December 2020

Understanding the Dynamic Visual World: From Motion to Semantics

Huaizu Jiang
CICS, UMass Amherst

UNDERSTANDING THE DYNAMIC VISUAL WORLD: FROM MOTION TO SEMANTICS

A Dissertation Presented

by

HUAIZU JIANG

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2020

College of Information and Computer Sciences

UNDERSTANDING THE DYNAMIC VISUAL WORLD: FROM MOTION TO SEMANTICS

A Dissertation Presented

by

HUAIZU JIANG

Approved as to style and content by:

Erik Learned-Miller, Chair

Subhransu Maji, Member

Liangliang Cao, Member

Mario Parente, Member

Deqing Sun, Member

James Allan, Chair
College of Information and Computer Sciences

DEDICATION

To my wife, Xiaoqiang Yan.

ACKNOWLEDGMENTS

I knew getting a Ph.D. would be difficult. But I didn't expect it was so tough, at least for the first couple of years. I published my first thesis-research paper in the middle of my third year. My first thesis project got published after three submissions, though one of the reviewers still gave negative comments in the third submission. Fortunately, I have gone through it with the help and support I have received from so many people, without which this dissertation would not have been possible.

First and foremost, I would like to thank my dear advisor, Prof. Erik Learned-Miller, for his guidance, patience, and encouragement. Erik's door is always open to me. There were countless times where I knocked his door while he was working on some other stuff and he always put aside what he was doing and devoted his energy to my problems. Erik is a great researcher. I am always impressed by his rigorous and original thinking as well as his insights about what important problems are in computer vision. I often wish I knew more about math, especially statistics, when I had discussions with Erik. His continuing long-term effort on developing algorithms to utilize unlabeled data also led me to believe the great potential of training machines to perceive the visual surroundings without relying on manual annotations, which we have seen recently in both computer vision and natural language processing. Erik is not only a great advisor in research, but also a mentor and friend personally. I remember I had to go back to China several times during my first year when my family was in China and made little progress on research. Erik was still patient and believed in me. He also gave me tremendous encouragement, help, and support for my faculty application, where he spent great effort revising my application materials. Erik is my role model on how to advise students.

I knew Dr. Deqing Sun when I worked on optical flow estimation back before staring my Ph.D. study. I was very excited to collaborate with him during my first internship at NVIDIA. Since then, we started long-term collaboration, where Deqing effectively served as my co-advisor. I am impressed by Deqing's deep knowledge and first-hand experiences in dynamic scene understanding and synthesis, especially in the field of optical flow estimation. He gave me numerous support for my fellowship and faculty applications, and also invited me to give a talk in a tutorial he co-organized. He even gave me a mock interview around Christmas in 2019, which helped me survive most of telephone interviews for faculty application. It's my great fortunate to work with Deqing during my Ph.D. journey.

I am grateful to my other committee members. Thanks to the collaboration with Prof. Subhransu Maji, I had an opportunity to investigate vision-language problems, which I am very passionate about. It's always fun and inspiring to chat and collaborate with Subhransu, where he often enlightened me with his sharp and original thinking. Prof. Liangliang Cao is a great mentor, collaborator, and friend. He gave me constructive comments about my thesis and presentation. I also thank him for inviting me to give a talk at Google NYC. I greatly appreciate valuable suggestions from Prof. Mario Parente on how to improve the presentation of my Ph.D. research, which played a critical role for my faculty application.

I am proud to be part of the Computer Vision and Graphics Lab at UMass Amherst, where I met a lot of energetic and eager young minds. I would like to thank: Aruni RoyChowdhury, Hang Su, SouYong Jin, Jong-Chyi Su, Chenyun Wu, Tsung-Yu Lin, Pia Bideau, Matheus Gadelha, Ashish Singh, Sreenivas Venkobarao, Zezhou Cheng, Gopal Sharma, Yang Zhou, Zhan Xu, Difan Liu, Dmitrii Petrov, Pratheba Selvaraju, Chetan Manjesh, Zitian Chen, Gustavo Perez, Zhaoliang Lun, Haibin Huang, Li Yang Ku, and Mikayla Timm. Aruni has the amazing ability to organize a team force working on a problem together. We co-authored two papers

and I enjoyed our collaboration. Thank you, SouYoung, for the joy you brought to the lab and particularly for those birthday cakes, songs, and photos. I benefited a lot from conversation with Hang, from implementation tricks to the big picture of computer vision research. I am surprised though that we haven't officially collaborated yet. I look forward to collaborating with you soon, Hang. Jong-Chyi, Tsung-Yu, and Zezhou are my best buddies to play pool (billiards) with. I spent almost every weekend of July and August of 2020 with Tsung-Yu playing pool when we interned together at Facebook. I miss those old days with you guys. My ECCV 2018 paper, which was my first official thesis project, was based on a Pia's method. Thank you, Pia, for patiently explaining to me the relationship between optical flow's angle and magnitude fields, camera motion, and motion segmentation. Thank you for collaborating with me, Chenyun and Jong-Chyi, on my first ever vision-language paper. It is always fun and inspiring to chat with Matheus. I often learned new stuff from him, particularly GAN and gaussian process. I collaborated with Ashish on an exciting project to mine visual commonsense knowledge. I appreciate his devotion to the project and having me as a collaborator. Thank you, Gopal, for organizing our lab meetings. I worked with Sreenivas on video interpolation as his master project. I hope to work with you again, Sreenivas. Thank you everyone in the Computer Vision and Graphics Lab for those sweet memories. I miss our group lunch and group dinner.

A number of people make College of Information and Computer Sciences (CICS) at UMass Amherst a great place to work in. I would like to thank Leeanne Leclerc, Eileen Hamel, and Malaika Ross for their help over different milestones toward getting my Ph.D. degree. Their great work made my life significantly easier. I'd also like to thank Laurie Downey, whom I always counted on to register conference trips and get reimbursement afterwards. I took the Reinforcement Learning course and later did my Synthesis Project (equivalent to the Quialify Exam) with Prof. Philip Thomas. Phil is one of the best teachers I met at UMass, where he can always clearly illustrate

not only the details but also the very motivation behind each algorithm. I am also grateful to the valuable suggestions from Laura Haas and James Allan for my faculty job offer. I met a lot of friends at CICS: Yue Wang, Qingyao Ai, Keping Bi, Xiaolan Wang, Dan Zhang, Pan Hu, Prof. Jiafeng Guo, Liu Yang, Bo Jiang, and Kun Tu. They made my life at Amherst much more joyful.

Most of work in this dissertation is joint work with my collaborators. I thank Prof. Greg Shakhnarovich, Prof. Michael Maire, and Dr. Gustav Larsson to collaborate with me in the early stage of my Ph.D. study. My gratitude goes to them for their valuable comments and suggestions as well as going through the tough reviewing process together with me. I did two joyful internships at NVIDIA. I would like to thank my mentors including Dr. Deqing Sun, Dr. Varun Jampani, Prof. Ming-Hsuan Yang, and Dr. Jan Kautz, for giving me the flexibility to work on the problems that I am passionate about. I am grateful to Zhaoyang Lv, who taught me the beauty of utilizing geometric constraints for dynamic scene understanding. I did another internship at Facebook AI Research (FAIR), collaborating with Dr. Xinlei Chen, Dr. Ishan Misra, and Dr. Marcus Rohrbach. I appreciate they gave me the offer to work on a research problem that is outside of my core research area. I learned a lot from Dr. Xinlei Chen on how to manage large-scale experiments. I also thank Dr. Xinlei Chen for having me on the winning entry of the VQA Challenge 2020. My sincere gratitude particularly goes to Dr. Varun Jampani, Prof. Ming-Hsuan Yang, and Dr. Xinlei Chen for writing reference letters for me.

I acknowledge the support from Adobe Fellowship and NVIDIA Graduate Fellowship. It was amazing to receive both of them in the same year. Thank you, Adobe and NVIDIA! Your support significantly boosted my career.

I am forever grateful to the unconditioned love and life-long support from my parents, parents-in-law, grandparents, and younger brother. I particularly thank my daughter, Cheng-Cheng. It's fascinating and much more rewarding to raise a truly

intelligent agent. Training our puppy, Chase, makes me believe that reinforcement learning does make sense. Finally, thanks goes to my wife, Xiaoqiang Yan, for her love, support, and scarifies along this tough journey. This dissertations is dedicated to her.

ABSTRACT

UNDERSTANDING THE DYNAMIC VISUAL WORLD: FROM MOTION TO SEMANTICS

SEPTEMBER 2020

HUAIZU JIANG

B.E., XI'AN JIAOTONG UNIVERSITY

M.E., XI'AN JIAOTONG UNIVERSITY

M.Sc., UNIVERSITY OF MASSACHUSETTS, AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Erik Learned-Miller

We live in a dynamic world, which is continuously in motion. Perceiving and interpreting the dynamic surroundings is an essential capability for an intelligent agent. Human beings have the remarkable capability to learn from limited data, with partial or little annotation, in sharp contrast to computational perception models that rely on large-scale, manually labeled data. Reliance on strongly supervised models with manually labeled data inherently prohibits us from modeling the dynamic visual world, as manual annotations are tedious, expensive, and not scalable, especially if we would like to solve multiple scene understanding tasks at the same time. Even worse, in some cases, manual annotations are completely infeasible, such as the motion vector of each pixel (*i.e.*, optical flow) since humans cannot reliably produce these types of labeling. In fact, living in a dynamic world, when we move around, motion

information, as a result of moving camera, independently moving objects, and scene geometry, consists of abundant information, revealing the structure and complexity of our dynamic visual world. As the famous psychologist James J. Gibson suggested, "we must perceive in order to move, but we also must move in order to perceive". In this thesis, we investigate how to use the motion information contained in unlabeled or partially labeled videos to better understand and synthesize the dynamic visual world.

This thesis consists of three parts. In the first part, we focus on the "move to perceive" aspect. When moving through the world, it is natural for an intelligent agent to associate image patterns with the magnitude of their displacement over time: as the agent moves, far away mountains don't move much; nearby trees move a lot. This natural relationship between the appearance of objects and their apparent motion is a rich source of information about the relationship between the distance of objects and their appearance in images. We present a pretext task of estimating the relative depth of elements of a scene (*i.e.*, ordering the pixels in an image according to distance from the viewer) recovered from motion field of unlabeled videos. The goal of this pretext task was to induce useful feature representations in deep Convolutional Neural Networks (CNNs). These induced representations, using 1.1 million video frames crawled from YouTube within one hour without any manual labeling, provide valuable starting features for the training of neural networks for downstream tasks. It is promising to match or even surpass what ImageNet pre-training gives us today, which needs a huge amount of manual labeling, on tasks such as semantic image segmentation as all of our training data comes almost for free.

In the second part, we study the "perceive to move" aspect. As we humans look around, we do not solve a single vision task at a time. Instead, we perceive our surroundings in a holistic manner, doing visual understanding using all visual cues jointly. By simultaneously solving multiple tasks together, one task can influence

another. In specific, we propose a neural network architecture, called SENSE, which shares common feature representations among four closely-related tasks: optical flow estimation, disparity estimation from stereo, occlusion detection, and semantic segmentation. The key insight is that sharing features makes the network more compact and induces better feature representations. For real-world data, however, not all annotations of the four tasks mentioned above are always available at the same time. To this end, loss functions are designed to exploit interactions of different tasks and do not need manual annotations, to better handle partially labeled data in a semi-supervised manner, leading to superior understanding performance of the dynamic visual world.

Understanding the motion contained in a video enables us to perceive the dynamic visual world in a novel manner. In the third part, we present an approach, called SuperSloMo, which synthesizes slow-motion videos from a standard frame-rate video. Converting a plain video into a slow-motion version enables us to see memorable moments in our life that are hard to see clearly otherwise with naked eyes: a difficult skateboard trick, a dog catching a ball, etc. Such a technique also has wide applications such as generating smooth view transition on a head-mounted virtual reality (VR) devices, compressing videos, synthesizing videos with motion blur, etc.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS . . . . . **v**

ABSTRACT . . . . . **x**

LIST OF TABLES . . . . . **xvi**

LIST OF FIGURES . . . . . **xviii**

CHAPTER

1. INTRODUCTION **1**

1.1 Motivation 2
1.2 Overview of the Dissertation 7
1.3 Other Work not in the Dissertation 10

2. A REVIEW OF OPTICAL FLOW ESTIMATION AND BEYOND **12**

2.1 Optical Flow Estimation 12

2.1.1 Classical Approaches 13
2.1.2 Neural Approaches 18
2.1.3 Discussions 22

2.2 From Optical Flow to 3D Scene Understanding 25

3. SELF-SUPERVISED RELATIVE DEPTH LEARNING FOR URBAN SCENE UNDERSTANDING **32**

3.1 Overview 32
3.2 Related Work 36
3.3 Inducing features by learning to estimate relative depth 38

3.3.1 Self-Supervised Relative Depth 38

3.3.2 Predicting Relative Depth From a Single Image 41

3.4 Experiments 42

3.4.1 Semantic Segmentation 42
3.4.2 Ablation Studies 44
3.4.3 Domain Adaptation by Pre-Training 45
3.4.4 Joint Semantic Reasoning 47
3.4.5 Monocular Absolute Depth Estimation 49

3.5 Conclusions and Discussions 50

4. SENSE: A SHARED ENCODER FOR SCENE FLOW ESTIMATION 52

4.1 Overview 52
4.2 Related Work 55
4.3 Semi-Supervised Scene Flow Estimation 59

4.3.1 Modular Network Design 59
4.3.2 Semi-Supervised Loss 62
4.3.3 Rigidity-based Warped Disparity Refinement for Scene Flow Estimation 67

