods. The top half of the table lists results reproduced from the original papers* and may not be directly comparable across different image resolutions and hardware types. The bottom half presents results obtained using a standardized experimentation protocol.

To do a fair comparison of inference timings between our and all the recent state of the art methods we define and follow a uniform experimentation protocol.

*As no public or open-source implementations for the majority of these methods exists.

Methods	All(%)			Noc(%)		
	$D1_{bg}$	$D1_{fg}$	$D1_{all}$	$D1_{bg}$	$D1_{fg}$	$D1_{all}$
GCNet (Kendall *et al.*, 2017)	2.21	6.16	2.87	2.02	5.58	2.61
PSMNet (Chang and Chen, 2018)	1.86	4.62	2.32	1.71	4.31	2.14
DeepPruner (Duggal *et al.*, 2019)	1.87	3.56	2.15	1.71	3.18	1.95
GwcNet-g (Guo *et al.*, 2019a)	1.74	3.93	2.11	1.61	3.49	1.92
CFNet (Shen *et al.*, 2021)	1.54	3.56	1.88	1.43	3.25	1.73
LEAStereo (Cheng *et al.*, 2020)	1.40	2.91	1.65	1.29	2.65	1.51
GANet (Zhang *et al.*, 2019)	1.55	3.82	1.93	1.40	3.37	1.73
GANetdeep (Zhang *et al.*, 2019)	1.48	3.46	1.81	1.34	3.11	1.63
ACVNet (Xu *et al.*, 2022)	**1.37**	**3.07**	**1.65**	**1.26**	**2.84**	**1.52**
LeanStereo (Ours)	1.87	3.51	2.15	1.74	3.15	1.97

Table 5.8: Comparison on the KITTI 2015 benchmark (test set).

Specifically, we follow a two-step process to measure the inference timings. In the first step, a warm-up cycle is used where a subset of images (20 images of size $512 \times 960^{\dagger}$) is used to warm up the GPU (NVIDIA RTX 2080 Ti). In the next phase a lager subset of images (400 images of size 512×960) is used to record the mean inference time with standard deviation. This whole two-step process is repeated k times and then the average of the $k = 3$ runs is reported.

We can observe that our optimized (TensorRT FP16 version) network is $17\times$, $\sim 9\times$, $27\times$, $14\times$ faster than state of the art methods like PSMNet (Chang and Chen, 2018), ACVNet (Xu *et al.*, 2022), LEAStereo (Cheng *et al.*, 2020) and CFNet (Shen *et al.*, 2021) respectively. Moreover, our method has better run-time and EPE relative to all the compared $2\mathcal{D}$ and real-time methods.

5.5 Conclusions

In this work, we exploit the recent advancements in deep neural networks design and propose a fast stereo network with a lightweight backbone. Moreover, to recover the lost performance, due to the lighter backbone, we propose to use learned attention weights for refining the cost-volume and train the network with LogL1 loss. Our lean backbone based network trained with a LogL1 loss and attention-based cost volume (i) remains faster and gives superior performance relative to real-time and $2\mathcal{D}$ methods; and (ii) gives decent performance compared to recent state of the art $3\mathcal{D}$ methods, while being much faster (up to 9 to $14\times$). Based on this work we can conclude that although it is feasible

[†]rounded to near values for model that take input as factor of x.

Method	EPE(px)	Avg. Inference time(msec)
DispNetC (Mayer *et al.*, 2016)	1.68	60
StereoNet (Khamis *et al.*, 2018)	1.10	**15**
DeepPruner-Fast (Duggal *et al.*, 2019)	0.97	61
AANet (Xu and Zhang, 2020)	0.87	62
DecNet (Yao *et al.*, 2021)	0.84	50
HITNet (Tankovich *et al.*, 2021)	**0.43**	54
ACVNet-Fast (Xu *et al.*, 2022)	0.77	48
EdgeStereo (Song *et al.*, 2019)	0.74	320
PSMNet (Chang and Chen, 2018)	0.88	619
GA-Net-deep (Zhang *et al.*, 2019)	0.84	3580
GA-Net-11 (Zhang *et al.*, 2019)	0.93	2362
GwcNet-g (Guo *et al.*, 2019a)	0.79	240
CFNet (Shen *et al.*, 2021)	0.97	483
LEAStereo (Cheng *et al.*, 2020)	0.78	945
ACVNet (Xu *et al.*, 2022)	0.49	302
LeanStereo (Ours)	0.75	35

Table 5.9: Inference time comparison of various $2\mathcal{D}$, real-time, and $3\mathcal{D}$ stereo methods. The top half of the table presents methods whose timings were taken directly from their original publications, while the bottom half shows methods whose inference times were recomputed using a standardized evaluation setup (Sec. 5.4.5).

to design lighter and faster $3\mathcal{D}$ stereo methods by leveraging the recent advancements in network designs of overlapping tasks such as image segmentation, *etc*., however simple replacement will lead to deteriorated performance. To recover this performance loss, task specific calibrations like introduction of better feature merging steps and loss functions are necessary.

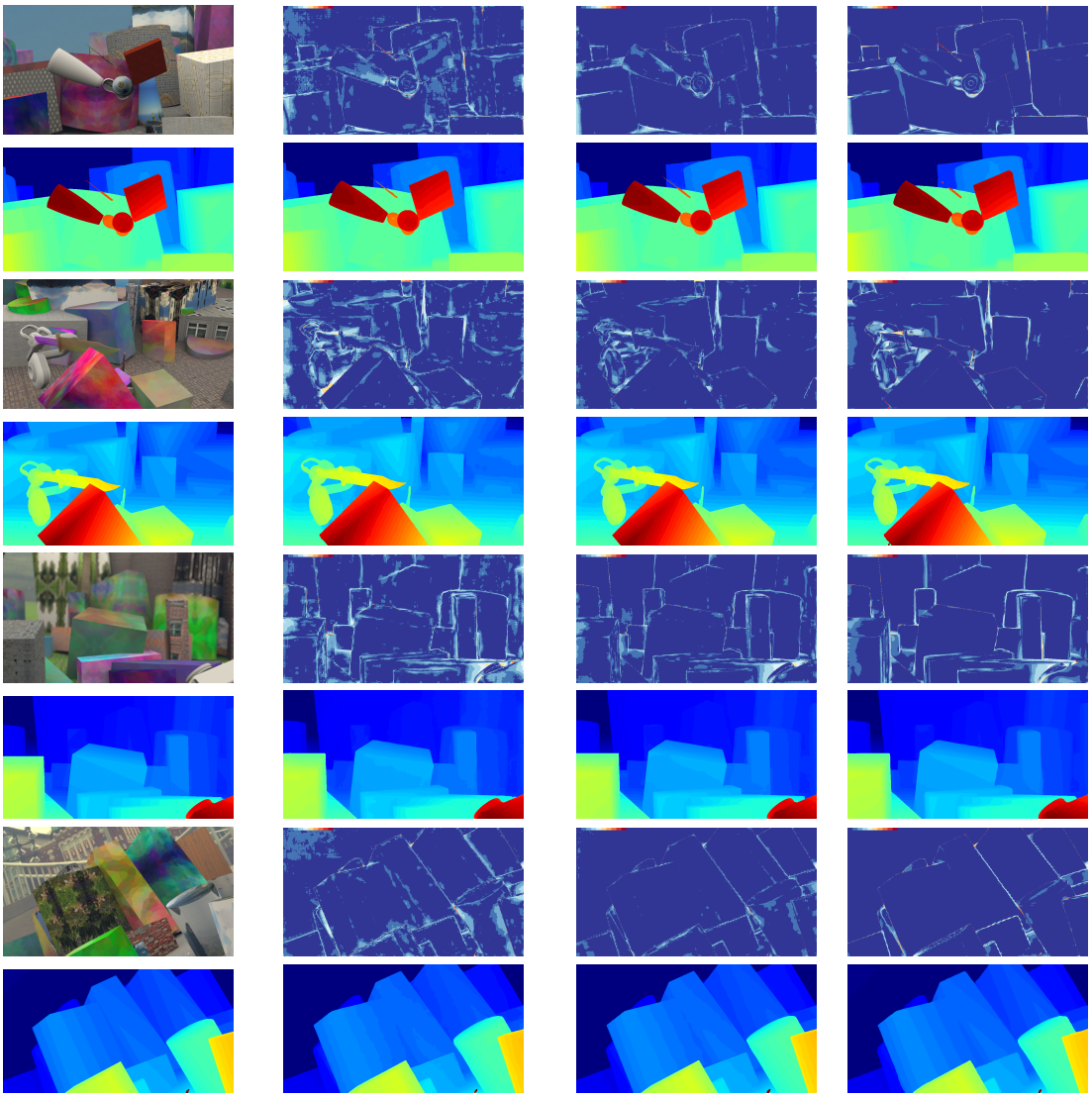


Figure 5.4: Qualitative results on sample SceneFlow images. Here the first and third rows represent the error maps w.r.t. ground truth. Darker red and blue colors represent higher and lower disparity errors, respectively.

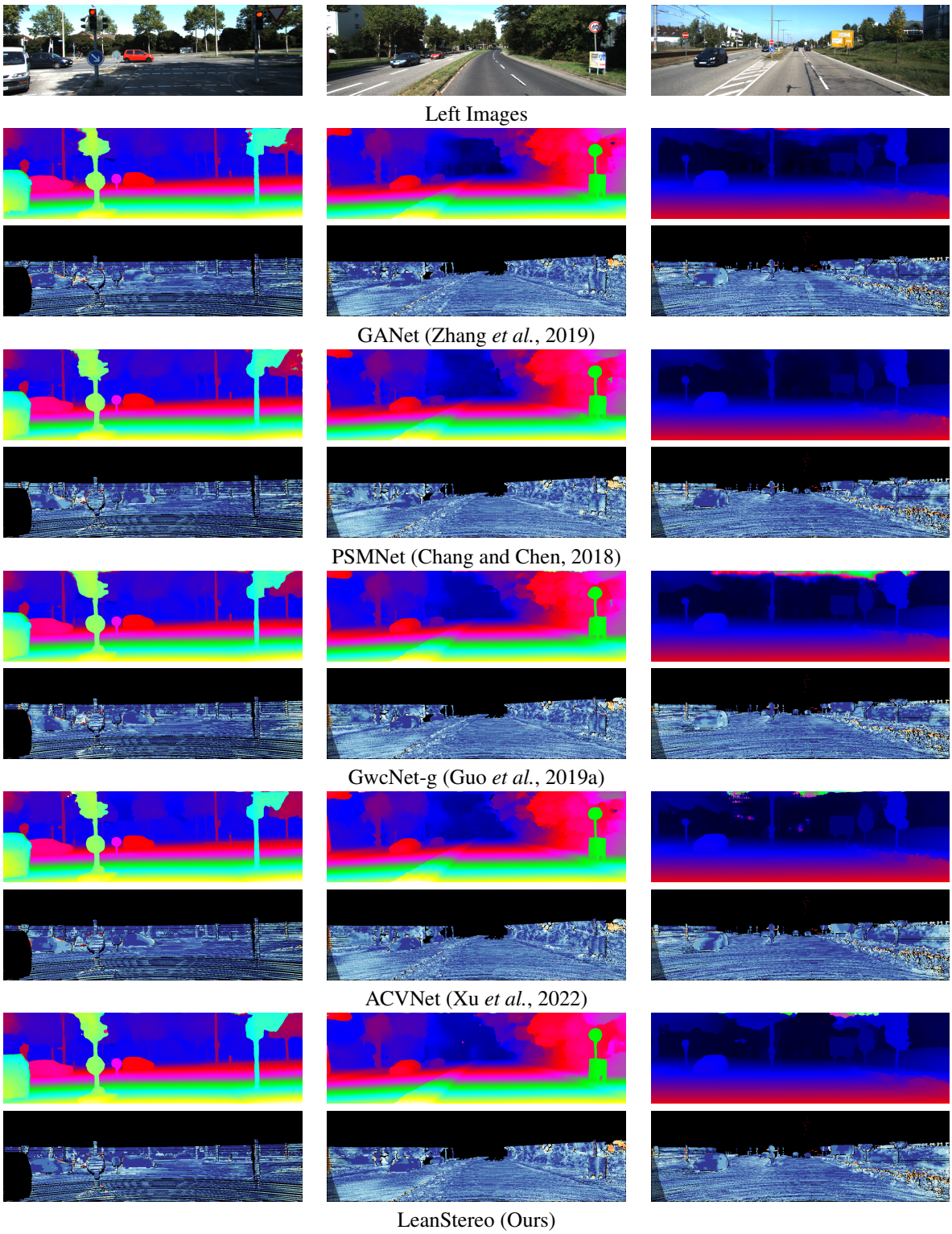


Figure 5.5: Qualitative performance (disparity images together with error maps) from the KITTI 2015 benchmark. Warmer colors in error maps denote larger values.

Chapter 6

Distilling Stereo Networks for Performant and Efficient Leaner Networks

6.1 Introduction

In previous chapters, we discussed methods for optimizing stereo networks, which often required extensive modifications to the network architecture to achieve comparable or superior performance to existing state-of-the-art networks. In this chapter, we take a different approach by leveraging knowledge distillation techniques and advancements in state-of-the-art stereo networks. The core idea is to train a small and lightweight student network to extract and learn knowledge from a larger and more complex teacher network.

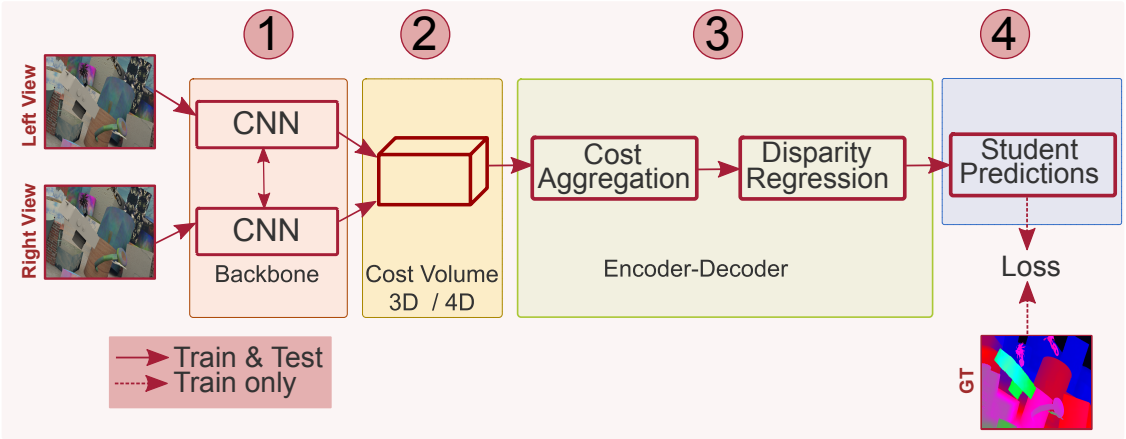


Figure 6.1: In this chapter, we leverage knowledge distillation to design leaner network. Specifically we consider multiple distillation points simultaneously to design a lightweight and accurate student network.

Specifically, we present our work titled ”Distilling Stereo Networks for Performant and Efficient Leaner Networks” by Rahim *et al* (2023). In this work, we propose a

lightweight student network that achieves competitive performance by learning intermediate representations from a larger and stronger teacher network. Our main objective is to systematically combine stereo methods and knowledge distillation techniques to determine effective distillation points and corresponding distillation losses. This allows us to design a lightweight stereo network that maintains competitive performance and inference times. Figure 6.1 illustrates the general pipeline of our approach, highlighting the multiple distillation points we consider simultaneously to design an accurate and lightweight student network.

While knowledge distillation has been successful in other computer vision domains (Liu et al., 2020; Beyer et al., 2022) for designing real-time lightweight student networks, there has been limited progress in the field of stereo vision (Gao et al., 2020). Currently, there is no existing open-source method for distilling stereo networks. The main challenge lies in defining an effective distillation pipeline for stereo matching, which involves addressing several design questions. These include selecting the student architecture or backbone, determining the points in the teacher network to distill information from, defining how to distill information from the four-dimensional cost volume, and deciding on suitable loss functions for different feature representations.

In this chapter, we address these design questions and propose efficient lightweight student networks using knowledge distillation. Our contributions are as follows:

1. We introduce a novel knowledge distillation pipeline for stereo matching, which involves:
   a) Carefully exploring different configurations to design an efficient student network.
   b) Empirically evaluating the importance of different distillation points.
   c) Exploiting learned attention weights for distilling the cost volume.
   d) Investigating the influence of different loss functions on the student network's performance.
   e) Designing a joint objective function to optimize the student network across different distillation points.
2. We conduct a detailed empirical evaluation of all the design choices involved in building an efficient student network using knowledge distillation.
3. As a result, our trained student network (called DSNet), built using the proposed knowledge distillation pipeline, gives reasonable performance compared to state of the art teacher network while having $3\times$ fewer parameters and being $8\times$ faster. Moreover, compared to performance oriented state-of-the-art methods (on SceneFlow dataset) like PSMNet (Chang and Chen, 2018), CFNet (Shen *et al.*, 2021), and LEAStereo (Cheng *et al.*, 2020) our student not only gives better performance but is $8\times$, $5\times$, and $8\times$ faster, respectively. On the other hand, compared to speed oriented methods our students perform better than all the tested methods on KITTI and SceneFlow benchmarks while being faster as well – *c.f*. Fig. 6.2. For instance our student network has an inference time of $25ms$ compared to $39ms$ of recently proposed Fast-ACVNet.

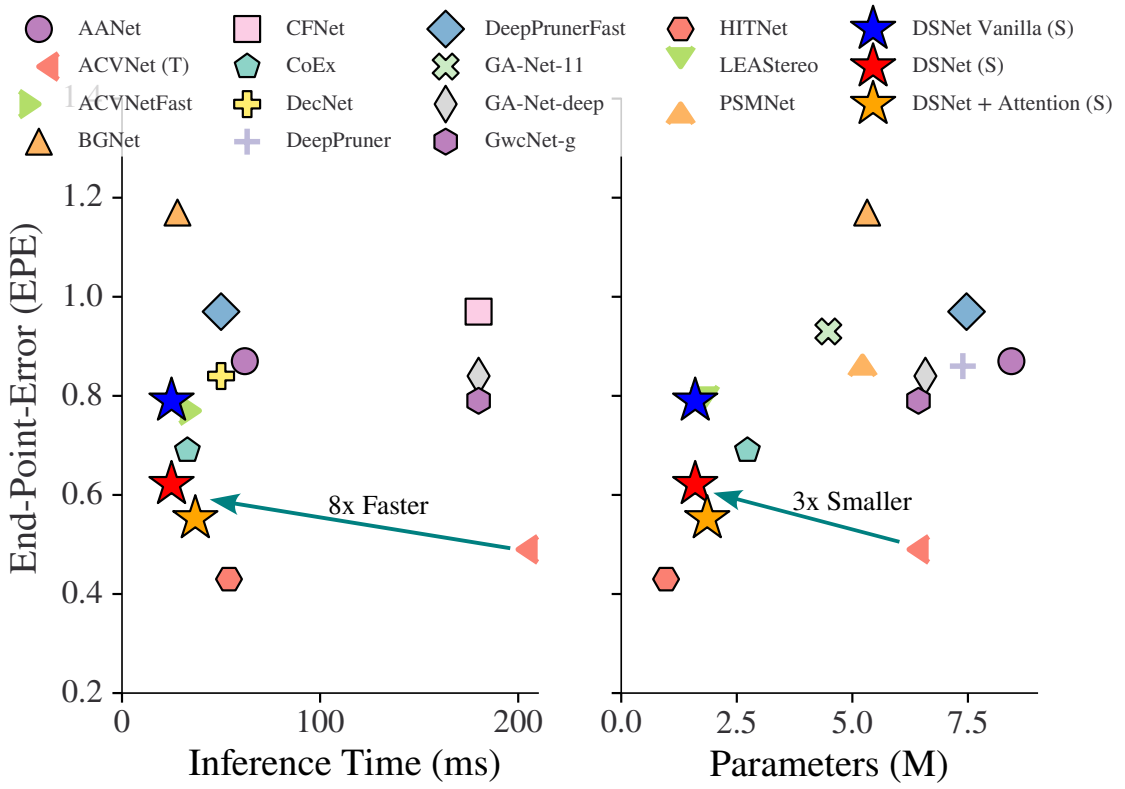


Figure 6.2: Comparison of our student network with other state of the art methods on SceneFlow test set. Compared to teacher network (ACVNet (Xu *et al.*, 2022)), our proposed student network (DSNet (S)) has 3× fewer parameters and is 8× faster.

6.2 Related Work

Recently, the computer vision community's focus has shifted from optimizing deep neural networks for performance to optimizing them for computational requirements (Sandler *et al.*, 2018; Tan and Le, 2019; Yu *et al.*, 2021; Hinton *et al.*, 2015), as models are becoming bigger and more resource hungry. Given this, numerous efforts have also been made to make stereo networks light-weight and real-time.

GwcNet (Guo *et al.*, 2019a) introduced group-wise correlation to reduce the size of costly 4D volume introduced in (Kendall *et al.*, 2017), to make the overall network faster and better. Anynet (Wang *et al.*, 2019) reduces the network size by deploying a coarse-to-fine resolution strategy that has been successful in $2\mathcal{D}$ networks. Deeppruner (Duggal *et al.*, 2019) uses patchmatch to narrow down the search range for disparities. MobileStereoNet (Shamsafar *et al.*, 2022) and depth-wise separable convolutions (Rahim *et al.*, 2021) have also significantly reduced the stereo network's computational require-

ments. StereoNet (Khamis *et al.*, 2018) uses a very low resolution cost volume to regress disparities at low resolution and then hierarchically improve the disparity maps to full resolution. MADNet (Tonioni *et al.*, 2019) introduced an unsupervised real-time stereo network that independently trains the sub-portions of the network. Although there have been many works to speed up stereo networks (Xu and Zhang, 2020; Yao *et al.*, 2021; Tankovich *et al.*, 2021; Song *et al.*, 2019) there is a long way to go, specifically for $3\mathcal{D}$ stereo networks, which are still very computation demanding.

Lately, knowledge distillation has been a very successful in other domains like face recognition (Luo *et al.*, 2016a), classification (Beyer *et al.*, 2022), object detection (Chen *et al.*, 2017), segmentation (Liu *et al.*, 2020), and semi-supervised (Xie *et al.*, 2020) and self-supervised (Grill *et al.*, 2020) learning to train the light weight networks. Knowledge distillation (Hinton *et al.*, 2015) is a technique of learning the output and / or intermediate activations of a bigger and stronger network (usually called a teacher network) by a compact and light-weight network called a student network. When is to comes to stereo matching, very limited research has been done in this direction (Gao *et al.*, 2020). Compact StereoNet (Gao *et al.*, 2020) has only explored one distillation point with only SmoothL_1 as loss function, therefore there is a need of a method that explores multiple distillation points with different possible loss functions. There exist no open source methods that extensively explore the knowledge distillation for stereo estimation task and can be used as baseline for further research. In this work, we aim to exploit the principle of knowledge distillation, given its success in numerous domains, to design an open-source real-time student stereo network.

6.3 Methodology

To build a knowledge distillation pipeline for stereo matching, one needs to answer the following questions:

- What should be the design of the student network?
- What network to use as a teacher?
- Should we use multiple distillation points or not? If yes, then what should be those distillation points?
- What loss functions to use for different types of features having different dimensions?

In the subsequent subsections, we address these questions in detail. We thoroughly examine each building module of a stereo network, with the aim of constructing a lightweight student network derived from a more robust teacher network.