4.4 Experiments 70

4.4.1 Implementation Details 70
4.4.2 Main Results 74
4.4.3 Ablation Studies 76

4.5 Conclusion 78

5. SUPER SLOMO: HIGH QUALITY ESTIMATION OF MULTIPLE INTERMEDIATE FRAMES FOR VIDEO INTERPOLATION 79

5.1 Overview 79
5.2 Related Work 81
5.3 Proposed Approach 83

5.3.1 Intermediate Frame Synthesis 84
5.3.2 Arbitrary-time Flow Interpolation 85
5.3.3 Training 91

5.4 Experiments 92

5.4.1 Dataset 92

5.4.2 Ablation Studies 95
5.4.3 Comparison with state-of-the-art methods 97
5.4.4 Unsupervised Optical Flow 102

5.5 Conclusion 102

6. CONCLUSION AND FUTURE WORK 103

6.1 Future Work 104

BIBLIOGRAPHY 106

LIST OF TABLES

Table **Page**

3.1 Comparisons of mean IoU scores of AlexNet FCN32s for semantic segmentation using different self-supervised models. CS=CityScapes, K=KITTI, CV=CamVid. 43

3.2 Mean IoU scores of semantic segmentation using different architectures on different datasets. CD=CityDriving, CS=CityScapes, CV=CamVid, and K=KITTI. 45

3.3 Results of joint semantic reasoning, including road segmentation and car detection. 48

3.4 Monocular depth estimation on the KITTI dataset using the split of Eigen *et al.* [27] (range of 0-80m). For model details, Arch.=Architecture, A=AlexNet, V=VGG16, and R=ResNet50. For training data, Class.=classification, I=ImageNet, CD=CityDriving, K=KITTI, CS=CityScapes. **pp** indicates test-time augmentation by horizontally flipping the input image. 50

4.1 Average EPE results on MPI Sintel optical flow dataset. "-ft" means fine-tuning on the MPI Sintel *training* set and the numbers in parentheses are results on the data the methods have been fine-tuned on. 72

4.2 Results on the KITTI optical flow dataset. "-ft" means fine-tuning on the KITTI *training* set and the numbers in the parenthesis are results on the data the methods have been fine-tuned on. 73

4.3 Results on KITTI stereo datasets (test set). 74

4.4 Results on KITTI2015 Scene flow dataset. CNN-based approaches need to deal with refinement of D2, where N and R indicates network and rigidity-based refinement, respectively. 75

4.5 Effectiveness of different tasks. 76

4.6 Ablation study of different loss terms. 77

5.1 Statistics of dataset we use to train our network. 92

5.2 Effectiveness of multi-frame video interpolation on the *Adobe240-fps* dataset. 94

5.3 Effectiveness of different components of our model on the *Adobe240-fps* dataset. 95

5.4 Results on the *UCF101* dataset. 96

5.5 Results on the *slowflow* dataset. 97

5.6 Results on the high-frame-rate *Sintel* dataset. 97

5.7 Optical flow results on the KITTI 2012 benchmark. 102

LIST OF FIGURES

Figure **Page**

1.1 Motion magnitude in the image plane is inversely proportional to an object's depth. (a) frame #291 of a video sequence, (b) frame #292 of the same video sequence, and (c) magnitude of pixels' motion vector (*i.e.*, optical flow) between (a) and (b), where intensity encodes the magnitude. Darker means the motion magnitude is larger. It can be see that an object moves faster in the image plane if it is closer to the camera. 3

1.2 Understanding the surroundings' motion is critical for an autonomous driving car. From top to bottom: (a) a typical setting of an autonomous driving car with stereo RGB cameras and a laser scanner mounted on the top, (b)(c) colorized ground-truth labels of optical flow and stereo disparity overlaid on top of RGB images, respectively. 5

1.3 Given two input images at time steps $T = 0$ and $T = 1$, knowing the motion of every pixel allows us to synthesize a video frame at an intermediate time step $T = t \in (0, 1)$ along the motion trajectory. 6

1.4 SuperSloMo converts a plain video into slow-motion by synthesizing multiple intermediate frames (two are shown here enclosed with red color). More demo videos are available at https://youtu.be/MjViy6kyiqs and https://tinyurl.com/vlrp8br. 9

2.1 Optical flow estimation. (a)(b) two consecutive video frames from [76], (c) an illustration of optical flow, where the motion vector of each pixel is represented by an arrow, and (d) a visualization of the colorized optical flow using the color key shown in (e), where the hue indicates motion direction and saturation corresponds to the motion magnitude. 13

2.2 Illustration of the encoder-decoder network structure widely used for optical flow estimation. The feature representation of two input images are obtained using the same encoder, which are then fed into a decoder to compute the optical flow. 19

2.3 End-point-error (EPE) of different optical flow estimation methods versus their published years on the MPI Sintel test set (final pass) [17]. 23

2.4 End-point-error (EPE) of different optical flow estimation methods versus their inference speed in terms of number of frames to process per second (FPS). At the same time, the size of the circle indicates the number of parameters for each method (larger means more parameters). 24

2.5 An illustration of the perspective projection model. The optical center of a camera is located at the origin O. There are two coordinate systems in the 3D world and the 2D image plane. The image plane is f away from the optical center of camera, which is an intrinsic parameter of the camera. A point P in the 3D world is projected onto the image plane at point q. 26

2.6 A typical configuration for scene flow estimation. Four images are available as input that are taken by two stereo cameras at two consecutive time steps. For math symbols, a subscript denotes the time step and a superscript indicates whether the image is taken by the left or the right camera. The goal is to infer the 3D motion of every pixel in the reference image (bottom left). 29

3.1 Sample frames from collected videos and their corresponding relative depth maps, where brightness encodes relative depth (the brighter the farther). From top to bottom: input image, relative depth image computed using Eq.(3.3), and predicted (relative) depth maps using our trained VGG16 FCN8s [122, 121]. There is often a black blob around the center of the image, a singularity in depth estimation caused by the focus of expansion. (a)(b)(c): images from the CityDriving dataset, (d): images from the KITTI dataset, and (e): images from the CityScapes dataset. 33

3.2 Samples of image pairs and computed translational optical flow that we use to recover the relative depth. From left to right: first images, second images, translational optical flow between input two images, and relative depth of the first images. 40

3.3 Ablation studies of performance by (a) varying number of pre-training images on KITTI and CamVid, (b) varying number of fine-tuning images of CityScapes. 45

3.4 Qualitative semantic segmentation results on CityScapes. From top to bottom: input images, predictions of FCN8s with no pre-training, our FCN8s pre-trained on CityDriving, our FCN8s pre-trained on CityDriving adapted to CityScapes, ImageNet FCN8s, and ground-truth annotations. The difference between the 2nd and 3rd rows shows a clear benefit of pre-training with relative depth prediction. The difference between 3rd rows and 4th rows shows the benefit our unsupervised domain adaptation using pre-training. 46

4.1 Given stereo videos, we train compact networks for several holistic scene understanding problems by sharing features. 53

4.2 Illustration of network design. Dashed arrows indicate shared weights. We have a single encoder for all input images and all different tasks and keep different decoders for different tasks. On the right, from top to bottom are: optical flow, forward occlusion mask, semantic segmentation, disparity, and disparity occlusion. The PPM (Pyramid Pooling Module) is not helpful for optical flow estimation. But thanks to the modular network design, we can flexibly configure the network. 58

4.3 Effects of adding distillation losses for semantic segmentation (middle) and occlusion (right) to the supervised loss. 63

4.4 Illustration of effectiveness of self-supervised loss. From top to bottom: input images, disparity estimations *without* using self-supervised loss, and disparity estimations *with* using self-supervised loss. We can see self-supervised loss helps greatly reduce artifacts in the sky region. 64

5.1 Illustration of intermediate optical flow approximation. The orange pixel borrows optical flow from pixels at the same position in the first and second images. 85

5.2 Samples of flow interpolation results, where $t = 0.5$. The entire scene is moving toward the left (due to camera translation) and the motorcyclist is independently moving left. The last row shows that the refinement from our flow interpolation CNN is mainly around the motion boundaries (the whiter a pixel, the bigger the refinement). 87

5.3 Samples of predicted visibility maps (best viewed in color), where $t=0.5$. The arms move downwards from $T=0$ to $T=1$. So the area right above the arm at $T=0$ is visible at t but the area right above the arm at $T=1$ is occluded (*i.e.*, invisible) at t. The visibility maps in the fourth row clearly show this phenomenon. The white area around arms in $V_{t\leftarrow 0}$ indicate such pixels in I_0 contribute most to the synthesized $\hat{I}_t$ while the occluded pixels in I_1 have little contribution. Similar phenomena also happen around motion boundaries (*e.g.*, around bodies of the athletes). 88

5.4 Network architecture of our approach. 89

5.5 Illustration of the architecture of our flow computation and flow interpolation CNNs. 90

5.6 Snapshot of our training data. 93

5.7 Performance comparisons on each sequence of the Middlebury dataset. Numbers are obtained from the Middlebury evaluation server. 96

5.8 Visual results of a sample from UCF101. Our model produces less artifacts around the brush and the hand (best viewed in color). Please see the supplementary material for more image and *video* results. From left to right: (a) two input images, (b) actual in-between, (c) PhaseBased [93], (d) FlowNet2 [6, 48], (e) DVF [79], (f) SepConv [97], and (g) ours. 98

5.9 Visual comparisons on the UCF101 dataset. (a) Ground truth in-between frame, interpolation results from (b) PhaseBased [93], (c) FlowNet2 [38, 48], (d) SepConv [97], (e) DVF [79], and (f) Ours. 99

5.10 Visual comparisons on the UCF101 dataset. (a) Ground truth in-between frame, interpolation results from (b) PhaseBased [93], (c) FlowNet2 [38, 48], (d) SepConv [97], (e) DVF [79], and (f) Ours. 100

5.11 PSNR at each time step when generating 31 intermediate frames on the high-frame-rate Sintel dataset. 101

CHAPTER 1

INTRODUCTION

Building intelligent systems is a long-standing grand goal for artificial intelligence researchers. Among all the capabilities, visual perception is a core skill set that allows such an intelligent system to perceive and interpret the visual surroundings through cameras like what we humans do from our eyes. For example, it is essential for an autonomous driving car to understand which part of a scene corresponds to objects like vehicles and pedestrians on the road, how far away they are, and how fast they move. Such basic visual perception skills enable further capabilities of a visually intelligent agent such as interacting with the surroundings and its peers. For instance, a visual navigation agent that looks for a TV in an apartment needs to understand where the free space is so as to avoid bumping into a wall. Such a navigation agent also needs to recognize objects in the apartment in order to find meaningful cues related to the target to decide which way to go to efficiently reach the destination.

Since the historical breakthrough of deep Convolutional Neural Networks [67] in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 [116], we have achieved significant success for visual perception research during the past decade, thanks to large-scale manually labeled data and massive computational resources (*e.g.*, GPUs). Almost every sub-field of computer vision has been dominated by deep learning models now, where the paradigm has shifted from designing hand-crafted visual features to learning the entire visual perception model in an end-to-end manner[1]. Nowadays, a visual perception model is able to recognize the species of

[1]In most cases, manually designing the network architecture is still necessary.

animals contained in images, segment an image into different instances, including even small objects like bottles, and also tell us where drivable area is from just a single image.

All of these success, however, are heavily dependent on *manually* labeled data. For example, we need to annotate millions of images to tell a visual perception model that what images are about dogs and what others are about cats. By sharp contrast, human beings have the remarkable capability to learn from limited data, with partial or little annotation. Reliance on strongly supervised models with manually labeled data inherently prohibits us from modeling the dynamic visual world, as manual annotations are tedious, expensive, and not scalable, especially if we would like to solve multiple scene understanding tasks at the same time. Even worse, in some cases, manual annotations are completely infeasible, such as the motion vector of each pixel (*i.e.*, optical flow) since humans cannot reliably produce these types of labeling.

1.1 Motivation

We live in a dynamic world, which is continuously in motion. Living in such a dynamic world, when we move around, motion information, as a result of a moving camera, independently moving objects, and scene geometry, consists of abundant information, revealing the structure and complexity of our dynamic visual world. We humans can effortless use the motion information to infer the structure of the 3D world and even control ourselves to walk [149]. As the famous psychologist James J. Gibson suggested, "we must perceive in order to move, but we also must move in order to perceive" [33]. In this thesis, we investigate how to use the motion information contained in unlabeled or partially labeled videos to reduce the reliance on manual annotations, which leads to a more precise understanding and synthesis of the dynamic visual world.