6.3.1 Student Network (S)

We first design a leaner student network by drawing influence from recent state of the art visual recognition and stereo-matching methods (Sandler *et al.*, 2018; Wang *et al.*, 2018;

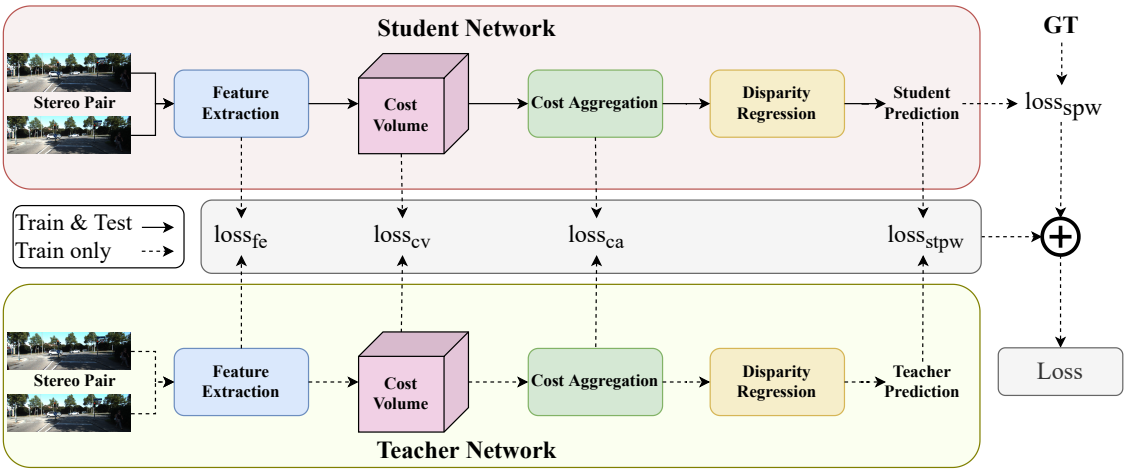


Figure 6.3: Proposed knowledge distillation pipeline for stereo matching. We use different distillation points with different types of loss functions to distill learned information from teacher to student. Here $loss_{fe}$ is feature extraction loss, $loss_{cv}$ is cost volume loss, $loss_{ca}$ is cost aggregation loss, $loss_{stpw}$ is student pixel-wise loss *w.r.t.* teacher and $loss_{spw}$ is student pixel-wise loss *w.r.t.* ground-truth(GT). *Loss* is overall learning objective defined by accumulating each component's loss as explained in Sec. 6.3.

Guo *et al.*, 2019a). A typical stereo network generally consist of four modules: feature extraction, cost volume, cost aggregation and disparity regression (Fig. 6.3). The next subsections contain in-depth details on these modules.

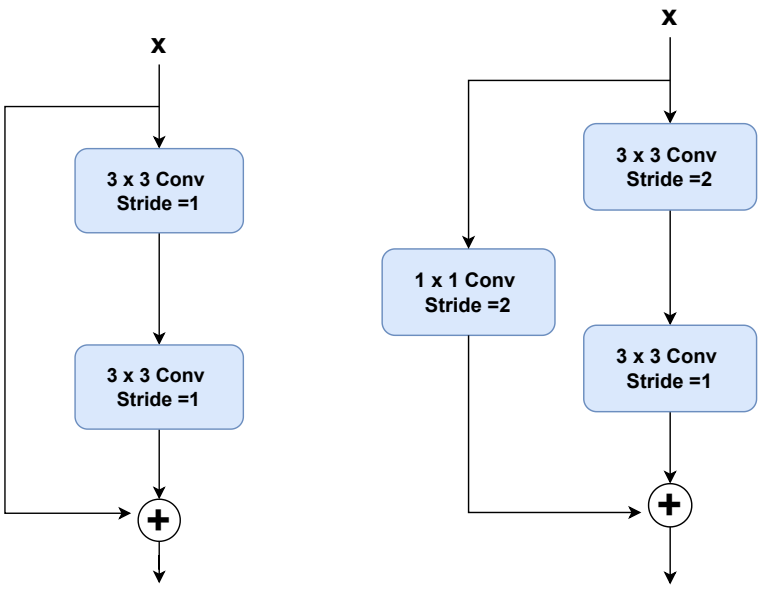
Feature Extraction:

The feature extraction module is basically a shared network with $2\mathcal{D}$ convolution operations and is responsible for modeling important features from the input image pair – *c.f*. Fig. 6.3. We keep our backbone design explicitly simple by restricting the network building blocks set to the one known to be computationally efficient like ReLU, kernel sizes of 3×3, *etc*.

Precisely, the complete $2\mathcal{D}$ feature backbone is build from a series of 3×3 convolutional layers, $B1$ and $B2$ blocks (Simonyan and Zisserman, 2014; He *et al.*, 2016; Zhang *et al.*, 2019). The $B1$ is constructed from two layers of 3×3 convolutions with a simple skip connection – *c.f*. Fig. 6.4(a). The $B2$ block is similar to $B1$ except that, to increase the receptive field size and to reduce the spatial dimensions, we replace the first convolutional layer with a strided convolution and also use a 1×1 strided convolution in the skip connection branch.

To address computational load requirements, we build and test three different variants of $2\mathcal{D}$ feature extraction modules, namely BB_{21}, BB_{18} and BB_{14}, with $21, 18$ and 14 convolutional layers, respectively – note we count the total number of convolutional layers in the module for uniform comparison. Table 6.1 contains the details of the BB_{21}

feature extraction module, here Level 1 and 2 means that the spatial resolution of the features maps is being reduced by $1/2$ and $1/4$ with respect to the input height (H) and width (W). As a final output of this module, we concatenate features from the last three layers of the network.



(a) B1 block (b) B2 block

Figure 6.4: Design of B1 and B2 blocks in feature extraction module.

Cost Volume Construction:

This module merges the output feature maps from left and right images to produce four-dimensional output. For cost volume construction we have explored and experimented with two different methods. In the first method, left and right image feature maps of size $320 \times H/4 \times W/4$ are merged via group-wise correlation to produce an output of $G \times D/4 \times H/4 \times W/4$, where G is the correlation group size and D is the maximum disparity.

Although this simple method leads to a faster student with good performance, to further improve the performance we have also experimented with a another method. In this method, we integrate learned attention with group-wise correlation to design the cost volume (Xu *et al.*, 2022). Specifically, first a small attention network is used to learn the attention weights from the output of the feature extraction module. Next, these learned weights are then applied to the group-wise correlation volume to refine and improve the overall disparity regression – *c.f*. Table 6.5.

No.	Level	Input Size	Operation	Output Size
1.	1	$3 \times H \times W$	3×3 Conv	$32 \times H/2 \times W/2$
2.	1	$32 \times H/2 \times W/2$	3×3 Conv	$32 \times H/2 \times W/2$
3.	1	$32 \times H/2 \times W/2$	3×3 Conv	$32 \times H/2 \times W/2$
4.	1	$32 \times H/2 \times W/2$	B1	$32 \times H/2 \times W/2$
5.	1	$32 \times H/2 \times W/2$	B1	$32 \times H/2 \times W/2$
6.	2	$32 \times H/2 \times W/2$	B2	$64 \times H/4 \times W/4$
7.	2	$64 \times H/4 \times W/4$	B1	$64 \times H/4 \times W/4$
8.	2	$64 \times H/4 \times W/4$	B1	$64 \times H/4 \times W/4$
9.	2	$64 \times H/4 \times W/4$	B1	$64 \times H/4 \times W/4$
10.	2	$64 \times H/4 \times W/4$	B1	$128 \times H/4 \times W/4$
11.	2	$128 \times H/4 \times W/4$	B1	$128 \times H/4 \times W/4$
Output	2	Layer 9, 10, and 11	Concat	$320 \times H/4 \times W/4$

Table 6.1: Architecture of student's (BB_{21} with 21 layers) feature extraction module.

Cost Aggregation:

The cost aggregation module takes the output from the $4\mathcal{D}$ cost volume and processes it by a series of $3\mathcal{D}$ encoder-decoder (ED) networks. Since this module uses $3\mathcal{D}$ convolutions it consumes the majority of the operations in the overall network. To keep the computational foot-print of the student network manageable we also explored different design choices for this module as well. To this end, we experimented with varying the number of ED networks and the number of $3\mathcal{D}$ layers (N) in each ED network. Overall we experimented with six different variants as explained in Sec. 6.4.3. Table 6.2 contains the design of a ED network composed of four layers of $3\mathcal{D}$ encoder convolutions and two layers of transposed $3\mathcal{D}$ decoder convolutions.

Disparity Regression:

In disparity regression, during training, we regress the disparity from multiple intermediate network outputs to improve overall features learning inside the network (Guo *et al.*, 2019a). Precisely, first the output of each ED network is passed through a pair of $3\mathcal{D}$ convolutions and upsampling operations to obtain an output volume of size $D \times H \times W$. This volume is then converted to probabilities via *softmax* and passed through *softargmin* to estimate the disparity map of size $H \times W$. During inference, disparities are estimated only from the output of the final ED network.

No.	Level	Input Size	Operation	Output Size
1.	3	$N \times D/4 \times H/4 \times W/4$	$3 \times 3 \times 3$ Conv	$(N \times 2) \times D/8 \times H/8 \times W/8$
2.	3	$(N \times 2) \times D/8 \times H/8 \times W/8$	$3 \times 3 \times 3$ Conv	$(N \times 2) \times D/8 \times H/8 \times W/8$
3.	4	$(N \times 2) \times D/8 \times H/8 \times W/8$	$3 \times 3 \times 3$ Conv	$(N \times 4) \times D/16 \times H/16 \times W/16$
4.	4	$(N \times 4) \times D/16 \times H/16 \times W/16$	$3 \times 3 \times 3$ Conv	$(N \times 4) \times D/16 \times H/16 \times W/16$
5a.	3	$(N \times 4) \times D/16 \times H/16 \times W/16$	$3 \times 3 \times 3$ Transpose Conv	$(N \times 2) \times D/8 \times H/8 \times W/8$
5b.	3	$(N \times 2) \times D/8 \times H/8 \times W/8$	Skip connection from 2.	$(N \times 2) \times D/8 \times H/8 \times W/8$
6a.	2	$(N \times 2) \times D/8 \times H/8 \times W/8$	$3 \times 3 \times 3$ Transpose Conv	$N \times D/4 \times H/4 \times W/4$
6b.	2	$N \times D/4 \times H/4 \times W/4$	Skip connection from 1.	$N \times D/4 \times H/4 \times W/4$

Table 6.2: $3\mathcal{D}$ Encoder-Decoder (ED) based cost aggregation module. Here 'N' is the number of $3\mathcal{D}$ convolution filters in a layer.

6.3.2 Teacher Network (T)

As a teacher network, we experimented with different state of the art publicly available stereo matching methods. Specifically, we explored ACVNet (Xu *et al.*, 2022), GwcNet (Guo *et al.*, 2019a) and Lac-Net (Liu *et al.*, 2022) as our teacher networks. ACVNet is a current state-of-the-art method (at the time of writing this thesis)* and thus is a strong candidate to be considered as a teacher network – we also find it to be the most promising teacher empirically as well. In addition, we also consider GwcNet and Lac-Net as teachers to evaluate how well the student network can learn coupled with teachers of different strengths and strong ground truth signals. We also explored using multiple teachers. However, since these methods have large memory foot-print, we were not able to fit more than one teacher into our training setup.

6.3.3 Loss Functions

For knowledge distillation, we investigate many different commonly used loss functions in image-level tasks, such as stereo matching, segmentation, mono-depth estimation *etc*. as well as those used by other distillation networks (Liu *et al.*, 2020; Park *et al.*, 2019; Chen *et al.*, 2021; Passalis and Tefas, 2018). Here we provide details of the loss functions that we have evaluated and which have been finally selected to train our student model. Overall, we find that in knowledge distillation for stereo matching not only the type of loss function is important but also at what point in the network the loss function is being used for information distillation – *c.f*. Sec. 6.4.

In the following discussion, Φ_d^S and Φ_d^T represent multi-dimensional student and teacher features respectively and d is feature matrix rank, which, depending on at the distillation point in the network, can be of rank 2, 3 or 4.

*Although HITNet (Tankovich *et al.*, 2021) is another very strong competing method but has no public implementation available.

L_2 Loss (MSE):

We use the L_2 loss for the training of our base student network, just like in many other image-level predictions tasks, where

$$L_2(\Phi,\Theta) = \| \Phi_d - \Theta_d \|^2$$

Here Φ represents the multi-dimensional student features, Θ represents the teacher network features and d is feature matrix rank, that, depending on at which distillation point the loss is being used, can be of rank 2,3 or 4.

SmoothL$_1$ Loss:

Recent stereo methods (Gao *et al.*, 2020; Xu *et al.*, 2022) use SmoothL$_1$ as loss function for learning disparities. SmoothL$_1$ is known to be much more robust against outliers and also leads to better learning of the model, as it avoids the problem of exploding gradients in some cases (Ren *et al.*, 2015).

$$SmoothL_1(\Phi_d^S,\Phi_d^T) = \begin{cases} 0.5\frac{\|\Phi_d^S-\Phi_d^T\|^2}{\tau}, & \text{if } |\Phi_d^S-\Phi_d^T| < \tau \\ |\Phi_d^S-\Phi_d^T| - 0.5\tau, & \text{else} \end{cases}$$

where we use $\tau = 1$.

LogL$_1$ Loss:

One of the downsides of using L_2 or L$_1$ (or both) as loss function for learning disparities is that these error functions either have constant influence on the learning (in case of L$_1$) or proportional to error (in case of L$_2$) without considering whether the error computed involves outliers or not. To this end, following the works in mono-disparity estimation (Hu *et al.*, 2019; Watson *et al.*, 2019; Lee *et al.*, 2018), we use log as a squeezing function on top of L$_1$ loss to penalize small errors severely as well. Precisely,

$$LogL_1(\Phi_d^S,\Phi_d^T) = \log(|\Phi_d^S-\Phi_d^T| + \varepsilon)$$

To keep loss values positive, we set $\varepsilon \geq 1$. Our experiments show that the LogL$_1$ loss leads to better performance and is more stable during the training.

Cosine Loss:

For matching features in the earlier layers and cost volumes, we use a cosine similarity based loss function, where

$$CosineLoss(\Phi_d^S,\Phi_d^T) = 1 - \cos(\Phi_d^S,\Phi_d^T)$$

KL Divergence (KLD) Loss:

In this loss function, we consider the teacher predicted disparities (or features) as true class probabilities (p_T) and student disparities as predictions probabilities (p_S) and measure the Kullback-Lieber divergence (KLD) as a loss. *I.e.*

$$KLDLoss(S \parallel T) = p_T \log p_T - p_T \log p_S,$$

In addition, we also evaluated many other commonly used loss functions for knowledge distillation. However, in our experiments these losses produced inferior performance so they are not reported any further. For instance, we also tested pair-wise loss (Liu *et al.*, 2020), Relational Knowledge Distillation (RKD) (Park *et al.*, 2019), Hierarchical Context Loss (HCL) (Chen *et al.*, 2021) and Probabilistic Knowledge Transfer(PKT) (Passalis and Tefas, 2018) losses.

6.3.4 Distillation Points

Initially in our baseline student network we used only one point to distill the output of the teacher's network – here distilling the output means supervising the student to match the teacher's output via a loss function. However the performance of the student network was too weak, thus to improve its performance we started exploring different distillation points and different loss functions (due to different type and dimensionality of the features) for each of the distillation points – *c.f.* Fig. 6.3.

For distilling the information from the feature extraction module, we investigated two different distillation points, one from the early layers (from layer 3 and 5 – *c.f.* Table 6.1) of network and one from the later layers (layers 9, 10 and 11) of the network. Since the cost volume is one of the most important modules in the overall pipeline, as it sets the base for learning of the disparities in the following modules, we set our second distillation point at the output of the cost volume module. The third distillation point was set at the output of the cost aggregation module. Finally the last distillation point was set at the output of the disparity regression module to match the student predicted disparities with the teacher's. All of these choices were thoroughly validated via detailed ablation studies – *c.f.* Sec. 6.4.3.

6.3.5 Learning Objective Function

Finally, based on all the distillation points and teacher and student network outputs, we define the overall objective function as:

$$\ell(S,T,GT) = \begin{cases} \lambda \ell_{fe}(S,T) + \lambda \ell_{cv}(S,T) + \\ \lambda_{ca}\ell_{ca}(S,T) + \lambda_{spw}\ell_{spw}(S,GT) + \\ \lambda_{stpw}\ell_{stpw}(S,T) \end{cases}$$

Here S, T and GT represent student, teacher and ground truth, respectively; and ℓ_{fe}, ℓ_{cv}, ℓ_{ca} are distillation point losses from the feature extraction, cost volume, and cost aggregation modules, respectively. ℓ_{spw} represents the student's pixel-wise loss *w.r.t.* GT while ℓ_{stpw} represents the student's pixel-wise loss *w.r.t.* T. We set $\lambda = 0.1$, $\lambda_{ca} = 0.1$, $\lambda_{spw} = 0.4$ and $\lambda_{stpw} = 0.4$ for our final student network.

6.4 Experiments and Results

In this section, we first introduce the datasets used for the training and evaluation of our experiments. We then present the implementation details, followed by the discussion on different design choices and results.

6.4.1 Datasets

For the training and evaluation of our models we use two well-known stereo benchmarks datasets, namely SceneFlow (Mayer *et al.*, 2016) and KITTI 2015 (Menze and Geiger, 2015). For measuring the generalization capability of our setup and trained student models we use two more stereo datasets, namely ETH3D (Schops *et al.*, 2017) and Middlebury (Scharstein *et al.*, 2014).

KITTI (Geiger *et al.*, 2012; Menze and Geiger, 2015) is a real-world dataset of driving scenes with sparse annotations. KITTI 2012 contains 194 training and 195 test pairs while KITTI 2015 contains 200 image pairs for training and testing each, with resolution 376×1240 pixels.

SceneFlow (Mayer *et al.*, 2016) is a large scale synthetic dataset with dense annotations. It contains 35,454 training images and 4,370 test images of size 540×960.

Evaluation metrics: We report mutliple evaluation metrics including D1, End-Point-Errors (EPE) (Mayer *et al.*, 2016; Menze and Geiger, 2015) and K-*px* errors. EPE (reported in *px* units) is the average disparity error. D1 is the percentage of pixels where the 3-*px* error is greater than 5% of the ground-truth. K-*px* is the percentage of pixels where the error is greater than K pixels, for our experiments $K \in \{1,2,3,4\}$. We also report the number of parameters in millions (M) and number of operations/MACs (Multiply-ACcumulate operations) in Giga-MACs (GMACs) for comparison[†]. For all of the evaluation metrics, lower values indicates better method.

6.4.2 Implementation Details

For network training, we use the Adam optimizer ($\lambda = 10^{-4}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$). All the training images are cropped to the size of 256×512 with the maximum disparity set to 192. Furthermore, all the images are standardized using the mean and standard

[†]1MAC = 1 multiplication + 1 addition. These operations are measured using https://github.com/sovrasov/flops-counter.pytorch

deviation of the ImageNet dataset (Deng *et al.*, 2009). Our final model is trained with a batch size of 8 on 4 GPUs. For the SceneFlow dataset, we train all our models for 64 epochs (like (Xu *et al.*, 2022)) with an initial learning rate of 10^{-4} that is reduced to half at the $20, 32, 40, 48$ and 56^{th} epochs. For the KITTI 2015 dataset, we fine-tuned our pre-trained SceneFlow model for 500 epochs. The initial learning rate was set to 10^{-4} which was later decreased to $1/5^{th}$ after 250 epochs.

Variant	Params (M)	MACs (G)	EPE (px)
BB_{56}-ED_3-N_{32}	6.52	246.27	0.73
BB_{21}	4.43	211.12	0.76
BB_{18}	4.26	207.48	0.77
BB_{14}	4.12	200.84	0.79
BB_{21}-ED_3	4.43	211.12	0.76
BB_{21}-ED_2	3.29	165.50	0.77
BB_{21}-ED_1	2.16	120.06	0.83
BB_{21}-ED_2-N_{24}	2.25	108.7	0.78
BB_{21}-ED_2-N_{16}	1.50	67.20	0.79
BB_{21}-ED_2-N_8	1.05	41.15	0.85

Table 6.3: Comparison of different student variants performance on SceneFlow dataset. For all the metrics, lower means better.

6.4.3 Evaluation of Design Choices

Student Network (S):

As discussed in Sec. 6.3, to reduce the computational complexity, we designed and tested student networks with different numbers of layers in the feature extraction module and different numbers of ED networks in the cost aggregation module. Initially, we started with a very strong baseline student network (BB_{56}-ED_3-N32) containing 56 convolutional layers in the feature extraction module and 3 ED networks (with each having 32 $3\mathcal{D}$ convolutional layers) in the cost aggregation module – this network was trained using SmoothL_1 loss. This network was then iteratively tailored down to have reasonable performance while maintaining a small computational footprint.

We started by reducing the number of layers in the $3\mathcal{D}$ feature extraction backbone. To this end, we trained three different variants BB_{21}, BB_{18}, BB_{14} with 21, 18 and 14 convolution layers in the backbone. As results in the Table 6.3 show, all of the variants give comparable performance with a very small difference in parameters and operations. Therefore, we choose to proceed with BB_{21} as having more layers can be helpful when we further reduce the student size in other parts of the network.

As observed by us and many others (Rahim *et al.*, 2021; Shamsafar *et al.*, 2022) that majority of operations in a $3\mathcal{D}$ stereo network are consumed by the ED networks in the cost aggregation module. So after finalizing the number of layers in the backbone network, we explored reducing the number of of ED networks in cost aggregation from 3, to 2 and eventually to 1. Table 6.3 shows the results of different variants of these student networks. As can be observed from the results, decreasing the number of ED networks leads to loss in performance. However, the drop in performance going from ED_3 to ED_2 is not as significant as the reduction in the number of MACs. However, going from ED_2 to ED_1 has noticeable impact on the performance. As a result we use ED_2 as cost-aggregation module for our candidate student network(s).

After finalizing the number of layers in the backbone network and number of ED networks we explored reducing the number of $3\mathcal{D}$ convolutional filters. Precisely, starting from a base student network BB_{21}-ED_2-N_{32} we further reduce the student computational footprint by lowering the number of $3\mathcal{D}$ filters from 24 to 16 and finally to 8. As it can be noted, reducing the number of $3\mathcal{D}$ filters lead to significant drop in MACs but with noticeable performance drop, from 0.78 EPE to 0.85. Consequently, we finally chose BB_{21}-ED_2-N_{16} as our candidate student network (DSNet Vanilla) for the best compromise between performance and speed.

Distillation Points and Losses:

Although our chosen candidate (DSNet Vanilla) has a lower computational footprint, it has weak performance compared to the teacher and other state of the art $3\mathcal{D}$ stereo methods. For instance, just trimming down the student network from the baseline results in a rise of EPE from 0.73 to 0.79. We recover this lost performance by introducing different distillation points and using well-tuned loss functions for each distillation point to choose the best performing loss function. We started from the selected student network as a baseline *i.e.* DSNet Vanilla (for knowledge distillation ablation studies) and as a first step replaced the SmoothL$_1$ by LogL$_1$ in student-pixel-wise loss (ℓ_{spw}). As we can observe from Table 6.4 that LogL$_1$ loss performs better than SmoothL$_1$ loss, we chose LogL$_1$ loss as reference ℓ_{spw} loss for the further experiments.

Next we included the feature extraction and cost volume distillation points. For the distillation of backbone features and cost volume, we observed that Cosine significantly reduces the EPE relative to other candidate loss functions and thus was chosen for representing ℓ_{fe} and ℓ_{cv}. We also found empirically that distilling backbone features from only earlier layers gave better performance relative to later layers features. Finally, for ℓ_{stpw} we chose SmoothL$_1$ to measure the pixel-wise loss between the predicted disparity maps from the student and teacher networks. Our student model, trained using selected loss functions and distillation points, is called DSNet.