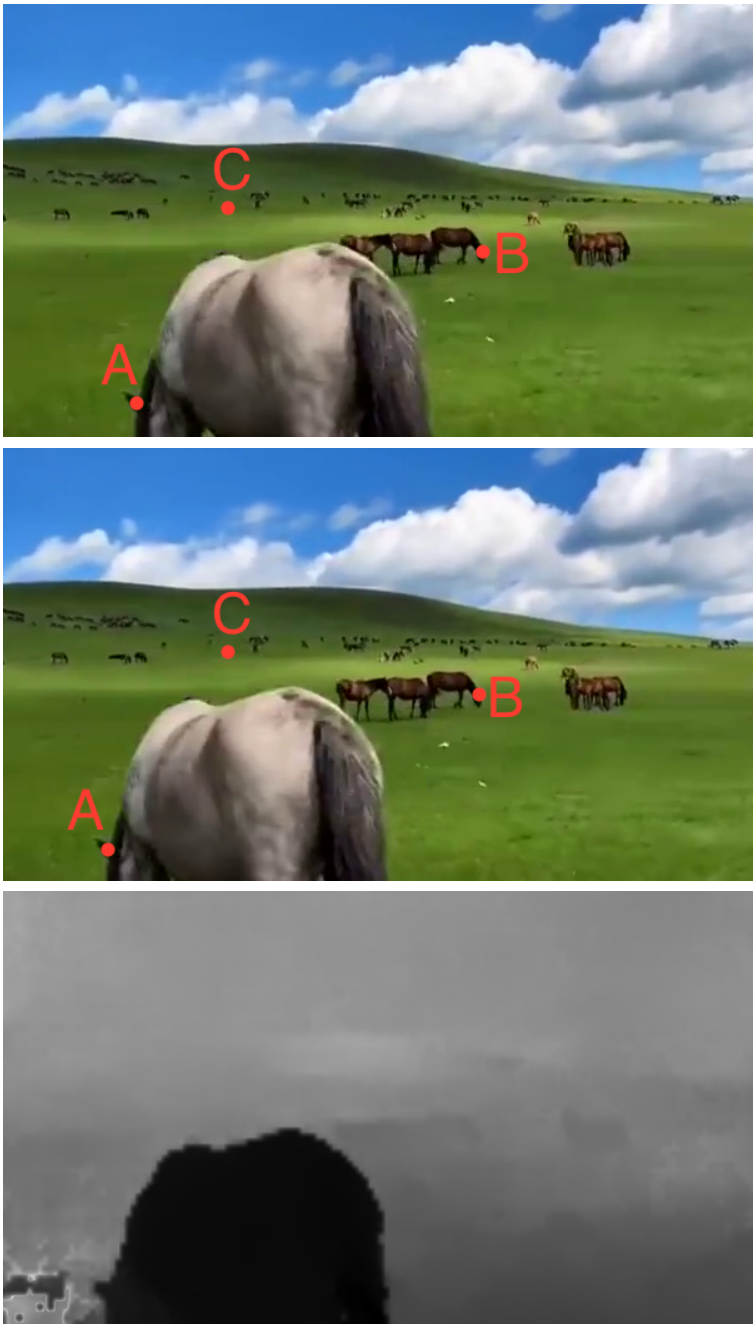
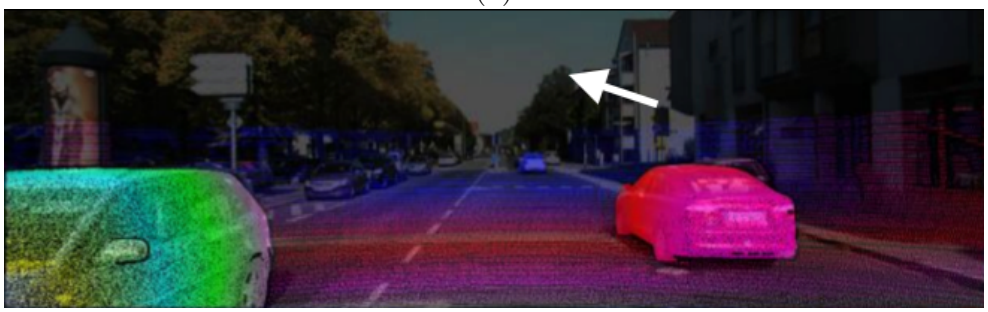


Figure 1.1: Motion magnitude in the image plane is inversely proportional to an object's depth. (a) frame #291 of a video sequence, (b) frame #292 of the same video sequence, and (c) magnitude of pixels' motion vector (*i.e.*, optical flow) between (a) and (b), where intensity encodes the magnitude. Darker means the motion magnitude is larger. It can be see that an object moves faster in the image plane if it is closer to the camera.

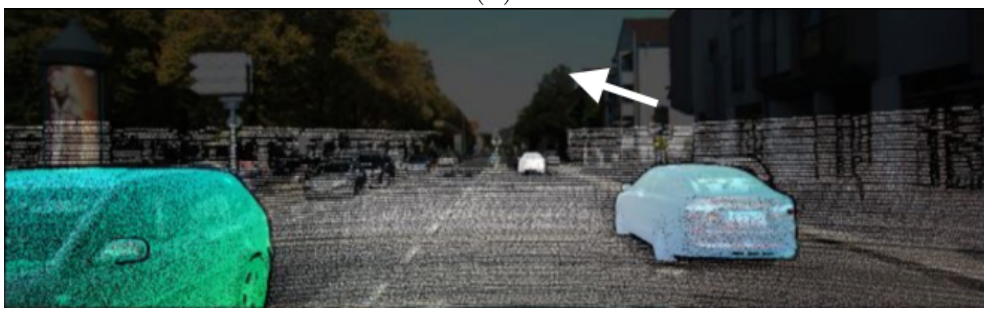
First of all, we examine *how to train a (virtual) agent to perceive the visual surroundings by wandering around*. When an agent, like an animal or a robot moves, its visual system is exposed to a shower of information. Usually, the speed with which something moves in the image is inversely proportional to its depth. As shown in Fig. 1.1, the point A, which is closer to the camera, moves about 50 pixels between two consecutive video frames, while the further point B moves only about 5 pixels. The furthest point C almost does not move at all. As an agent continues to experience visual stimuli under its own motion, it is natural for it to form associations between the appearance of objects and their relative motion in the image plane. For example, an agent may learn that objects that look like mountains typically don't move in the image (or change appearance much) as the agent moves. Objects like nearby buildings and bushes, however, appear to move rapidly in the image as the agent changes position relative to them. This continuous pairing of images with motion acts as a kind of automatic supervision that could eventually allow an agent both to understand the depth of objects and to group pixels into objects by this predicted depth. Thus, by moving through the world, an agent may learn to obtain useful representations to perceive *static* scenes, for instance, segmenting an image into different semantic regions and also detecting objects such as pedestrians and cars.

We also study *how to teach machines, such as an autonomous driving car or a navigation robot, to understand the visual surroundings's motion in order to move safely*. It needs to understand where objects are, such as other vehicles and pedestrians, how far away they are, and how fast they move. Fig. 1.2(a) shows a typical configuration of an autonomous driving car [32], where two RGB stereo cameras and a laser scanner are mounted on top of the car. Motion information of the scene can be recovered from the captured videos by computing optical flow and two images taken by the two stereo images provide depth cues via stereo disparity, both of which can be combined to recover the 3D motion of the surroundings. Optical flow and stereo dis-

(a)



(b)



(c)

Figure 1.2: Understanding the surroundings' motion is critical for an autonomous driving car. From top to bottom: (a) a typical setting of an autonomous driving car with stereo RGB cameras and a laser scanner mounted on the top, (b)(c) colorized ground-truth labels of optical flow and stereo disparity overlaid on top of RGB images, respectively.

parity ground-truth labels are inherently infeasible to collect for human annotators. As a result, laser points provided by the scanner are usually used to generate the ground-truth annotations. Due to the capturing range of the laser scanner, however, such annotations are usually sparse. As shown in Fig. 1.2(b) and (c), ground-truth

labels are consistently missing in the top region of the RGB images. Even worse, for the bottom part, only less than 20% of pixels have ground-truth labels, which poses grand challenge for training a visual perception model. On the other hand, humans own remarkable capability to learn from sparse data, which may stem in part from our ability to interpret the dynamic visual world holistically. When interpreting a scene, people simultaneously infer many properties such as depth, motion, location, and the semantic categories of different elements. Solving one task is often helpful in solving others; for example, a car's motion is likely to be rigid while a person's movement is non-rigid. Thus, motion could provide a cue for object identification, or conversely, knowledge of the object could help us interpret the motion.

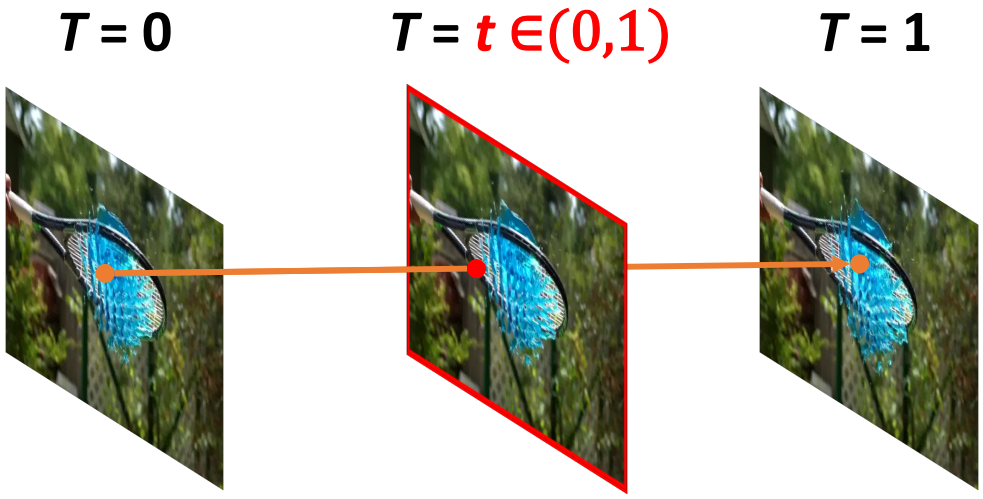


Figure 1.3: Given two input images at time steps $T = 0$ and $T = 1$, knowing the motion of every pixel allows us to synthesize a video frame at an intermediate time step $T = t \in (0, 1)$ along the motion trajectory.

Finally, understanding the motion contained in a video enables us to *synthesize the dynamic visual scenes in a novel manner that is otherwise hard to clearly see by our naked eyes*. For instance, there are many moments in our lives that we might want to record with a camera in slow- motion because they are either memorable or otherwise hard to see clearly with our naked eyes: the first time a baby walks, a

difficult skateboard trick, a dog catching a ball, etc. While it is possible to take 240 frame-per-second (fps) videos with a cell phone nowadays, professional high-speed cameras are still required for higher frame rates, which is unaffordable for an unprofessional consumer. Besides, many of the moments we would like to record in a slow-motion mode are unpredictable, and as a result, are recorded at standard frame rates. Recording everything at high frame rates is impractical–it requires large memories and is power-intensive for mobile devices. We can resort to a visual perception model to solve this problem. As shown in Fig. 1.3, we can synthesize intermediate video frames along the estimated motion trajectory between two input images.

1.2 Overview of the Dissertation

There are three major parts in this thesis, corresponding to the three problems raised in the previous section.

Move to perceive. In Chapter 3, we present a proxy (surrogate or pre-text) task where we train a deep neural network to learn visual representations from relative scene depth recovered from motion field of unlabeled videos. We start by training a deep network, using fully automatic supervision, to predict relative scene depth from single images. The relative depth training images are automatically derived from simple videos of cars moving through a scene, using recent motion segmentation techniques, and no human-provided labels. The proxy task of predicting relative depth from a single image induces features in the network that result in large improvements in a set of downstream tasks including semantic segmentation, joint road segmentation and car detection, and monocular (absolute) depth estimation, over a network trained from scratch. The improvement on the semantic segmentation task is greater than that produced by any other automatically supervised methods. Moreover, for monocular depth estimation, our unsupervised pre-training method even outperforms supervised pre-training with ImageNet. In addition, we demonstrate benefits from

learning to predict (again, completely unsupervised) relative depth in the specific videos associated with various downstream tasks (e.g., KITTI). We adapt to the specific scenes in those tasks in an unsupervised manner to improve performance. In summary, for semantic segmentation, we present state-of-the-art results among methods that do not use supervised pre-training, and we even exceed the performance of super-vised ImageNet pre-trained models for monocular depth estimation, achieving results that are comparable with state-of-the-art method

Perceive to move. We present a semi-supervised approach in Chapter 4 to estimate 3D motion of the scene (known as scene flow estimation). We introduce a compact network for holistic scene flow estimation, called SENSE, which shares common encoder features among four closely-related tasks: optical flow estimation, disparity estimation from stereo, occlusion estimation, and semantic segmentation. Our key insight is that sharing features makes the network more compact, induces better feature representations, and can better exploit interactions among these tasks to handle partially labeled data.With a shared encoder, we can flexibly add decoders for different tasks during training. This modular design leads to a compact and efficient model at inference time. Exploiting the interactions among these tasks allows us to introduce distillation and self-supervised losses in addition to super-vised losses, which can better handle partially labeled real-world data. SENSE achieves state-of-the-art results on several optical flow benchmarks and runs as fast as networks specifically designed for optical flow. It also compares favorably against the state of the art on stereo and scene flow,while consuming much less memory.

Motion-based slow-motion video synthesis. In Chapter 5, we introduce an approach, called SuperSloMo, to synthesize multiple intermediate frames to generate both spatially and temporally coherent slow-motion videos (known as video interpolation), as shown in Fig. 1.4. While most existing methods focus on single-frame interpolation, we propose an end-to-end convolutional neural network for variable-

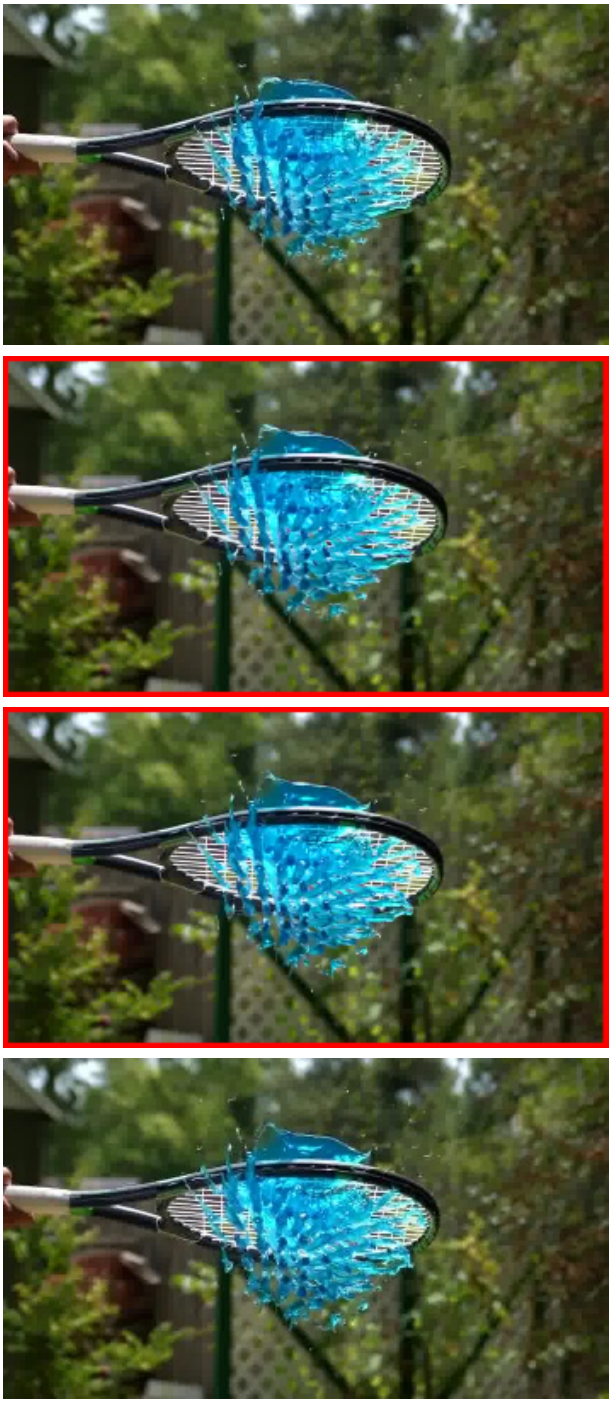


Figure 1.4: SuperSloMo converts a plain video into slow-motion by synthesizing multiple intermediate frames (two are shown here enclosed with red color). More demo videos are available at https://youtu.be/MjViy6kyiqs and https://tinyurl.com/vlrp8br.

length multi-frame video interpolation, where the motion interpretation and occlusion reasoning are jointly modeled. We start by computing bi-directional optical flow between the input images using a U-Net architecture.These flows are then linearly combined at each time step to approximate the intermediate bi-directional optical flows.These approximate flows, however, only work well in locally smooth regions and produce artifacts around motion boundaries. To address this shortcoming, we employ another U-Net to refine the approximated flow and also predict soft visibility maps. Finally, the two input images are warped and linearly fused to form each intermediate frame. By applying the visibility maps to the warped images before fusion,we exclude the contribution of occluded pixels to the interpolated intermediate frame to avoid artifacts. Since none of our learned network parameters are time-dependent, our approach is able to produce as many intermediate frames as needed. To train our network, we use 1,132 240-fps video clips, containing 300K individual video frames. Experimental results on several datasets, predicting different numbers of interpolated frames, demonstrate that our approach per-forms consistently better than existing methods.

1.3 Other Work not in the Dissertation

During my doctoral training, I also studied two other major problems that are not covered in this thesis.

First of all, together with my collaborators, we investigated object detection, where the goal is to localize objects from different categories in a single image, such as human faces [59]. Yet such object detectors still make mistakes even with high confidence, which are called hard examples. Moreover, the domain gap between training data and testing data often leads to inferior detection results. How can we improve the object detector? Instead of manually annotating more data to re-train the model, which is expensive and tedious, my colleagues and I used temporal consistency

in video frames to identify those hard examples automatically in [63]. The object detector is then fine-tuned using automatically mined hard examples. Better object detection performance can be achieved on faces, pedestrians, and other categories, even when the testing data is significantly different from the training data [115].

The other major problem I had chances to visit is to *use natural language as a scaffold to advance visual perception.* Natural language, as an effective tool, is often used to explain and describe what people perceive about the visual world. When explaining visual scenes in terms of question-answer pairs (*i.e.*, Visual Question Answering) and describing a visual scene using natural language, various visual perception capabilities are needed, such as human action recognition and object localization. Together with my collaborators, we demonstrated that visual perception skills emerge when training an agent to explain and describe visual scenes using natural language [126, 60].

CHAPTER 2

A REVIEW OF OPTICAL FLOW ESTIMATION AND BEYOND

In this chapter, we briefly review classical and latest trends of using neural networks for optical flow estimation, which is a building block for the entire thesis, as well as other downstream tasks, particularly 3D scene understanding.

2.1 Optical Flow Estimation

Optical flow, as a result of the relative motion of objects and camera, captures every single pixel's motion in terms of movement of brightness patterns in the image plane from one video frame to another. In specific, given a video sequence I the optical flow $(u_{x,y,t}, v_{x,y,t})$ at the pixel (x, y) from time step t to $t+1$ is defined in such a way that two corresponding pixels $I(x, y, t)$ and $I(x + u_{x,y,t}, j + v_{x,y,t}, t + 1)$ have similar brightness (pixel) values. Mathematically, it means

$$I(x, y, t) = I(x + u_{x,y,t}, y + v_{x,y,t}, t + 1). \tag{2.1}$$

It is often known as the *brightness constancy* assumption for optical flow estimation. For simplicity, we omit the time step subscript for optical flow and simply use $u_{x,y}$ and $v_{x,y}$. We also define two extra symbols I_t and I_{t+1} to denote the video frame at the time step t and $t + 1$, respectively.

Fig. 2.1 shows an illustration of estimated optical flow between two consecutive images. Throughout this thesis, we will use colorized optical flow like the one shown

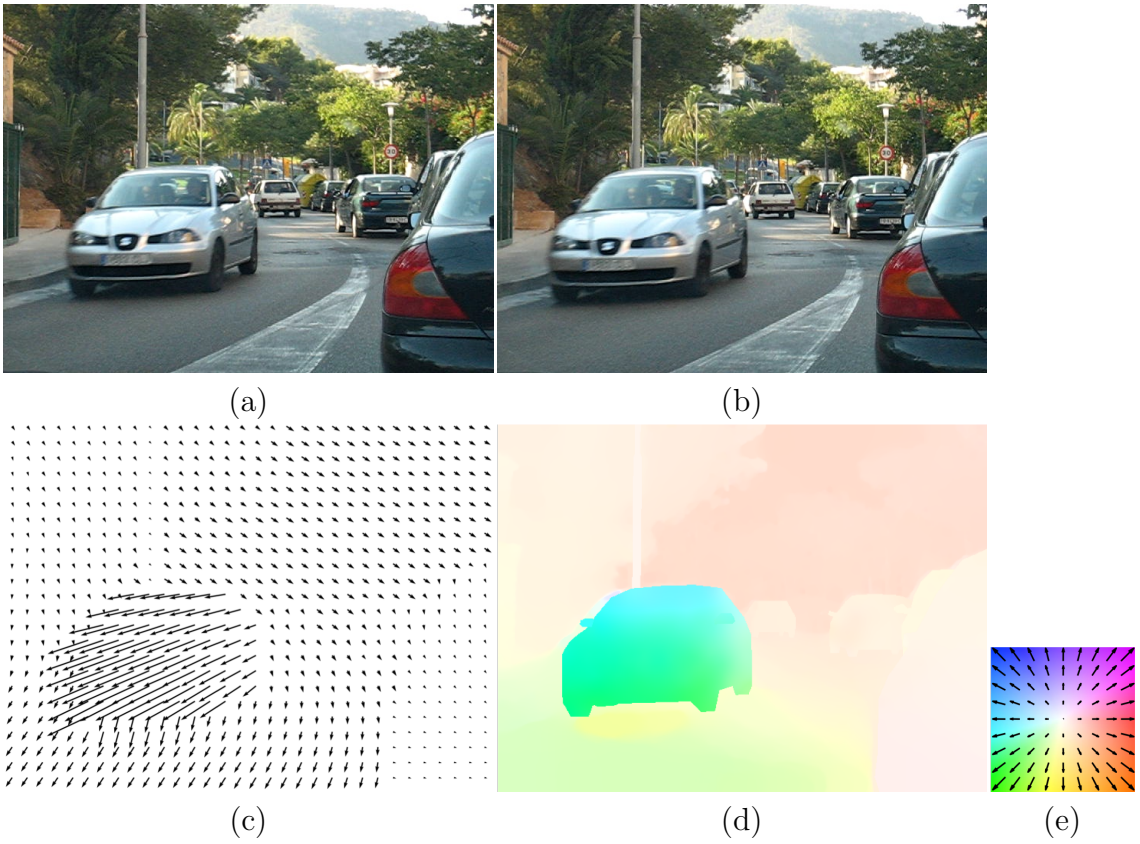


Figure 2.1: Optical flow estimation. (a)(b) two consecutive video frames from [76], (c) an illustration of optical flow, where the motion vector of each pixel is represented by an arrow, and (d) a visualization of the colorized optical flow using the color key shown in (e), where the hue indicates motion direction and saturation corresponds to the motion magnitude.

in Fig. 2.1(d) to visualize estimated optical flow, where the hue indicates motion direction and saturation corresponds to the motion magnitude..

2.1.1 Classical Approaches

As defined in Eq.(2.1), optical flow estimation is an ill-posed problem, where there are two unknown variables for each pixel but only one constraint is available, known as the *aperture problem*. To solve this problem, the additional *spatial smoothness* constraint is usually adopted, where it is assumed that two adjacent pixels have *similar* optical flow values. To jointly optimize the brightness constancy and spatial

smoothness objectives, an energy function is usually defined as follows [129]:

$$E(I_t, I_{t+1}, \mathbf{u}, \mathbf{v}) = \sum_{x,y} E_D(I_t, I_{t+1}, \mathbf{u}, \mathbf{v}) + \lambda \sum_{x,y} E_S(\mathbf{u}, \mathbf{v}), \tag{2.2}$$

where $\mathbf{u}$ and $\mathbf{v}$ define the optical flow field for all pixels in I_t, corresponding to the horizontal and vertical components, respectively. E_D captures the brightness consistency and E_S enforces the spatial smoothness constraint. λ is a hyper-parameter balancing such two terms. They are typically defined as

$$E_D(I_t, I_{t+1}, \mathbf{u}, \mathbf{v}) = \rho\left(I_t(x, y) - I_{t+1}(x + u_{x,y}, y + v_{x,y})\right), \tag{2.3}$$

$$E_S(\mathbf{u}, \mathbf{v}) = \rho(u_{x,y} - u_{x+1,y}) + \rho(u_{x,y} - u_{x,y+1}) + \tag{2.4}$$

$$\rho(v_{x,y} - v_{x+1,y}) + \rho(v_{x,y} - v_{x,y+1}), \tag{2.5}$$

where $\rho(\cdot)$ is a penalty function. Such an energy function is related to the standard pairwise Markov Random Field (MRF) defined over a 4-pixel-neighborhood.

Such an energy minimization-based approach is first studied in the seminal paper by Horn and Schunck [41], where the penalty function is simply defined as $\rho(z) = z^2$. Despite its simplicity, such an "old" approach is found to work surprisingly well in a later test-of-time-award-winning paper [129] with appropriate optimization techniques. Other forms of penalty functions are also used, including the most robust convex Charbonnier penalty $\rho(z) = \sqrt{z^2 + \epsilon^2}$ [15] and the non-convex robust Lorentzian penalty $\rho(z) = \log(1 + \frac{z^2}{2\sigma^2})$ [11].

The energy function Eq.(2.2) is defined over all pixels in I_t, which is often referred to as a *global* approach. There also exists *local* approaches, where the energy function is defined for each pixel over a small neighborhood, such as the Lucas-Kanade approach [83]. If we do a Taylor expansion around the position (x, y, t) for $I(x + u_{x,y}, y + v_{x,y}, t + 1)$, we have:

$$I(x+u_{x,y}, y+v_{x,y}, t+1) = I(x,y,t) + \frac{\partial I}{\partial x}u_{x,y} + \frac{\partial I}{\partial y}v_{x,y} + \frac{\partial I}{\partial t}. \tag{2.6}$$

Consider the brightness constancy assumption in Eq.(2.1), the optical flow $u_{x,y}$ and $v_{x,y}$ should satisfy the following equation.

$$\frac{\partial I}{\partial x}u_{x,y} + \frac{\partial I}{\partial y}v_{x,y} + \frac{\partial I}{\partial t} = 0. \tag{2.7}$$

Again, there are no sufficient constraints to solve two variables from a single equation. Similar to Horn and Schunck [41], Lucas and Kanade [83] assume the motion is constant within a local neighborhood $[x-k, y-k] \times [x+k, y+k]$ in I_t around the position (x, y), whre k is the radius of the neighborhood. In specific, we have

$$\begin{bmatrix} \frac{\partial I}{\partial x}|_{(x-k,y-k)} & \frac{\partial I}{\partial x}|_{(x-k,y-k)} \\ \frac{\partial I}{\partial x}|_{(x-k+1,y-k)} & \frac{\partial I}{\partial x}|_{(x-k+1,y-k)} \\ \vdots & \vdots \\ \frac{\partial I}{\partial x}|_{(x+k,y+k)} & \frac{\partial I}{\partial x}|_{(x+k,y+k)} \end{bmatrix} \begin{bmatrix} u_{x,y} \\ v_{x,y} \end{bmatrix} = - \begin{bmatrix} \frac{\partial I}{\partial t}|_{(x-k,y-k)} \\ \frac{\partial I}{\partial t}|_{(x-k+1,y-k)} \\ \vdots \\ \frac{\partial I}{\partial t}|_{(x+k,y+k)} \end{bmatrix}. \tag{2.8}$$

Such a linear system can be solved if its system matrix (the first matrix in the left side) is invertible, which is not always true however. For a particular spatial position in a homogeneous region, the spatial image gradients (*i.e.*, $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$) are small. As a result, the smaller eigenvalue of the system matrix may be close to 0 and there is not a reliable estimation of optical flow.

Such a global approach, Horn and Schunck [41], and a local one, Lucas and Kanade [83], have their own advantages and limitations. On the one hand, the global approach [41] provides *dense* optical flow values for every single pixel. But it does not offer confidence value for the quality of the estimated optical flow. It is also more subject to noise [15]. On the other hand, the local approach only provides optical flow values at *sparse* locations. An interpolation is usually required to obtain the dense

optical flow field. In [15], such global and local spatial constraints are integrated in a single energy function to be robust to noise as [83] and yet provide dense optical flow field similar to [41], combining the best of two approaches.

As a fundamental problem in visual perception, optical flow estimation has attracted a lot of research attention. Variants of [41] and [83] have been proposed to address their limitations, especially for the former one. Optical flow estimation essentially is about finding pixel correspondences between two images, which is another long-studied classical problem in computer vision, particularly using feature descriptors extracted at sparse locations, such as SIFT [82] and its successive variants. One of the effort is to provide more constraints in the energy function for optical flow estimation by adding high-quality descriptor-based matches between two images, such as LDOF (large-displacement optical flow) [14]. Feature descriptors are designed in essence to be robust to factors such as lighting change, rotation, etc. These high-quality sparse matches are able to more accurately capture large-displacement motion and serve as seeds to generate dense optical flow fields by minimizing the energy function.