To further improve the performance of the student network, we train another student variant (DSNet+Attention) by adding an attention learning mechanism and distill the attention weights from the teacher network. From the results, we can observe that the

Loss Function	ℓ_{spw}	ℓ_{cv}	ℓ_{fe}	ℓ_{ca}	ℓ_{stpw}	ℓ_{cv} + Attention	EPE(px)
SmoothL_1	✓						0.79
LogL_1	✓						<u>0.76</u>
MSE	✓	✓					0.75
KLD	✓	✓					0.72
Cosine	✓	✓					<u>0.71</u>
LogL_1	✓	✓					0.77
MSE	✓	✓	✓				0.74
KLD	✓	✓	✓				0.73
Cosine	✓	✓	✓				<u>0.71</u>
LogL_1	✓	✓	✓				0.74
MSE	✓	✓	✓	✓			0.71
KLD	✓	✓	✓	✓			<u>0.65</u>
Cosine	✓	✓	✓	✓			0.66
LogL_1	✓	✓	✓	✓			0.69
LogL_1	✓	✓	✓	✓	✓		0.65
smoothL1	✓	✓	✓	✓	✓		<u>0.62</u>
Cosine	✓	✓	✓	✓	✓	✓	<u>0.55</u>

Table 6.4: Results of ablation study of distillation points and losses on SceneFlow (Mayer *et al.*, 2016) dataset.

Loss Function for added module	ℓ_{spw}	ℓ_{cv}	ℓ_{fe}	ℓ_{stpw}	ℓ_{ca}	ℓ_{cv} + Attention	EPE (px)	D1 (%)	3-px (%)	2-px (%)	1-px (%)
LogL_1	✓						0.76	2.60	3.09	3.89	6.05
Cosine	✓	✓					0.71	2.49	2.98	3.80	6.04
Cosine	✓	✓	✓				0.71	2.48	2.97	3.78	6.01
SmoothL_1	✓	✓	✓	✓			0.68	2.45	2.97	3.83	6.21
KLD	✓	✓	✓	✓	✓		0.62	2.20	2.70	3.54	5.90
Cosine	✓	✓	✓	✓	✓	✓	0.55	1.99	2.43	3.18	5.17

Table 6.5: Impact of using different loss functions and distillation points on the performance of DSNet.

attention mechanism improves the EPE significantly, with a slight increase in computational efforts of our student model.

Table 6.5 shows the final chosen distillation points and the corresponding loss functions used in DSNet. Overall, it is important to use multiple distillation points along with a matching loss function to achieve the best performance. For distilling the feature

information from teacher to student (in the network's inner distillation points) the Cosine loss is the best choice for the loss function followed by the KLD loss. Distilling backbone features only from earlier layers gave better performance relative to later layers features. For matching disparities between student and ground-truth (ℓ_{spw}) LogL_1 is the best choice while using SmoothL_1 to measure pixel-wise loss (ℓ_{stpw}) between predicted disparity maps of student and teacher networks gives better performance.

Adding an attention learning mechanism in the student (DSNet+Attention) and distilling the attention weights from the teacher network to the student leads to noticeable improvement in the performance. However, this comes at the cost of increased computational footprint. For instance, the inference time of DSNet+Attention increases from $25ms$ of DSNet to $37ms$.

Teacher Network (T):

As discussed in Sec. 6.3.2, we experimented with different models as teacher networks. Overall, we observed that having a strong teacher ACVNet (Xu *et al.*, 2022), as expected, results in better learning of the student network – substituting ACVNet with GwcNet (Guo *et al.*, 2019a)) as teacher network increases the EPE from 0.62 to 0.74. We also note that the design of the cost volume also plays an important role, for instance GwcNet turns out to be a better teacher than Lac-Net, although Lac-Net is a better performing method than GwcNet (replacing GwcNet with Lac-Net increases the EPE to 0.79). From this, we can conclude that the cost volume plays a vital role in overall stereo pipeline and aligning the design between the cost volumes of student and teacher is important for distillation.

Impact of Longer Training Regime:

Training the student for longer epochs (up to 300 epochs on the SceneFlow dataset like Tankovich *et al.* (2021)) leads to the student matching the teacher performance. However, we observed that this leads to overfitting and poor generalization capabilities across the datasets. Moreover, training for these many epochs required three weeks on our computer setup, so this was not explored any further.

6.4.4 Results

This section presents the quantitative and qualitative results of the final student model(s) on different datasets. Following Tankovich *et al.* (2021); Bangunharcana *et al.* (2021); Xu *et al.* (2022) we perform comparisons at two levels, *i.e.* (i) with the performance oriented methods (having runtime $>$ 100 *ms*); and (ii) with speed oriented $3\mathcal{D}$ stereo methods.

SceneFlow:

Table 6.6 (top-half) compares the results of our trained DSNet+Attention with teacher and performance oriented state-of-the-art stereo matching methods on the SceneFlow test set. We can observe that despite having 2 to 3× less MACs and parameters our method performs better than state of the art methods like LEAStereo, CFNet, GwcNet, *etc*. Moreover, this student network gives comparable performance compared to the teacher network ACVNet while having 3× fewer parameters and operations. With respect to model size, our network also has the smallest memory footprint (7MB) compared to all the other methods tested.

Target	Method	EPE (*px*)	MACs (*G*)	Params (*M*)	Red. MACs	Red. Params
Performance	GCNet (Kendall *et al.*, 2017)	1.84	718.01	3.18	7.9×	1.7×
	PSMNet (Chang and Chen, 2018)	0.88	256.66	5.22	2.8×	2.8×
	GwcNet (Guo *et al.*, 2019a)	0.79	246.27	6.43	2.7×	3.5×
	GANetdeep (Zhang *et al.*, 2019)	0.84	670.25	6.58	7.3×	3.5×
	CFNet (Shen *et al.*, 2021)	0.97	177.50	23.05	1.9×	12.4×
	LEAStereo (Cheng *et al.*, 2020)	0.78	156.5	**1.81**	1.7×	1.0×
	ACVNet (T) (Xu *et al.*, 2022)	**0.49**	240.05	6.23	2.6×	3.4×
	DSNet+Attention (Student)	0.55	**91.33**	1.86	-	-
Speed	DeepPrunerFast (Duggal *et al.*, 2019)	0.97	51.83	7.47	0.8×	4.7×
	AANet (Xu and Zhang, 2020)	0.87	188.76	8.44	2.8×	5.3×
	DecNet† (Yao *et al.*, 2021)	0.84	-	-	-	-
	BGNet (Xu *et al.*, 2021)	1.17	**20.33**	5.32	0.3×	3.3×
	CoEx (Bangunharcana *et al.*, 2021)	0.69	61	2.73	0.9×	1.7×
	Fast-ACVNet† (Xu *et al.*, 2022)	0.64	-	-	-	-
	HITNet† (Tankovich *et al.*, 2021)	**0.43**	146.01	**0.97**	1.6×	0.5×
	DSNet (Student)	0.62	67.20	1.60	-	-

Table 6.6: Comparison of our student networks with performance and speed oriented state-of-the-art stereo methods on the SceneFlow test set. MACs and Params calculations of all these methods have been reproduced using identical size input images. †without public implementation. Red. indicates the reduction factor of our models compared to other methods.

Table 6.6 (bottom-half) compares DSNet with other speed oriented methods. Here DSNet gives comparable performance to the existing methods while being faster. This validates that using our proposed distillation pipeline one can train a faster and leaner network with better performance.

Fig. 6.5 shows qualitative results of our method on SceneFlow test set in comparison with other existing methods. From the disparity errors we can see that our method gives

reliable disparity estimates that appears better than GwcNet, they are comparable with ACVNet.

KITTI:

Tables 6.7 and 6.8 compare the result of our submitted method with other methods on the KITTI benchmark datasets. For KITTI2012, we report 4-*px* and 3-*px* (non-occluding (noc) and all) errors – *c.f*. Table 6.7. We report the D1 error for foreground *(fg)*, background *(bg)* and all pixels from the benchmark page of KITTI2015 – *c.f*. Table 6.8. We can draw similar conclusions as in the case of the SceneFlow dataset. Compared to performance oriented methods, optimized (TensorRT FP16) DSNet+Attention is significantly faster (*e.g*. 6× relative to ACVNet) with reasonable performance – in fact it gives identical performance to CFNet and GANet while being 5× faster. Compared to speed oriented methods, DSNet has consistently lower errors on both datasets compared to all other real-time stereo methods including Fast-ACVNet (Xu *et al.*, 2022), AANet (Xu and Zhang, 2020) and DeepPrunerFast (Duggal *et al.*, 2019), while running at 40 FPS.

Target	Method	3-*px* (noc)	3-*px* (all)	4-*px* (noc)	4-*px* (all)
Performance	GCNet (Kendall *et al.*, 2017)	1.77	2.30	1.36	1.77
	PSMNet (Chang and Chen, 2018)	1.49	1.89	1.12	1.42
	GwcNet (Guo *et al.*, 2019a)	1.32	1.70	0.99	1.27
	GANetdeep (Zhang *et al.*, 2019)	1.19	1.60	0.91	1.23
	CFNet (Shen *et al.*, 2021)	1.23	1.58	0.92	1.18
	LEAStereo (Cheng *et al.*, 2020)	**1.13**	**1.45**	**0.83**	**1.08**
	ACVNet (Xu *et al.*, 2022)	**1.13**	1.47	0.86	1.12
Speed	DeepPrunerFast (Duggal *et al.*, 2019)	-	-	-	-
	AANet (Xu *et al.*, 2022)	1.91	2.42	1.46	1.87
	DecNet (Yao *et al.*, 2021)	-	-	-	-
	BGNet (Xu *et al.*, 2021)	1.77	2.15	-	-
	CoEx (Bangunharcana *et al.*, 2021)	1.55	1.93	1.15	1.42
	Fast-ACVNet (Xu *et al.*, 2022)	1.68	2.13	1.23	1.56
	HITNet (Tankovich *et al.*, 2021)	1.41	1.89	1.14	1.53
	DSNet (Student)	1.36	1.79	1.03	1.37
	DSNet+Attention (Student)	**1.22**	**1.63**	**0.93**	**1.24**

Table 6.7: Quantitative comparison of our student networks with performance and speed oriented state-of-the-art stereo methods on the KITTI 2012 (Geiger *et al.*, 2012) benchmark dataset.

Qualitative results on the KITTI benchmark are shown in Fig. 6.6.

Target	Method	$D1_{bg}$	$D1_{fg}$	$D1_{all}$	Runtime (ms)
Performance	GCNet (Kendall *et al.*, 2017)	2.21	6.16	2.87	900
	PSMNet (Chang and Chen, 2018)	1.86	4.62	2.32	310‡
	GwcNet (Guo *et al.*, 2019a)	1.74	3.93	2.11	180‡
	GANetdeep (Zhang *et al.*, 2019)	1.48	3.46	1.81	180
	CFNet (Shen *et al.*, 2021)	1.54	3.56	1.88	180
	LEAStereo (Cheng *et al.*, 2020)	1.40	**2.91**	**1.65**	300
	ACVNet (T) (Xu *et al.*, 2022)	**1.37**	3.07	**1.65**	200‡
Speed	DeepPrunerFast (Duggal *et al.*, 2019)	2.32	3.91	2.59	50‡
	AANet (Xu *et al.*, 2022)	1.99	5.39	2.55	62
	DecNet (Yao *et al.*, 2021)	2.07	3.87	2.37	50
	BGNet (Xu *et al.*, 2021)	2.07	4.74	2.51	28‡
	CoEx (Bangunharcana *et al.*, 2021)	1.79	3.82	2.13	33‡
	Fast-ACVNet (Xu *et al.*, 2022)	1.82	3.93	2.17	39‡
	HITNet (Tankovich *et al.*, 2021)	1.74	**3.20**	1.98	54‡
	DSNet (Student)	1.70	3.56	2.01	**25**
	DSNet+Attention (Student)	**1.56**	3.84	**1.94**	**37**

Table 6.8: Quantitative comparison of our student networks with performance and speed oriented state-of-the-art stereo methods on KITTI2015 (Menze and Geiger, 2015) benchmark datasets. ‡denotes the runtime on identical hardware (NVIDIA RTX 3090) with same image sizes.