In a similar effort, SIFT flow [77] is proposed to provide dense correspondences between two images, which are not necessarily two consecutive video frames, based on the matching of pixel-wise SIFT descriptors. SIFT flow is not particularly designed for optical flow estimation, although it is based on a similar energy function for optical flow estimation. While SIFT flow is able to capture *semantic* correspondences between two general images, such as heads of two different dogs or wheels of a bicycle and a motorcycle, it does not perform well for optical flow estimation in particular.

The optimization of the energy function is usually performed in an iterative manner, where it usually converges to a local minimum. Initialization plays a critical role in the optimization. Instead of adding sparse matches as a term in the energy function for optical flow estimation, an *interpolation* approach is proposed in [109] to

propagate sparse correspondences to other positions that have no correspondences, where an edge-aware distance metric (*i.e.*, geodesic distance) are used to fit a local affine model at each pixel based on nearby sparse matches.

Correspondences found by generic feature descriptors like SIFT [82] may not be correct matches for optical flow. After all, matches found based the similarities of SIFT descriptors over the entire image are two positions that have similar local brightness contrast statistics. To overcome this issue, dense correspondence fields are constructed in [4], where a hierarchical correspondence search strategy is utilized. Forward-backward correspondence consistency verification is further used to filter outlier matching, leading to more accurate matching. The estimated dense correspondences are then used as an initialization for minimizing an energy function to get the final optical flow estimation as in [109].

Matching accuracy of hand-crafted feature descriptors, such as SIFT [82], usually suffers from factors like lighting change, motion blur, rotation, etc. Inspired by the success of deep neural networks for other computer vision problems, a 6-layer convolutional neural network is proposed in [110] to find accurate correspondences between two consecutive video frames. In contrast to a typical neural network, where parameters are learned from the training date, filters in the matching network [110] are patches from the first image. It essentially captures the *correlation* between patches from two video frames to measure their similarity score, which we shall see in next section is widely used in state-of-the-art neural network-based optical flow estimation approaches.

All aforementioned approaches rely on manually designed data terms E_D and spatial terms E_S in the energy function. Sun *et al.* [128] propose to learn their statistics directly from the data with a probabilistic model. In such a data-driven model, high-order constancy statistics may be mined from data, not just the brightness constancy. Additionally, the spatial smoothness term is also learned from data based

on the image intensity structure (spatial gradients). This supervised learning model still heavily relies on the form of energy function proposed by Horn and Schunck [41] in early 1980s. We shall see in next section that state-of-the-art supervised learning approaches using deep convolutional neural networks (CNNs) learn to estimate optical flow field from the data by purely minimizing the error between model predictions and the ground-truth annotations, where the brightness constancy and spatial smoothness are *implicitly* enforced.

2.1.2 Neural Approaches

Unlike the classical energy minimization-based approaches, including the learning-based on [128], *supervised* deep learning models aim to directly minimize a loss function

$$l(\theta) = \sum_{x,y} \rho\left(\mathbf{w}(x,y), \hat{\mathbf{w}}(x,y;\theta)\right), \tag{2.9}$$

where $\mathbf{w} = (\mathbf{u}, \mathbf{v})$ represents the ground-truth annotations of the optical flow field and $\hat{\mathbf{w}}$ is the estimated optical flow as output of a deep neural network, which is parameterized by θ. The goal of training a deep neural network is to obtain optimal parameters θ^* such that the empirical loss value over the entire training set is minimized:

$$\theta^* = \min_{\theta} \frac{1}{N} \sum_{i=0}^{N-1} l_i(\theta). \tag{2.10}$$

Here l_i is the loss value over the i-th training sample and N is the total number of all training samples. The optimization is usually performed in an end-to-end manner using a variant of the stochastic gradient descent optimizer, such as momentum-SGD, Adam [66], etc.

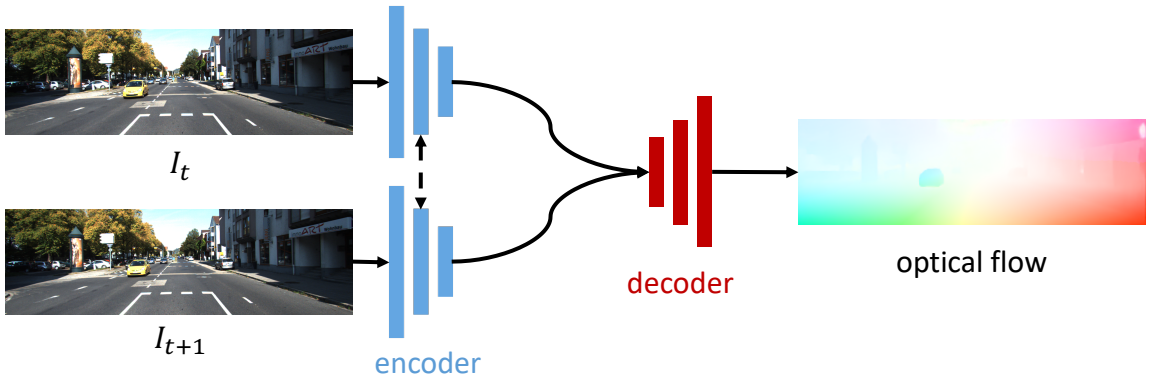


Figure 2.2: Illustration of the encoder-decoder network structure widely used for optical flow estimation. The feature representation of two input images are obtained using the same encoder, which are then fed into a decoder to compute the optical flow.

Training data-hungry deep neural networks for optical flow requires a huge amount of data with ground-truth annotations, which are almost infeasible to collect for real-world data. As a result, researchers usually use synthetic data for training, where optical flow annotations come for free when the scene is rendered. To date, the most often used synthetic data are FlyingChairs [26] and FlyingThings3D [90], which contain 22,872 and 39,280 images for training, respectively. It is worth noting that FlyingThings3D provides other form of annotations to train models for stereo disparity and scene flow estimations.

FlowNet [26] is the first attempt to train deep networks for optical flow estimation. There are two variants: FlowNetS and FlowNetC, both of which have an encoder-decoder structure, as shown in Fig. 2.2. The same encoder is used to computer feature representations for the input images, which are then fed into the decoder to compute optical flow. FlowNetS simply *concatenates* the feature representations of the two images and let the decoder to learn how to compute optical flow. In contrast, FlowNetC borrows the idea of computing *matches* between feature presentations (or images), which is widely used in classical approaches, as we have seen in the previous

section. Denote the feature vectors of two input images at locations i and j as $\mathbf{h}_i$ and $\mathbf{h}_j$, respectively. FlowNetC computes the matching cost as follows.

$$c(\mathbf{h}_i, \mathbf{h}_j) = \frac{1}{D}\mathbf{h}_i^T \mathbf{h}_j, \tag{2.11}$$

where D is the dimension of a feature vector. Usually for a fixed position i in the first image's feature map, a set of candidates j in the second image within a local neighborhood are considered to compute the matching cost, forming a so-called *cost volume*, which is now widely used in state-of-the-art neural approaches.

As a first attempt, neither FlowNetS nor FlowNetC achieves satisfactory accuracy for optical flow estimation. To further improve the accuracy, FlowNet2 [48] concatenates FlowNetS and FlowNetC variants in a cascade manner, where the optical flow estimation is progressively refined. For the first time, a deep neural network approach reports better or on-par results with classicial well-engineered approaches. For qualitative results, FlowNet2 is able to capture the motion of thin structures, producing sharp motion boundaries, thanks to the refinement network.

Although FlowNet and FlowNet2 achieve reasonably good accuracy for optical flow estimation, they have 32M and 162M parameters, respectively, costing a lot of GPU memory for inference. A much more compact SpyNet [104] is proposed. It borrows the idea of computing optical flow in a coarse-to-fine manner by using a *pyramid* structure that is extensively used in classical approaches. In specific, an image pyramid is constructed through downsampling, where in the coarsest level, an optical flow is estimated by feeding concatenated images as input to a network. For a finer level, such an optical flow is upsampled and a *residual* optical flow is estimated to obtain more accurate estimation. SpyNet achieves comparable accuracy to classical approaches as well as the FlowNet variants, yet has merely 1.2M parameters, yielding a compact model.

PWC-Net [131] extends the pyramid structure used in SpyNet. It uses multi-scale feature representations that naturally come from the encoder because of downsampling operations. Therefore, instead of finding correspondences between two images, PWC-Net does so on two feature maps. Moreover, optical flow estimation for a coarse pyramid level is used to *warp* feature representations of the second image on a finer level when computing the cost volume, so that it only needs to search for correspondences in relatively small neighborhood, resulting in small computation burden. PWC-Net not only performs better than FlowNet2 but also is more compact, containing only 8.75M parameters. A similar architecture is concurrently studied in LiteFlowNet [45].

Denote the dimension of a feature vector x_i as $C \times H \times W$[1], where H and W are the spatial height and width, respectively. Denote the range of the local neighborhood of search for correspondences in the horizontal and vertical directions as U and V, respectively. The dimension of cost volume used in PWC-Net is $(1 \times U \times V) \times H \times W$, where the matching cost over the local neighborhood is *flattened*, yielding a 3D feature map so that 2D convolutions are applied on top of it to estimate optical flow. However, the flatten operation destroys the topology of the local neighborhood. VCN [158] is proposed to avoid the flatten operation. Furthermore, it computes the matching cost at very single feature channel, leading to a cost volume with a dimension of $C \times U \times V \times H \times W$. As a results, 4D convolutions are needed to process such a 5D feature map, which lacks native support in most of deep learning frameworks. Instead, a 4D convolution is converted to two 3D convolutions. VCN achieves better accuracy for optical flow estimation than PWC-Net mainly because of such 5D cost volumes.

[1]We omit the batch dimension here for simplicity.

Both PWC-Net [131] and VCN [158] have to maintain a cost volume that computes matching cost in a relatively small neighborhood (*e.g.*, in a 9×9 region) in order to get an efficient model. Searching in a local neighborhood, however, prohibits the model from capturing very large displacement. To address this issue, a cost volume between every single pair of positions between two feature maps are used in RAFT [135]. A recurrent module is then employed to estimate optical flow residuals, where only a small portion of the cost volume around the current estimated correspondences is used. RAFT significantly reduces the error of optical flow estimation on standard benchmarks while still maintaining reasonable efficiency.

2.1.3 Discussions

In previous sections, we briefly review both classical and recent neural approaches for optical flow estimation. Now let's take a look at the progress of the entire community. Do we move toward the right direction? What do neural approaches bring us when the entire community shifted from classical energy minimization-based approaches to deep neural networks with supervision purely from the data?

In Fig. 2.3, we show the end-point-error (EPE) of different methods versus their published years on the MPI Sintel test set (final pass) [17]. We can clearly see that the EPE keeps going down by 65.8% from the year of 2010 (classic++ [129]) to 2020 (RAFT [135][2]). Initially, when the community shifted to deep neural network-based approaches in the year of 2015, neural approaches do not perform as well as classical well-engineered ones. In 2018, PWC-Net [131], for the first time, shows that neural approaches can achieve lower EPE than classical approaches. Both VCN and RAFT keep pushing the EPE downward.

[2]By using more training data and more than two images as input, the EPE can go down to 2.86, a 71.3% reduction compared with classic++ [129].

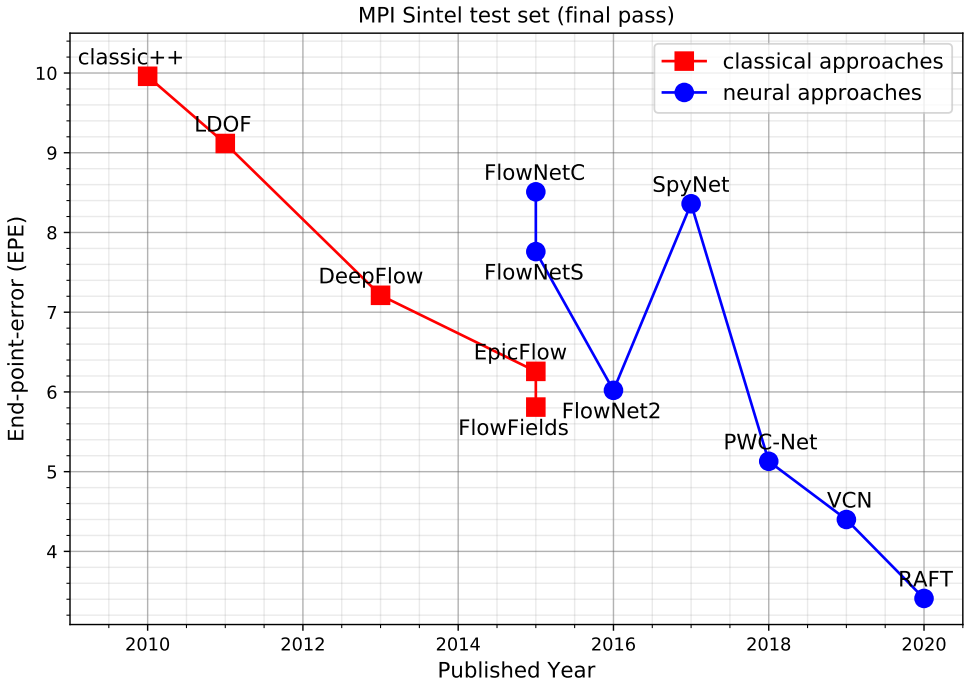


Figure 2.3: End-point-error (EPE) of different optical flow estimation methods versus their published years on the MPI Sintel test set (final pass) [17].

In addition to the significant reduction of error metrics, neural approaches run substantially faster than classical approaches. Because of the nature of neural networks, which are essentially about matrix (or tensor) multiplication and addition, neural approaches can be easily parallelized, utilizing the massive computation units in modern GPUs. It takes about 800s for classic++ [129] to compute optical flow between two images while RAFT [135] only needs 0.3s on a mid-end GTX 1080ti GPU[3], which is nearly 2,700 times faster in terms of number of frames to process per second (FPS).