Method	KITTI 2012 D1-all	KITTI 2015 D1-all	Middlebury 2-*px*	ETH3D 1-*px*
PSMNet (Chang and Chen, 2018)	15.1	16.3	39.5	23.8
GWCNet (Guo *et al.*, 2019a)	12.0	12.2	37.4	11.0
CasStereo (Gu *et al.*, 2020)	11.8	11.9	40.6	**7.8**
GANet (Zhang *et al.*, 2019)	10.1	11.7	32.2	14.1
DeepPrunerFast (Duggal *et al.*, 2019)	16.8	15.9	30.8	-
BGNet (Xu *et al.*, 2021)	24.8	20.1	37.0	-
CoEx (Bangunharcana *et al.*, 2021)	13.5	11.6	25.5	-
Fast-ACVNet (Xu *et al.*, 2022)	12.4	10.6	20.1	-
DSNet (S)	**8.6**	**8.8**	**14.24**	9.2

Table 6.9: Generalization performance of student network on cross-domain datasets. Here the network is only trained on the SceneFlow training dataset and evaluated on these diverse real worlds datasets.

Generalization Capacity of the Student Network

To see how well our student networks generalize across the datasets and domains, we have evaluated our model performance trained on synthetic SceneFlow datasets on real-world datasets. Specifically, our model trained on the SceneFlow dataset (Mayer *et al.*, 2016) is used to estimate the disparity maps for KITTI (Geiger *et al.*, 2012; Menze and Geiger, 2015), ETH3D (Schops *et al.*, 2017) and Middlebury (Scharstein *et al.*, 2014) datasets. Table 6.9 shows that DSNet generalizes extremely well and outperform all the tested methods. Fig. 6.7 shows qualitative generalization results of the student model (DSNet) trained on the SceneFlow dataset and evaluated on unseen dataset images. We can once again observe that using the proposed knowledge distillation pipeline one can train a student network that not only performs well on the trained dataset but also generalizes well across the datasets.

6.5 Conclusions

In this work we have presented a novel knowledge distillation pipeline by combining insights from stereo methods with general knowledge-distillation techniques. To this end, we have done a thorough systematic study of different design choices for building a leaner and faster stereo network with good performance. We find that distilling stereo networks requires careful design of the backbone along with the right selection of distillation points and loss functions. Our trained student networks rival performance-oriented methods while offering significantly faster execution. Moreover, when compared to speed oriented methods they give comparable performance. As the presented pipeline is generic it can be easily used as a baseline for trimming down real world $3\mathcal{D}$ stereo networks. Since no public existing work is already available on this topic we believe this work can serve as a baseline to build further strong knowledge distillation methods for stereo networks.

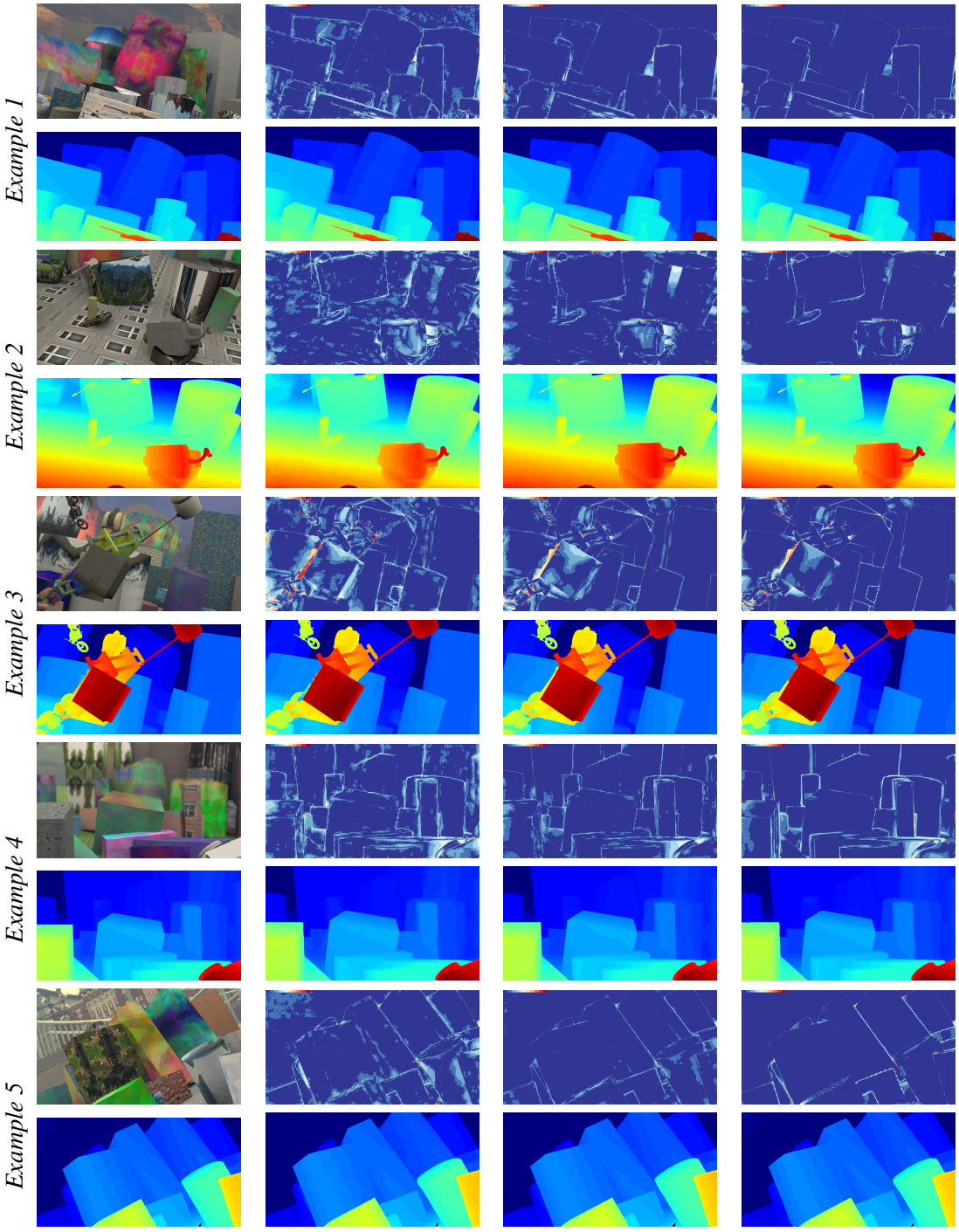


Figure 6.5: Qualitative results on SceneFlow test images. Here the first, third and fifth rows represent the left input images and error maps *w.r.t.* ground truth (GT). Seocnd, third and last rows show GT and predicted disparity maps. In error maps, darker red means high disparity errors and as the color gets more blue (darker blue) it represents lower disparity errors.

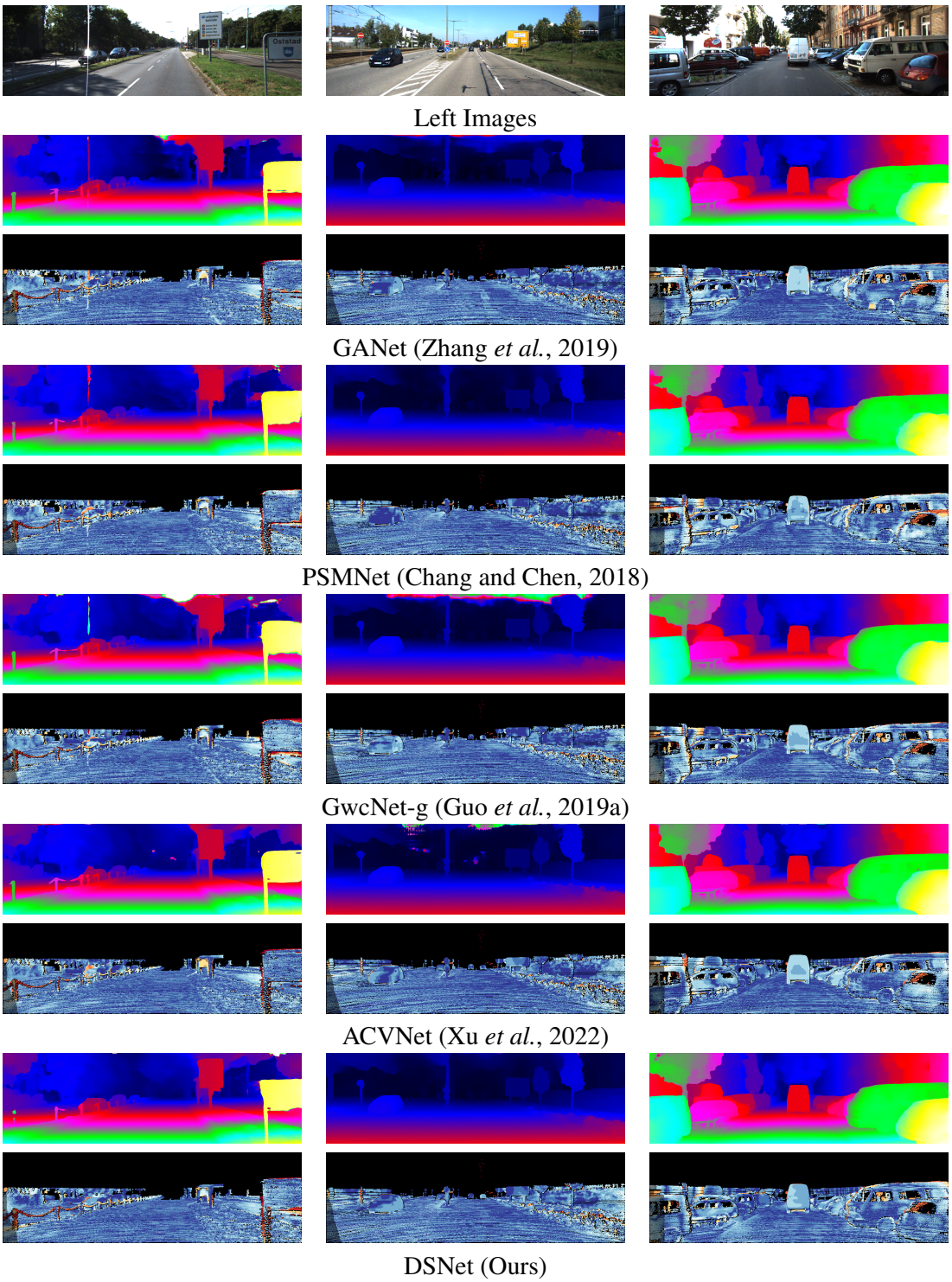


Figure 6.6: Results (disparity images together with error maps) of running student network on KITTI 2015 benchmark dataset – these images are downloaded from the benchmark website. Warmer colors in error maps denote larger error values.

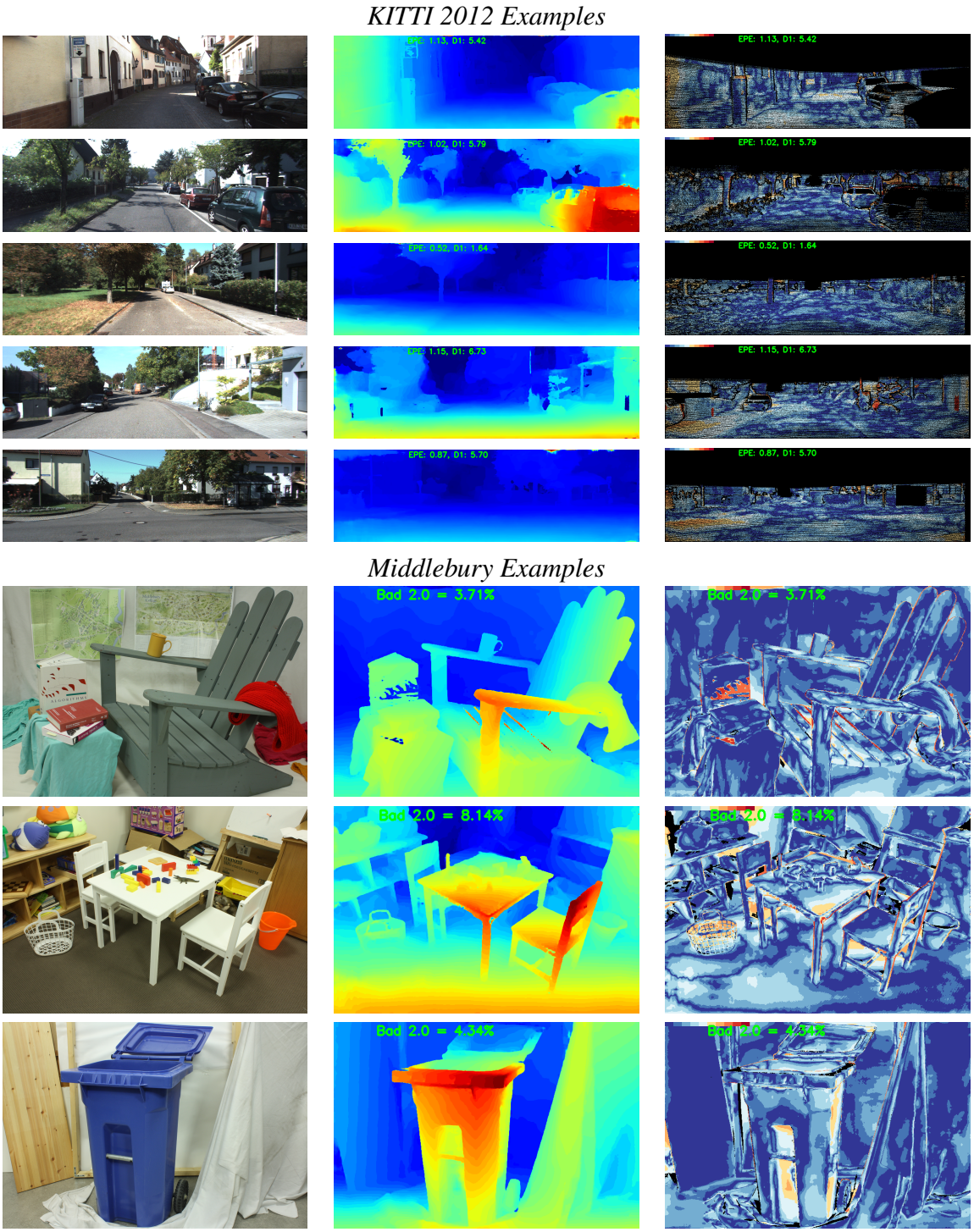


Figure 6.7: Generalization results of DSNet, trained on the SceneFlow dataset and evaluated on KITTI and Middlebury datasets.

Chapter 7

Conclusions

Existing stereo methods can be broadly classified into two categories: two-dimensional ($2\mathcal{D}$) or three-dimensional ($3\mathcal{D}$) methods, based on the rank of convolution kernels used in later stages to fuse information from the left and right views. While $3\mathcal{D}$ methods outperform $2\mathcal{D}$ ones significantly, they suffer from substantial computational requirements. This limits their practicality for real-world applications, as they are both bulky and slow to execute.

In this thesis, our primary focus was on optimizing the model size (measured by the number of parameters) and inference runtime (measured by the number of floating-point operations) of the methods investigated. To accomplish this objective, we employed a two-pronged approach. Firstly, we dedicated our efforts to exploring lightweight alternatives for individual network operators, aiming to optimize the networks at the component level. Concurrently, we introduced novel lightweight end-to-end models at the other end of the spectrum.

In Chapter 3, we conducted a thorough profiling of $3\mathcal{D}$ stereo networks to identify their key performance bottlenecks. Our analysis revealed that the extensive use of three-dimensional convolutions, which serve as the fundamental building blocks, are the primary reason for higher computational requirements of these models. Notably, in the case of GANetdeep, approximately 94% of the floating-point operations are consumed by the module that incorporates $3\mathcal{D}$ convolutions. Based on these findings, we propose the adoption of light-weight separable convolutions as a replacement for the computationally intensive three-dimensional convolutions. Given that $3\mathcal{D}$ convolutions operate on a $4\mathcal{D}$ volume, we thoroughly investigated different options to achieve separability for reducing their computational footprint. Specifically, we explored feature-wise separability (referred to as feature-wise separable $3\mathcal{D}$ convolution or FwSC), disparity-wise separability (referred to as disparity-wise separable $3\mathcal{D}$ convolution or DwSC), and combined feature- and disparity-wise separability (referred to as feature-disparity-wise separable $3\mathcal{D}$ convolution or FDwSC). Our comprehensive evaluations consistently demonstrate that FwSC surpasses the performance of three-dimensional convolutions when applied to state-of-the-art stereo methods. Moreover, FwSC achieves remarkable reductions of up to approximately $7\times$ in terms of operations and $4\times$ in terms of parameters compared to three-dimensional convolutions. This noteworthy improvement can be attributed to

the existence of redundant information within the four-dimensional cost volume, where three-dimensional separable convolutions possess adequate representation capabilities to capture the crucial information necessary for disparity regression. While FDwSC also yields promising results on both the SceneFlow and KITTI datasets, its performance slightly lags behind that of FwSC.

In Chapter 4, we built on the work done in Chapter 3 and explored the light-weight $2\mathcal{D}$ separable convolutions based modules for building a pair of optimized stereo networks, called as $2\mathcal{D}$-MobileStereoNet and $3\mathcal{D}$-MobileStereoNet. Both of these networks share a common backbone and are built using efficient $2\mathcal{D}$ separable convolutional blocks. However, they differ in how they merge and process information for disparity regression in later stages of the network. For example, $2\mathcal{D}$-MobileStereoNet combines left and right image features to construct a $2\mathcal{D}$ cost volume and utilizes $2\mathcal{D}$ convolutions in the encoder-decoder part of the network. In contrast, $3\mathcal{D}$-MobileStereoNet constructs a $4\mathcal{D}$ volume and employs $3\mathcal{D}$ convolutions in the encoder-decoder module. To further reduce computational requirements, we replace the $3\mathcal{D}$ convolutions in these encoder-decoder modules with their separable counterparts. These blocks were selected after a detailed thorough empirical investigation. Overall, $2\mathcal{D}$-MobileStereoNet achieves a lower error rate compared to other methods while having fewer parameters and operations. Similarly, $3\mathcal{D}$-MobileStereoNet, despite having significantly fewer parameters and operations, delivers competitive performance relative to other $3\mathcal{D}$ stereo methods.

In contrast to our earlier works, which focused on optimizing individual operators and layers using separable convolutions, our later works explored the replacement of entire network backbones with more efficient alternatives, to further reduce computational requirements. In Chapter 5, we introduced LeanStereo, a lightweight network. LeanStereo is designed to be leaner but still having enough modeling capacity. This is accomplished through the use of a two-branch backbone architecture, consisting of a shallow branch and a deep branch. The shallow branch focuses on modeling pixel-level information and is designed with fewer layers but a higher number of feature maps per layer. This configuration allows for fine-grained details to be captured effectively. In contrast, the deep branch is responsible for modeling semantic-level information and consists of a relatively higher number of layers but with fewer feature maps per layer. The information from both the left and right branches is combined using a specially designed aggregation layer before being passed into an attention-based cost volume. Our experiments demonstrated that incorporating the attention-based cost volume was effective in mitigating the performance degradation caused by the use of a leaner backbone. Besides we found that using LogL1 loss with this proposed architecture leads to better performance. Overall, the results indicate that LeanStereo exhibits competitive performance in comparison to other $3\mathcal{D}$ stereo methods, while achieving a remarkable speedup of $9\times$ to $27\times$.

In Chapter 6, we continued our journey into design and exploration of more efficient methods and leverage knowledge distillation for building leaner networks. Our main objective was to train a student network that achieves performance on par with the teacher network without the need for extensive architectural modifications. To achieve this, we

adopted a systematic approach that combined insights from stereo methods and knowledge distillation techniques, leading to the development of DSNet. Throughout our research, we made several key observations. We found that when distilling knowledge from the teacher to the student network in stereo networks, it is essential to use multiple distillation points. Furthermore, the choice of objective function at each distillation point plays a crucial role in encouraging the desired information transfer. Notably, our approach yielded leaner and faster networks without compromising performance. For instance, when compared to performance-oriented state-of-the-art approaches such as PSMNet (Chang and Chen, 2018), CFNet (Shen *et al.*, 2021), and LEAStereo (Cheng *et al.*, 2020) on the SceneFlow dataset, our student network outperforms them significantly and exhibits speeds that are $8\times$, $5\times$, and $8\times$ faster, respectively. Furthermore, when evaluated against speed-oriented methods, our student network surpasses all tested methods in KITTI benchmark, demonstrating superior performance, while maintaining faster processing speeds. Overall, our work highlights the importance of knowledge distillation and our novel approach in designing efficient stereo networks with superior performance and faster processing speeds.

In short, our work on optimizing stereo methods has significant implications for practical applications. While our methods have been tested on server-grade GPUs, we believe they have the potential to be adapted for edge devices due to their generality. An important future direction is to extend our work and conduct detailed analysis specifically for edge devices. This avenue holds promise for further advancements in deploying efficient stereo vision systems in real-world scenarios.

Abbreviations

Acronym	Meaning
1-*px*	One-Pixel Error
2-*px*	Two-Pixel Error
3-*px*	Three-Pixel Error
4-*px*	Four-Pixel Error
CA	Cost-Aggregation
CNN	Convolutional Neural Network
CV	Cost-Volume
DB	Deep Branch
DNN	Deep Neural Network
DWSCs	Disparity-Wise Separable Convolutions
EPE	End Point Error
FE	Feature Extraction
FB	Feature-Based
FLOPs	Floating Point Operations
GMACs	Giga Multiply Accumulates
GPU	Graphics Processing Unit
KLD	Kullback-Leibler Divergence
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
LiDAR	Light Detection and Ranging
MACs	Multiply Accumulates
MSE	Mean Squared Error
Ops	Operations (GMACs)
Params	Parameters (M)
PSMNet	Pyramid Stereo Matching Network
RMSE	Root Mean Squared Error
ResNet	Residual Network
SB	Shallow Branch
SGM	Semi-Global Matching
SPW	Student Pixel-Wise Loss
STPW	Student Teacher Pixel-Wise Loss
SfM	Structure from Motion

Acronym	**Meaning**
SceneFlow	Scene Flow Datasets
ToF	Time of Flight

Bibliography

Anandan, P. (1989). A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, **2**(3), 283–310.

Bangunharcana, A., Cho, J. W., Lee, S., Kweon, I. S., Kim, K.-S., and Kim, S. (2021). Correlate-and-Excite: Real-time stereo matching via guided cost volume excitation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3542–3548. IEEE.

Batsos, K., Cai, C., and Mordohai, P. (2018). CBMV: A coalesced bidirectional matching volume for disparity estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 2060–2069.

Beyer, L., Zhai, X., Royer, A., Markeeva, L., Anil, R., and Kolesnikov, A. (2022). Knowledge Distillation: A good teacher is patient and consistent. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 10925–10934.

Bleyer, M. and Gelautz, M. (2005). A layered stereo matching algorithm using image segmentation and global visibility constraints. *ISPRS Journal of Photogrammetry and remote sensing*, **59**(3), 128–150.

Chang, J.-R. and Chen, Y.-S. (2018). Pyramid Stereo Matching Network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 5410–5418, Salt Lake City, UT. IEEE.

Chen, G., Choi, W., Yu, X., Han, T., and Chandraker, M. (2017). Learning efficient object detection models with knowledge distillation. *Advances in neural information processing systems (NIPS)*, **30**.

Chen, P., Liu, S., Zhao, H., and Jia, J. (2021). Distilling knowledge via knowledge review. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 5008–5017.

Cheng, X., Zhong, Y., Harandi, M., Dai, Y., Chang, X., Li, H., Drummond, T., and Ge, Z. (2020). Hierarchical neural architecture search for deep stereo matching. *Advances in neural information processing systems (NIPS)*, **33**, 22158–22169.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 248–255.

Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., and Brox, T. (2015). FlowNet: Learning optical flow with convolutional networks. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2758–2766.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). CARLA: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR.

Duggal, S., Wang, S., Ma, W.-C., Hu, R., and Urtasun, R. (2019). DeepPruner: Learning efficient stereo matching via differentiable patchmatch. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 4384–4393.

Egnal, G. (2000). Mutual information as a stereo correspondence measure. Technical Report MS-CIS-00-20, Computer and Information Science, University of Pennsylvania, Philadelphia, USA.

Feichtenhofer, C. (2020). X3D: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 203–213.