As a fundamental task in computer vision, optical flow estimation is usually part of an entire pipeline for other downstream tasks, such as action recognition, video

[3]It can can be further accelerated using a higher-end GPU and other techniques such as mixed-precision, quantization, pruning, etc.

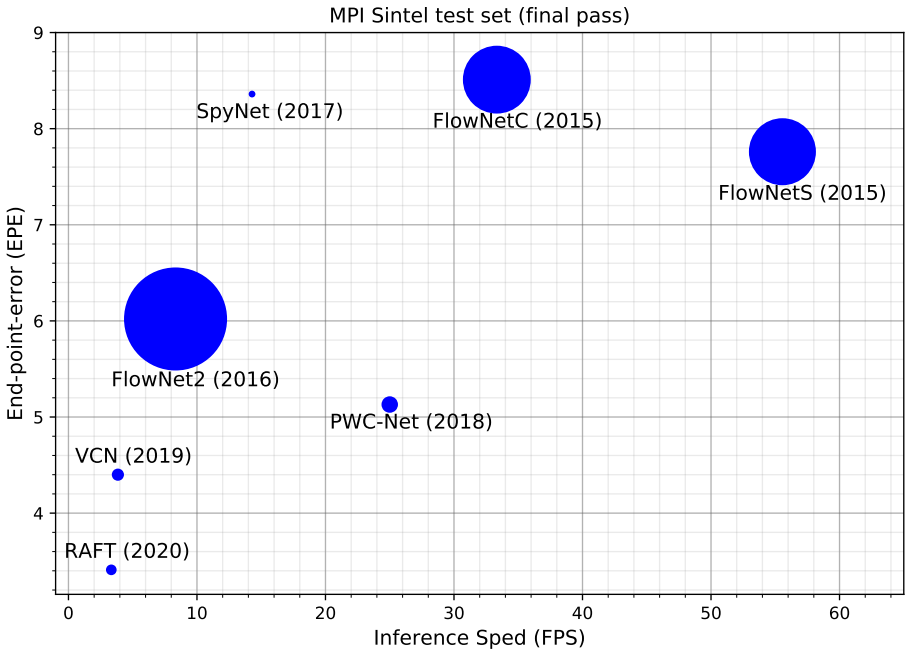


Figure 2.4: End-point-error (EPE) of different optical flow estimation methods versus their inference speed in terms of number of frames to process per second (FPS). At the same time, the size of the circle indicates the number of parameters for each method (larger means more parameters).

editing, 3D scene understanding, etc. Therefore, a lightweight yet well-perform approach is required. In Fig. 2.4, we show the EPE versus the inference speed in terms of number of frames to process per second (FPS) for each method, as well as their parameters. In the early stage of neural approaches, the improvement of accuracy mainly comes from more parameters, for example, from FlowNetS to FlowNet2 that has several cascades of concatenated FlowNetC and FlowNetS for refinement. PWC-Net shows that by using techniques, including pyramid and warping, that are widely used in classical approaches, a deep neural network does not have to have a huge number of parameters in order to perform well. Maintaining a balance of efficiency and effectiveness of a deep neural network for optical flow estimation will be the main trend for optical flow estimation.

Does the success of neural approaches render classical ones useless nowadays? Do we still need to consider the brightness constancy and spatial smootheness constraints used in classical approaches? For the first question, injecting prior or inductive bias, like what PWC-Net does (for example, pyramid and warping operations), often leads to an optical flow estimation model that not only achieves lower error rate than simply treating neural networks as a blackbox but also requires fewer parameters to more effectively utilize the existing training data.

For the second question, *supervised* optical flow estimation approaches purely learn from training data, where the brightness constancy and spatial smoothness constraints are implicitly enforced in the annotations of training data. *Unsupervised* optical flow estimation approaches [107, 91, 53] still relies on brightness constancy and spatial smoothness assumptions to compute the loss as supervision to update neural networks' parameters.

2.2 From Optical Flow to 3D Scene Understanding

Optical flow is a projection of 3D motion of the scene onto the image plane, which is a joint function of scene depth, camera motion, moving objects, and the camera's intrinsic parameters. For illustration, let's take a look at the perspective projection model of a camera, as shown in Fig. 2.5. A coordinate system is centered at the origin O, where the optical center of a camera is located at. Another local coordinates system is attached to the image plane, which is f away from the optical center of the camera. f is an intrinsic parameter of the camera, known as *focal length*. Suppose there is a point $P = (X_P, Y_P, Z_P)$ in the 3D world, which is projected onto the image plane at point $q = (x_q, y_q)$. According to the geometric constraints, we have

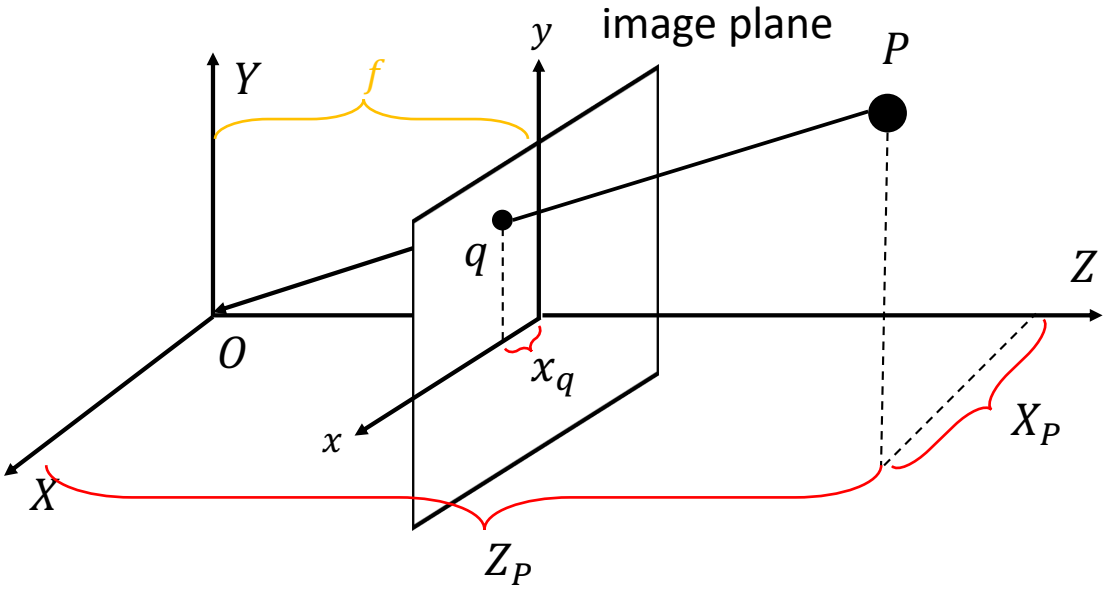


Figure 2.5: An illustration of the perspective projection model. The optical center of a camera is located at the origin O. There are two coordinate systems in the 3D world and the 2D image plane. The image plane is f away from the optical center of camera, which is an intrinsic parameter of the camera. A point P in the 3D world is projected onto the image plane at point q.

$$x_q = f\frac{X_P}{Z_P}, \quad y_q = f\frac{Y_P}{Z_P}. \tag{2.12}$$

As explained in [16], if we take derivatives with respect to the time step t, we will have

$$x'_q = \frac{X'_P}{Z_P} - \frac{X_P Z'_P}{Z_P^2}, \tag{2.13}$$

$$y'_q = \frac{Y'_P}{Z_P} - \frac{Y_P Z'_P}{Z_P^2}, \tag{2.14}$$

where we omit the constant f for simplicity. (x'_q, y'_q) capture the pixel q's motion in the image plane, which is optical flow as we have seen in the previous section. So we have

$$u_q = x'_q, \quad v_q = y'_q. \tag{2.15}$$

(X'_P, Y'_P, Z'_P) are the motion of the point P in the 3D world. Assume *the scene is relatively static to the camera* and the motion is purely caused by the relative movement of the camera, which consists of two factors: a translation and a rotation. Denote P's position by $\mathbf{r}_P = (X_P, Y_P, Z_P)^T$, its motion by $\mathbf{V}_P = (X'_P, Y'_P, Z'_P)^T$, the camera motion's translational component by $\mathbf{T}$, and its rotational component (*i.e.*, the angular velocity) by ω. We have

$$\mathbf{V}_P = -\mathbf{T} - \omega \times \mathbf{r}_P, \tag{2.16}$$

where $\times$ is the cross product between two vectors. If we define the components of $\mathbf{T}$ and ω along the X, Y, Z axes as

$$\mathbf{T} = (U, V, W)^T, \quad \omega = (A, B, C)^T, \tag{2.17}$$

and substitute them into Eq.(2.16), we have

$$X'_P = -U - BZ_P + CY_P, \tag{2.18}$$

$$Y'_P = -V - CX_P + AZ_P, \tag{2.19}$$

$$Z'_P = -W - AY_P + BX_P. \tag{2.20}$$

According to Eq.(2.14), we have

$$u_q = \left(-\frac{U}{Z_P} - B + Cy_q\right) - x_q\left(-\frac{W}{Z_P} - Ay_q + Bx_q\right), \tag{2.21}$$

$$v_q = \left(-\frac{V}{Z_P} - Cx_q + A\right) - y_q\left(-\frac{W}{Z_P} - Ay_q + Bx_q\right). \tag{2.22}$$

These two equations can be re-written as

$$u_q = u_q^{tr} + u_q^{ro}, \quad v_q = v_q^{tr} + v_q^{ro}, \tag{2.23}$$

where (u_q^{tr}, v_q^{tr}) denotes the translational component of the optical flow and (u_q^{ro}, v_q^{ro}) the rotational component. In specific,

$$u_q^{tr} = \frac{-U + x_q W}{Z_P}, \qquad v_q^{tr} = \frac{-V + y_q W}{Z_P}, \tag{2.24}$$

$$u_q^{ro} = A x_q y_q - B(x_q^2 + 1) + C y_q, \qquad v_q^{ro} = A(y_q^2 + 1) - B x_q y_q - C x_q. \tag{2.25}$$

We can clearly see that optical flow, a pixel's motion in the image plane, is essentially about the projection of the relative motion of the scene and the camera, which is jointly defined by the scene structure (depth Z_P) and the camera's motion $\mathbf{V} = (U, V, W)^T$ and $\omega = (A, B, C)^T$. It is assumed, however, in the derivations so far that the scene is static. For an independently moving object, such a relationship does not hold. Therefore, to accurately model the scene's motion, we need to know optical flow, scene structure, and camera motion, and where the static and moving parts of the scene are (known as motion segmentation). For the rest of this section, we will briefly review recent research on 3D scene understanding using optical flow.

Motion Segmentation. Inspired this derivation based on the perspective projection [16], a motion segmentation algorithm is proposed in [9], where pre-computed optical flow is used as input to estimate the translational and rotational components of the camera motion. The angle and magnitude fields of optical flow is then integrated into a probabilistic model to segment moving objects by assigning each pixel to its most likely motion model in a Bayesian fashion. Optical flow is often used as input for recent neural network-based approaches [138, 139], where motion segmentation is modeled a binary segmentation problem to be learned in an end-to-end manner using annotations of moving objects. In recent work [20], an instance segmentation

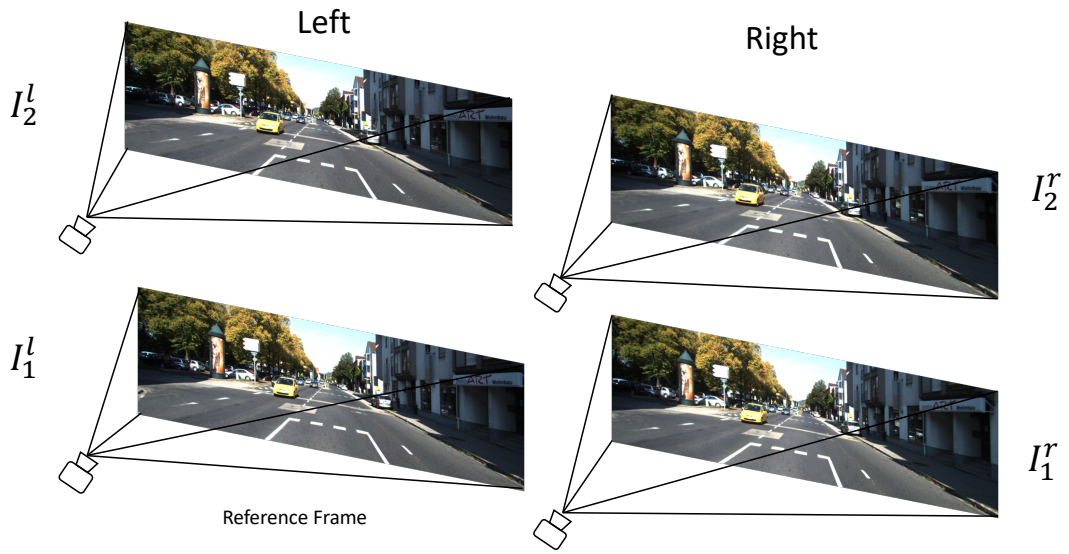


Figure 2.6: A typical configuration for scene flow estimation. Four images are available as input that are taken by two stereo cameras at two consecutive time steps. For math symbols, a subscript denotes the time step and a superscript indicates whether the image is taken by the left or the right camera. The goal is to infer the 3D motion of every pixel in the reference image (bottom left).

approach is proposed, which has two branches, corresponding to RGB images and optical flow as input, respectively. Such two branches are progressively trained toward segmenting class-agnostic moving instances.

3D Motion Estimation. Estimating 3D motion of a scene is critical in some scenarios. For example, for an autonomous driving car or a navigation robot in an apartment, it is essential to know how far away other obstacles and objects are away and how fast they move. Fig. 2.6 demonstrates a typical configuration for scene flow estimation using two stereo cameras, where four images are available as input that are taken at two consecutive time steps. For scene flow estimation, optical flow is an essential input. In [92], an energy function based on a discrete-continuous Conditional Random Field (CRF) is proposed. Each object in the scene is represented by its optical flow, stereo disparity, and rigid motion in 3D. By minimizing the energy function, the estimated scene flow should be consistent with such evidence, where smoothness constraints are also considered in the energy function. In recent work [87],

it extends such an energy function by jointly considering optical flow, stereo disparity, and instance segmentation, taking advantages of great progress of all these three fields thanks to deep learning models. Moreover, a Gaussian-Newton optimization approach is adopted, which can be easily parallelized on GPUs, leading to not only more accurate scene flow estimation, but also significantly faster processing speed.

Joint Optical Flow Estimation and Scene Segmentation. Scene understanding tasks are usually closely related. Solving multiple tasks jointly often leads to more accurate scene understanding than solving each of separately. Specifically, estimating motion of a dynamic scene involves not only merely brightness pattern matching, but also understanding of the scene including object co-occurrences, intuitive physics, etc. For instance, we know a tennis ball may incur a large displacement after it is hit to a racket. Furthermore, there could be other tennis balls in the background. Understanding which tennis ball is the correct correspondent could depend upon a knowledge of the physics of tennis.

In [120], semantic image segmentation, where each pixel is assigned to one of semantic categories, such as tree, road, etc, and optical flow estimation are studied at the same time. Based on the segmentation of the scene, different motion models are constructed for different scenes. For example, motion of the road area is modeled with homographies and independently moving objects' motion is considered as an affine model plus deviations. Such different motion models are then integrated into a layer model, where each layer corresponds to a region in the image segmentation, to compose the scene's optical flow estimation. Such semantic optical flow estimation leads to more accurate motion estimation than relying on motion of brightness patterns. In [153], motion segmentation (termed as rigidity estimation) and optical flow estimation are jointly investigated. In static scenes, camera motion and scene structure (depth) that are estimated using multiple frames using a structure-from-motion

(SfM) algorithm are used to compute optical flow. While in moving regions, results of using an existing optical flow estimation are used.

Unsupervised Scene Understanding. Since there are strong constraints between different scene understanding tasks, can we train a model to solve them without using any manual annotations, purely using their interactions as supervision signal? A unsupervised learning approach using deep convolutional neural networks is proposed in [161] to estimate depth, optical flow, and camera pose for unlabeled monocular videos. The supervision comes from the photometric reconstruction error (based on the brightness constancy assumption) by projecting a video frame to its neighboring ones. In [105], four tasks, including depth prediction, optical flow estimation, camera motion estimation, and motion segmentation are solved at the same time. They are integrated into a common framework, where the segmentation of static scene and independently moving objects are simultaneously reasoned. The optimization is performed in an iterative manner, similar to the expectation-maximization (EM) algorithm.

CHAPTER 3

SELF-SUPERVISED RELATIVE DEPTH LEARNING FOR URBAN SCENE UNDERSTANDING

3.1 Overview

How does a newborn agent learn about the world? When an animal (or robot) moves, its visual system is exposed to a shower of information. Usually, the speed with which something moves in the image is inversely proportional to its depth.[1] As an agent continues to experience visual stimuli under its own motion, it is natural for it to form associations between the appearance of objects and their relative motion in the image. For example, an agent may learn that objects that look like mountains typically don't move in the image (or change appearance much) as the agent moves. Objects like nearby buildings and bushes, however, appear to move rapidly in the image as the agent changes position relative to them. This continuous pairing of images with motion acts as a kind of automatic supervision that could eventually allow an agent both to understand the depth of objects and to group pixels into objects by this predicted depth. Thus, by moving through the world, an agent may learn to predict properties (such as depth) of *static* scenes.

A flurry of recent work has shown that *proxy tasks* (also known as *pretext* or *surrogate tasks*) such as colorization [71, 165], jigsaw puzzles [98], and others [147, 1, 100, 94, 101, 29, 99, 72], can induce features in a neural network that provide strong pre-training for subsequent tasks. In this chapter, we introduce a new proxy task: estimation of relative depth from a single image. We show that a network that has been pre-trained, without human supervision, to predict relative scene depth provides a powerful starting point from which to fine-tune models for a variety of urban

(a) (b) (c) (d) (e)

Figure 3.1: Sample frames from collected videos and their corresponding relative depth maps, where brightness encodes relative depth (the brighter the farther). From top to bottom: input image, relative depth image computed using Eq.(3.3), and predicted (relative) depth maps using our trained VGG16 FCN8s [122, 121]. There is often a black blob around the center of the image, a singularity in depth estimation caused by the focus of expansion. (a)(b)(c): images from the CityDriving dataset, (d): images from the KITTI dataset, and (e): images from the CityScapes dataset.

scene understanding tasks. Not only does this automatically supervised starting point outperform all other proxy task pre-training methods. For monocular depth understanding, it even performs better than the heavily supervised ImageNet pre-training, yielding results that are comparable with state-of-the-art methods.

To estimate relative scene depths without human supervision, we use a recent motion segmentation technique [9] to estimate relative depth from geometric constraints between a scene's motion field and the camera motion. We apply it to simple, publicly available YouTube videos taken from moving cars. Since this technique estimates depth *up to an unknown scale factor*, we compute *relative depth* of the scene during the pre-training phase, where each pixel's value is in the range of $[0, 1]$ denoting its depth percentile over the entire image.[1]

Unlike work that analyzes video paired with additional information about direction of motion [56], our agent learns from "raw egomotion" video recorded from cars moving through the world. Unlike methods that require videos of moving ob-

[1]Later, we will fine-tune networks to produce absolute depths.

jects [100], we neither depend on, nor are disrupted by, moving objects in the video. Once we have relative depth estimates for these video images, we train a deep network to predict the relative depth of each pixel from a *single image*, *i.e.*, to predict the relative depth *without* the benefit of motion. One might expect such a network to learn that an image patch that looks like a house and spans 20 pixels of an image (about 100 meters away) is significantly farther away than a pedestrian that spans 100 image pixels (perhaps 10 meters away). Figure 3.1 illustrates this prediction task and shows sample results obtained using a standard convolutional neural network (CNN) in this setting. For example, in the leftmost image of Fig. 3.1, an otherwise unremarkable traffic-light pole is clearly highlighted by its relative depth profile, which stands out from the background. Our hypothesis is that to excel at relative depth estimation, the CNN will benefit by learning to recognize such structures.

The goal of our work is to show that pre-training a network to do relative depth prediction is a powerful proxy task for learning visual representations. In particular, we show that a network pre-trained for relative depth prediction (from automatically generated training data) improves training for downstream tasks including semantic segmentation, joint semantic reasoning of road segmentation and car detection, and monocular (absolute) depth estimation. We obtain significant performance gains on urban scene understanding benchmarks such as KITTI [32, 31] and CityScapes [19], compared to training a segmentation model from scratch. Compared to nine other proxy tasks for pre-training, our proxy task consistently provides the highest gains when used for pre-training. In fact, our performance on semantic segmentation and joint semantic reasoning tasks comes close to that of equivalent architectures pre-trained with ImageNet [21], a massive labeled dataset. Finally, for the monocular (absolute) depth estimation, *our pre-trained model achieves better performance than an ImageNet pre-trained model*, using both VGG16 [122] and ResNet50 [37] architectures.

As a final application, we show how our proxy task can be used for *domain adaptation*. One might assume that the more similar the domain of unlabeled visual data used for the proxy task (here, three urban scene understanding tasks) is to the domain in which the eventual semantic task is defined (here, semantic segmentation), the better the representation learned by pre-training. This observation allows us to go beyond simple pre-training, and effectively provide a domain adaptation mechanism. By adapting (fine-tuning) a relative depth prediction model to targets obtained from unlabeled data in a novel domain (say, driving in a new city) we can improve the underlying representation, priming it for better performance on a semantic task (*e.g.*, segmentation) trained with a small labeled dataset from this new domain. In experiments, we show that pre-training on unlabeled videos from a target city, absent any labeled data from that city, consistently improves all urban scene understanding tasks.

In total, our work advances two pathways for integrating unlabeled data with visual learning.

- We propose a novel proxy task for self-supervised learning of visual representations; it is based on learning to predict relative depth, inferred from unlabeled videos. This unsupervised pre-training leads to better results over all other proxy tasks on the semantic segmentation task, and even outperforms supervised ImageNet pre-training for absolute depth estimation.

- We show that our task can be used to drive domain adaptation. Experiments demonstrate its utility in scene understanding tasks for street scenes in a novel city. Our adapted model achieves results that are competitive with state-of-the-art methods (including those that use *large supervised pre-training*) on the KITTI depth estimation benchmark.

Such methods of extracting knowledge from unlabeled data are likely to be increasingly important as computer vision scales to real-world applications; here massive dataset size can outpace herculean annotation efforts.

3.2 Related Work

Self-supervised learning. The idea of formulating supervised prediction tasks on unlabeled data has been leveraged for both images and videos. The idea, often called self-supervision, is most typically realized by removing part of the input and then training a network to predict it. This can take the form of deleting a spatial region and trying to inpaint it [101], draining an image of color and trying to colorize it [70, 165, 71], or removing the final frame in a sequence and trying to hallucinate it [106, 125, 89, 144, 157, 81]. Generative Adversarial Networks, used for inpainting and several future frame prediction methods, can also be used to generate realistic-looking samples from scratch. This has found secondary utility for unsupervised representation learning [103, 124, 25]. Another strategy is to extract patches and try to predict their spatial or temporal relationship. In images, this has been done for pairs of patches [22] or for 3-by-3 jigsaw puzzles [98]. In videos, it can be done by predicting the temporal ordering of frames [94, 72]. The correlation of frames in video is also a rich source of self-supervised learning signals. The assumption that close-by frames are more similar than far apart frames can be used to train embeddings on pairs [95, 51, 57] or triplets [147] of frames. A related idea that the representation of interesting objects should change slowly through time dates back to Slow Feature Analysis [152].

The works most closely related to ours may be [56, 1, 100], which aim to learn useful visual representations from unlabeled videos as well. Jayaraman & Grauman [56] learn a representation equivariant to ego-motion transformations, using ideas from metric learning. Agrawal *et al.* [1] concurrently developed a similar method that

uses the ego-motion directly as the prediction target as opposed to as input to an equivariant transformation. Both of these works assume knowledge of the agent's own motor actions, which limits their evaluation in sample size due to lack of publicly available data. In our work, the ego-motion is inferred through optical flow, which means we can leverage large sources of crowd-sourced data, such as YouTube videos. Pathak *et al.* [100] use optical flow and a graph-based algorithm to produce unsupervised segmentation maps. A network is then trained to approximate these maps, driving representation learning. The reliance on moving objects, as opposed to a moving agent, could make it harder to collect good data. Using a method based on ego-motion, the agent can promote its own representation, learning simply by moving, instead of having to find objects that move.

There is also work on using multi-modal sensory input as a source of supervision. Owens *et al.* [99] predict statistics of ambient sounds in videos. Beyond studying a single source of self-supervision, combining multiple self-supervision sources is increasingly popular. In [23], a set of self-supervision tasks are integrated via a multi-task setting. Wang *et al.* [148] propose to combine instance-level as well as category-level self-supervision. Both [23, 148] achieve better performance than a single model.

Unsupervised learning of monocular depth estimation. A single-image depth predictor can be trained from raw stereo images, by warping the right image with a depth map predicted from the left and training it to reconstruct the left image [30, 35]. This idea was extended in recent work to support fully self-supervised training on regular video, by predicting both depth and camera pose difference for pairs of nearby frames [169, 141].

Although [169, 141] are closely related to our work in the sense of unsupervised (or self-supervised) learning of depth and ego-motion from unlabeled videos, our work differs from them in two ways. First, neither of these two works emphasizes more general-purpose feature learning. Second, neither of them demonstrates their

scalability to large-scale YouTube videos. [169] requires intrinsic camera parameters that are not available for most YouTube videos; our approach relies on optical flow only. [141] only reports experimental results on standard benchmark datasets, whose scale is an order of magnitude smaller than videos we use. It is unclear whether the heuristics (*e.g.*, the manually set camera intrinsic parameters, number of motion clusters) are robust to YouTube videos in the wild.

3.3 Inducing features by learning to estimate relative depth

As a proxy task, our goal is to induce a feature representation $f(I)$ of an RGB image $I(x, y)$ by predicting its depth image $z(x, y)$, where the representation $f(I)$ could be transferred to other downstream tasks (*e.g.*, semantic segmentation) with fine-tuning. In section 3.3.1, we introduce technical details of gathering images and corresponding depth maps. In section 3.3.2, we provide details of training CNNs to learn the feature representation $f(I)$.

3.3.1 Self-Supervised Relative Depth

As described above, we automatically produce depth images for video frames by analyzing the motion of pre-existing videos. In our experiments, we used three sets of videos: YouTube videos, videos from the KITTI database [32, 31], and videos from the CityScapes database [19]. The YouTube videos consist of 135 videos taken from moving cars in major U.S. cities.[2] We call this dataset CityDriving. The stability of the camera in these videos makes them relatively easy for the depth estimation procedure. Some of the videos are extremely long, lasting several hours. The CityDriving dataset features a large number of man-made structures, pedestrians and cars. Following [73], we only keep two consecutive frames if they have moderate motion (*i.e.*, neither too slow nor too fast). To eliminate near duplicate frames, two consecutive

[2]They are crawled from a YouTube playlist, taking less than an hour.

depth maps must be at least 2 frames apart. We keep only the first one of two consecutive frames and the computed depth image. In total, we gathered 1.1M pairs of RGB images and their corresponding depth maps, where the typical resolution is 640×360. Similarly, we collect 30K and 24K pairs of RGB images and their relative depth maps for CityScapes and KITTI, respectively.