Gao, Q., Zhou, Y., Li, G., and Tong, T. (2020). Compact StereoNet: stereo disparity estimation via knowledge distillation and compact feature extractor. *IEEE Access*, **8**, 192141–192154.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 3354–3361. IEEE.

Godard, C., Mac Aodha, O., Firman, M., and Brostow, G. J. (2019). Digging into self-supervised monocular depth estimation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 3828–3838.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., *et al.* (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems (NIPS)*, **33**, 21271–21284.

Gu, X., Fan, Z., Zhu, S., Dai, Z., Tan, F., and Tan, P. (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 (CVPR)*, pages 2495–2504.

Guo, X., Yang, K., Yang, W., Wang, X., and Li, H. (2019a). Group-wise correlation stereo network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 3273–3282.

Guo, X., Yang, K., Yang, W., Wang, X., and Li, H. (2019b). Group-wise correlation stereo network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 3273–3282.

Hartley, R. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 770–778.

Hinton, G., Vinyals, O., Dean, J., *et al.* (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, **2**(7).

Hirschmuller, H. (2003). *Stereo Vision Based Mapping and Immediate Virtual Walkthroughs*. Ph.D. thesis, School of Computing, De Montfort University, Leicester, UK.

Hirschmuller, H. (2005). Accurate and efficient stereo processing by semi-global matching and mutual information. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, volume 2, pages 807–814.

Hirschmuller, H. (2007). Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, **30**(2), 328–341.

Hirschmüller, H., Innocent, P. R., and Garibaldi, J. (2002). Real-time correlation-based stereo vision with reduced border errors. *International Journal of Computer Vision*, **47**(1), 229–246.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hu, J., Ozay, M., Zhang, Y., and Okatani, T. (2019). Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. In *2019 IEEE winter conference on applications of computer vision (WACV)*, pages 1043–1051. IEEE.

Hua, Y., Kohli, P., Uplavikar, P., Ravi, A., Gunaseelan, S., Orozco, J., and Li, E. (2020). Holopix50k: A large-scale in-the-wild stereo image dataset. In *CVPR Workshop on Computer Vision for Augmented and Virtual Reality, Seattle, WA, 2020.*

Huang, X., Wang, P., Cheng, X., Zhou, D., Geng, Q., and Yang, R. (2019). The apolloscape open dataset for autonomous driving and its application. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, **42**(10), 2702–2719.

Ilg, E., Saikia, T., Keuper, M., and Brox, T. (2018). 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 (ECCV)*, pages 614–630.

Kanade, T., Kano, H., Kimura, S., Yoshida, A., and Oda, K. (1995). Development of a video-rate stereo machine. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 3, pages 95–100. IEEE.

Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A., and Bry, A. (2017). End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 66–75.

Khamis, S., Fanello, S., Rhemann, C., Kowdle, A., Valentin, J., and Izadi, S. (2018). Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 573–590.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Klaus, A., Sormann, M., and Karner, K. (2006). Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 15–18. IEEE.

Kolmogorov, V. and Zabih, R. (2001). Computing visual correspondence with occlusions using graph cuts. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 2, pages 508–515. IEEE.

Lee, J.-H., Heo, M., Kim, K.-R., and Kim, C.-S. (2018). Single-image depth estimation based on fourier domain analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 330–339.

Liang, Z., Feng, Y., Guo, Y., Liu, H., Chen, W., Qiao, L., Zhou, L., and Zhang, J. (2018). Learning for disparity estimation through feature constancy. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 2811–2820.

Liu, B., Yu, H., and Long, Y. (2022). Local similarity pattern and cost self-reassembling for deep stereo matching networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 1647–1655.

Liu, Y., Shu, C., Wang, J., and Shen, C. (2020). Structured knowledge distillation for dense prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

Loop, C. and Zhang, Z. (1999). Computing rectifying homographies for stereo vision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, volume 1, pages 125–131.

Loshchilov, I., Hutter, F., *et al.* (2017). Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, **5**.

Luo, P., Zhu, Z., Liu, Z., Wang, X., and Tang, X. (2016a). Face model compression by distilling knowledge from neurons. In *Thirtieth AAAI conference on artificial intelligence*.

Luo, W., Schwing, A. G., and Urtasun, R. (2016b). Efficient deep learning for stereo matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 5695–5703.

Maddern, W., Pascoe, G., Linegar, C., and Newman, P. (2017). 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, **36**(1), 3–15.

Matthies, L., Kanade, T., and Szeliski, R. (1989). Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, **3**(3), 209–238.

Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., and Brox, T. (2016). A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 4040–4048.

Menze, M. and Geiger, A. (2015). Object scene flow for autonomous vehicles. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 3061–3070.

Pang, J., Sun, W., Ren, J. S., Yang, C., and Yan, Q. (2017). Cascade Residual Learning: A two-stage convolutional neural network for stereo matching. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, pages 887–895.

Park, W., Kim, D., Lu, Y., and Cho, M. (2019). Relational knowledge distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 3967–3976.

Passalis, N. and Tefas, A. (2018). Learning deep representations with probabilistic knowledge transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 268–284.

Poggi, M., Tosi, F., Batsos, K., Mordohai, P., and Mattoccia, S. (2021). On the synergies between machine learning and binocular stereo for depth estimation from images: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, **44**(9), 5314–5334.

Qiu, Z., Yao, T., and Mei, T. (2017). Learning spatio-temporal representation with pseudo-3d residual networks. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 5533–5541.

Rahim, R., Shamsafar, F., and Zell, A. (2021). Separable convolutions for optimizing 3d stereo networks. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3208–3212. IEEE.

Rao, Z., He, M., Dai, Y., Zhu, Z., Li, B., and He, R. (2020). NLCA-Net: A non-local context attention network for stereo matching. *APSIPA Transactions on Signal and Information Processing*, **9**, e18.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems (NIPS)*, **28**.

Saikia, T., Marrakchi, Y., Zela, A., Hutter, F., and Brox, T. (2019). AutoDispNet: Improving disparity estimation with automl. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1812–1823.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 4510–4520.

Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, **47**, 7–42.

Scharstein, D., Hirschmüller, H., Kitajima, Y., Krathwohl, G., Nešić, N., Wang, X., and Westling, P. (2014). High-resolution stereo datasets with subpixel-accurate ground truth. In *German conference on pattern recognition (GCPR)*, pages 31–42. Springer.

Schonberger, J. L., Sinha, S. N., and Pollefeys, M. (2018). Learning to fuse proposals from multiple scanline optimizations in semi-global matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 739–755.

Schops, T., Schonberger, J. L., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M., and Geiger, A. (2017). A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 3260–3269.

Seki, A. and Pollefeys, M. (2017). SGM-Nets: Semi-global matching with neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 231–240.

Shaked, A. and Wolf, L. (2017). Improved stereo matching with constant highway networks and reflective confidence learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 4641–4650.

Shamsafar, F., Woerz, S., Rahim, R., and Zell, A. (2022). MobileStereoNet: Towards lightweight deep networks for stereo matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2417–2426.

Shen, Z., Dai, Y., and Rao, Z. (2021). CFNet: Cascade and fused cost volume for robust stereo matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 13906–13915.

Shen, Z., Dai, Y., Song, X., Rao, Z., Zhou, D., and Zhang, L. (2022). PCW-Net: Pyramid combination and warping cost volume for stereo matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 280–297. Springer.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Song, X., Zhao, X., Hu, H., and Fang, L. (2019). EdgeStereo: A context integrated residual pyramid network for stereo matching. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part V 14*, pages 20–35. Springer.

Sun, J., Li, Y., Kang, S. B., and Shum, H.-Y. (2005). Symmetric stereo matching for occlusion handling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, volume 2, pages 399–406. IEEE.

Szeliski, R. (2022). *Computer vision: algorithms and applications*. Springer Nature.

Tan, M. and Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR.

Tankovich, V., Hane, C., Zhang, Y., Kowdle, A., Fanello, S., and Bouaziz, S. (2021). HITNet: Hierarchical iterative tile refinement network for real-time stereo matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 14362–14372.

Tao, T., Koo, J. C., and Choi, H. R. (2008). A fast block matching algorthim for stereo correspondence. In *2008 IEEE Conference on Cybernetics and Intelligent Systems*, pages 38–41.

Tonioni, A., Tosi, F., Poggi, M., Mattoccia, S., and Stefano, L. D. (2019). Real-time self-adaptive deep stereo. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 195–204.

Tulyakov, S., Ivanov, A., and Fleuret, F. (2018). Practical deep stereo (PDS): Toward applications-friendly deep stereo matching. *Advances in neural information processing systems (NIPS)*, **31**, 5871–5881.

Viola, P. and Wells III, W. M. (1997). Alignment by maximization of mutual information. *International journal of computer vision*, **24**(2), 137–154.

Wang, R. J., Li, X., and Ling, C. X. (2018). Pelee: A real-time object detection system on mobile devices. *Advances in neural information processing systems (NIPS)*, **31**, 1963–1972.

Wang, Y., Lai, Z., Huang, G., Wang, B. H., Van Der Maaten, L., Campbell, M., and Weinberger, K. Q. (2019). Anytime stereo image depth estimation on mobile devices. In *2019 international conference on robotics and automation (ICRA)*, pages 5893–5900. IEEE.

Watson, J., Firman, M., Brostow, G. J., and Turmukhambetov, D. (2019). Self-supervised monocular depth hints. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2162–2171.

Wu, Z., Wu, X., Zhang, X., Wang, S., and Ju, L. (2019). Semantic stereo matching with pyramid cost volumes. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 7484–7493.

Wulff, J., Butler, D. J., Stanley, G. B., and Black, M. J. (2012). Lessons and insights from creating a synthetic optical flow benchmark. In *Computer Vision–ECCV 2012. Workshops and Demonstrations: Florence, Italy, October 7-13, 2012, Proceedings, Part II 12*, pages 168–177. Springer.

Xie, Q., Luong, M.-T., Hovy, E., and Le, Q. V. (2020). Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 10687–10698.

Xu, B., Xu, Y., Yang, X., Jia, W., and Guo, Y. (2021). Bilateral grid learning for stereo matching networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 12497–12506.

Xu, G., Cheng, J., Guo, P., and Yang, X. (2022). ACVNet: Attention concatenation volume for accurate and efficient stereo matching. *arXiv preprint arXiv:2203.02146*.

Xu, H. and Zhang, J. (2020). AANet: Adaptive aggregation network for efficient stereo matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 1959–1968.

Yang, G., Zhao, H., Shi, J., Deng, Z., and Jia, J. (2018). SegStereo: Exploiting semantic information for disparity estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 636–651.

Yang, G., Song, X., Huang, C., Deng, Z., Shi, J., and Zhou, B. (2019). DrivingStereo: A large-scale dataset for stereo matching in autonomous driving scenarios. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 899–908.

Yang, Q., Wang, L., Yang, R., Stewénius, H., and Nistér, D. (2008). Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, **31**(3), 492–504.

Yao, C., Jia, Y., Di, H., Li, P., and Wu, Y. (2021). A decomposition model for stereo matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 6091–6100.

Ye, R., Liu, F., and Zhang, L. (2019). 3d depthwise convolution: Reducing model parameters in 3d vision tasks. In *Advances in Artificial Intelligence: 32nd Canadian Conference on Artificial Intelligence, Canadian AI 2019, Kingston, ON, Canada, May 28–31, 2019, Proceedings 32*, pages 186–199. Springer.

Yee, K. and Chakrabarti, A. (2020). Fast deep stereo with 2d convolutional processing of cost signatures. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 183–191.

Yoon, K.-J. and Kweon, I. S. (2006). Adaptive support-weight approach for correspondence search. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, **28**(4), 650–656.

Yu, C., Gao, C., Wang, J., Yu, G., Shen, C., and Sang, N. (2021). Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *International Journal of Computer Vision*, **129**(11), 3051–3068.

Žbontar, J. and LeCun, Y. (2016). Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, **17**(65), 1–32.

Zhang, F., Prisacariu, V., Yang, R., and Torr, P. H. (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 (CVPR)*, pages 185–194.

Zhang, X., Zhou, X., Lin, M., and Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 6848–6856.