Denote the instantaneous coordinates of a point P in the environment by $(X, Y, Z)^T$, and the translational velocity of the camera in the environment by $(U, V, W)^T$. Let the motion field component (idealized optical flow) of the point P (in the image plane) be (u, v), corresponding to the horizontal and vertical image motion, respectively. The motion field can be written as the sum of translation and rotation components[3]

$$u = u_t + u_r, \qquad v = v_t + v_r, \tag{3.1}$$

where the subscript t and r denote translation and rotation, respectively. According to the geometry of perspective projection [40], the following equations hold if the motion of the camera is purely translational,

$$u_t = \frac{-U + xW}{Z}, \qquad v_t = \frac{-V + yW}{Z}, \tag{3.2}$$

where x and y are the coordinates of the point P in the image plane (the origin is at the image center).

Note that the depth Z can be estimated from either one of these equations. However, the estimate can be unstable if either u_t or v_t is small. To obtain a more robust estimate of Z, we square the two equations above and add them:

$$Z = \sqrt{\frac{(-U + xW)^2 + (-V + yW)^2}{u_t^2 + v_t^2}}. \tag{3.3}$$

[3]Any motion in the image is due to the relative motion of a world point and the camera. This addresses motion of the object, the camera, or both.

Figure 3.2: Samples of image pairs and computed translational optical flow that we use to recover the relative depth. From left to right: first images, second images, translational optical flow between input two images, and relative depth of the first images.

Because we can only recover $(U, V, W)^T$ up to scale (see below), we can only compute the depth map of an image up to scale. To induce feature representations, we use depth orderings of pixels in an image. We compute the relative depth $z \in [0, 1]$ of the pixel P as its depth percentile (divided by 100) across all estimated depth values for the image. Since these percentiles are invariant to the velocity's unknown scale, we do not need to recover the absolute scale of velocity. Examples of these automatically obtained depth maps are given in Figure 3.1 and Figure 3.2.

To compute the optical flow, we use the state-of-the-art unsupervised method [42]. It first computes sparse pixel matchings between two video frames. It then interpolates to get dense pixelwise optical flow fields from sparse matchings, where we replace the supervised edge detector [24] with its unsupervised version [73]. Based on the optical flow, we use the method proposed in [9] to recover the image motion of each pixel due to translational motion only (u_t, v_t), and also, the global camera motion $(U, V, W)^T$ up to an unknown scale factor. Specifically, the rotation of the camera can be estimated by finding the rotation such that the remaining motion, by removing the rotational component from the motion field, can be well-explained by angle fields,

which are the angle part of the motion field. This procedure produces a translational optical flow field (u_t, v_t), and a set of regions in the image corresponding to the background and different object motions, along with the motion directions (U, V, W) of those regions. We refer readers to [9] for more technical details.

In summary, to obtain the depth map of each frame from a video, we:

- compute the optical flow (u, v) between a pair of frames [42];
- estimate the translational component (u_t, v_t) of the optical flow and the direction of camera translation (U, V, W) from the optical flow, using the method of [9];
- estimate the scene depth Z using Eq. 3.3, up to an unknown scale factor, from the translational component of the optical flow and the camera direction estimate and convert it to relative depth $z \in [0, 1]$.

3.3.2 Predicting Relative Depth From a Single Image

While a CNN for predicting depth from a single image is a core component of our system, we are primarily interested in relative depth prediction as a proxy task, rather than an end in itself. We therefore select standard CNN architectures and focus on quantifying the power of the depth task for pre-training and domain adaptation, compared to using the same networks with labeled data. Specifically, we work with variants of the standard AlexNet [67], VGG16 [122], and ResNet50 [37] architectures.

Given an RGB image I, we need pixelwise predictions in the form of a depth image z, so we modify both AlexNet, VGG16, and ResNet50 to produce outputs with the same spatial resolution as the input image. In particular, we consider Fully Convolutional Networks (FCNs) [121] and an encoder-decoder with skip connections [35, 169]. Detailed discussions can be found in the experiment section.

Since the relative depth (*i.e.*, the depth percentile) is estimated over the entire image, it is essential to feed the entire image to the CNN to make a prediction.

For CityDriving and KITTI, we simply resize the input image to 224×416 and 352×1212. For CityScapes, we discard the bottom 20% portion or so of each video frame containing mainly the hood of a car, which remains static over all videos and makes the relative depth estimation inaccurate (recall our relative depth estimation is mainly based on motion information). The cropped input image is then resized to 384×992. During training, we employ horizontal flipping and color jittering for data augmentation. Since relative depth serves as a proxy, rather than an end task, even though the relative depth estimation is not always correct, the network is able to tolerate some degree of noise as shown in [100]; we can then repurpose the network's learned representation.

In all experiments, we use L_1 loss for each pixel when training for depth prediction, *i.e.*, we train networks to regress the relative depth values. All AlexNet, VGG16, and ResNet50 variants are trained for 30 epochs using the Adam optimizer [66] with momentum of $\beta_1 = 0.9, \beta_2 = 0.999$, and weight decay of 0.0005. The learning rate is 0.0001 and is held constant during the pre-training stage.

3.4 Experiments

We consider three urban scene understanding tasks: semantic segmentation, joint semantic reasoning consisting of road segmentation and object detection [136], and monocular absolute depth estimation.

3.4.1 Semantic Segmentation

We consider three datasets commonly used for evaluating semantic segmentation. Their main characteristics are summarized below:

KITTI [113]: 100 training images, 46 testing images, spatial dimensions of 370×1226, 11 classes.

Table 3.1: Comparisons of mean IoU scores of AlexNet FCN32s for semantic segmentation using different self-supervised models. CS=CityScapes, K=KITTI, CV=CamVid.

pre-training method	supervision source	CS	K	CV
supervised	ImageNet labels	**48.1**	**46.2**	**57.4**
none	-	40.7	39.6	44.0
tracking [147]	motion	41.9	42.1	50.5
moving [1]	ego-motion	41.3	40.9	49.7
watch-move [100]	motion seg.	41.5	40.8	51.7
frame-order [94]	motion	41.5	39.7	49.6
context [101]	appearance	39.7	-	37.8
object-centric [29]	appearance	39.6	39.1	48.0
colorization [70, 71]	appearance (color)	42.9	35.8	53.2
cross-channel [166]	misc.	36.8	40.8	46.3
audio [99]	video soundtrack	39.6	40.7	51.5
Ours	depth	**45.4**	**42.6**	**53.4**

CamVid [13, 12]: 367 training images, 101 validation images, 233 testing images, spatial dimensions of 720×960, 11 classes.

CityScapes [19]: 2975 training images, 500 validation images, 1525 testing images, spatial dimensions of 1024×2048, 19 classes. We conduct experiments on images at half resolution.

The first two datasets are much too small to provide sufficient data for "from scratch" training of a deep model; CityScapes is larger, but we show below that all three datasets benefit from pre-training. We use the curated annotations of the CamVid dataset released by [68]. As a classical CNN-based model for semantic segmentation, we report results of different variants of the Fully Convolutional Network (FCN) [121].

We compare our results to those obtained with other self-supervision strategies surveyed in Section 3.2. Since only AlexNet pre-trained models are available for most of the previous self-supervised methods, we also train an AlexNet. During training,

the inputs are random crops of 352×352 for KITTI and 704×704 for CamVid. Each FCN32s using different pre-training models is trained for 600 epochs with a batch size of 16 using 4 GPUs. For CityScapes, the inputs to the network are random crops of 512×512. Each FCN32s is trained for 400 epochs with a batch size of 16. In addition to the random crops, random horizontal flips and color jittering are also performed. The CNNs at this stage (learning segmentation) are trained or fine-tuned using the Adam optimizer, where weight decay is 0.0005. For the learning rate, we use 0.0001 and decrease it by a factor of 10 at the 400th epoch (300th epoch for CityScapes).

Quantitative comparisons can be found in Table 3.1.[4] Our pre-trained model performs significantly better than the model learned from scratch on all three datasets, validating the effectiveness of our pre-training. Moreover, we obtain new state-of-the-art results on all three urban scene segmentation datasets among methods that use self-supervised pre-training. In particular, our model outperforms all other self-supervised models with motion cues (the first four self-supervised models in Table 3.1).

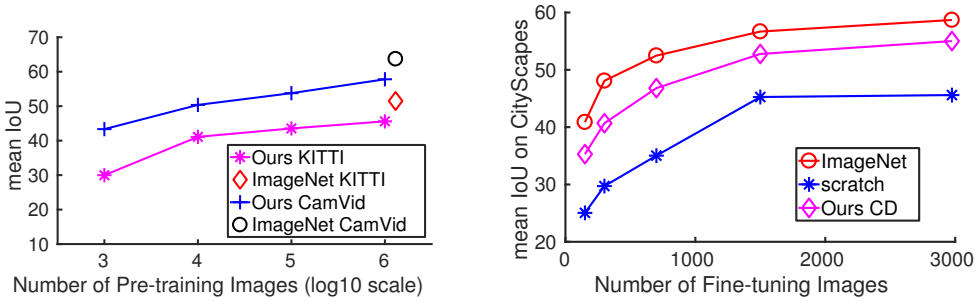
3.4.2 Ablation Studies

We perform ablation studies using VGG16 FCN32s on the semantic segmentation datasets. Specifically, we study the following aspects.

Number of pre-training images. Figure 3.3(a) demonstrates that the performance of our depth pre-trained model scales linearly with the log of the number of pre-training images on CamVid, which is similar to the conclusion of [100].

On KITTI, our pre-trained model initially has a big performance boost when the number of pre-training images increases from 1K to 10K. With enough data (more

[4]We were unable to get meaningful results with [101] on KITTI and with [56] on all three segmentation datasets.



(a) (b)

Figure 3.3: Ablation studies of performance by (a) varying number of pre-training images on KITTI and CamVid, (b) varying number of fine-tuning images of CityScapes.

Table 3.2: Mean IoU scores of semantic segmentation using different architectures on different datasets. CD=CityDriving, CS=CityScapes, CV=CamVid, and K=KITTI.

pre-training	FCN32s			FCN16s			FCN8s		
	CS	CV	K	CS	CV	K	CS	CV	K
ImageNet	**58.7**	**63.7**	**51.5**	**62.9**	**65.9**	**55.3**	**63.4**	**67.0**	**56.4**
scratch	45.4	41.0	32.4	51.3	44.1	33.1	51.6	44.3	34.2
Ours CD	55.0	57.8	45.6	57.6	**59.0**	47.7	59.8	**60.3**	48.6
Ours CD+K	56.0	**58.5**	46.0	56.9	58.8	**48.2**	58.9	60.1	49.0
Ours CD+CS	**56.2**	**58.5**	**47.4**	**58.5**	58.8	47.8	**60.5**	59.9	**49.6**

than 10K), the performance also scales linearly with the log of the number of pre-training images.

Number of fine-tuning images. Figure 3.3(b) shows that every model (ImageNet, scratch, our depth pre-trained model) benefits from more fine-tuning data on the CityScapes dataset. For both ImageNet and our depth pre-trained models, it suggests that more fine-tuning data is also beneficial for transferring the previously learned representations to a new task.

3.4.3 Domain Adaptation by Pre-Training

In the experiments described above, the two stages (pre-training on self-supervised depth prediction, followed by supervised training for segmentation) rely on data that

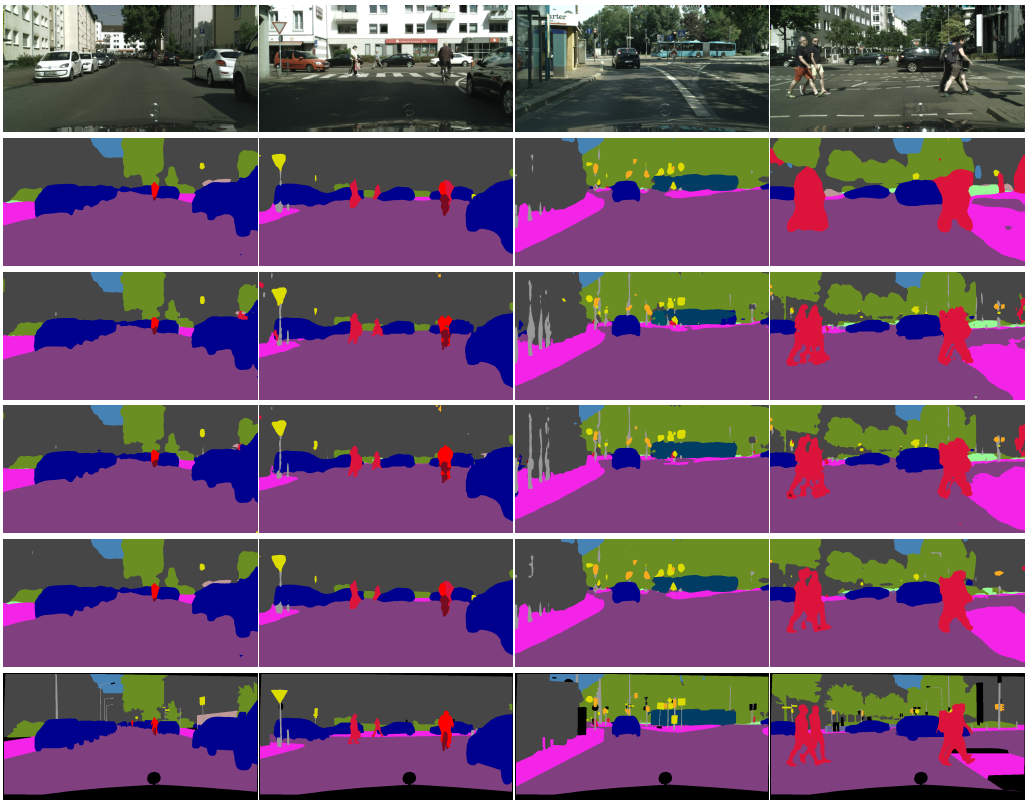


Figure 3.4: Qualitative semantic segmentation results on CityScapes. From top to bottom: input images, predictions of FCN8s with no pre-training, our FCN8s pre-trained on CityDriving, our FCN8s pre-trained on CityDriving adapted to CityScapes, ImageNet FCN8s, and ground-truth annotations. The difference between the 2nd and 3rd rows shows a clear benefit of pre-training with relative depth prediction. The difference between 3rd rows and 4th rows shows the benefit our unsupervised domain adaptation using pre-training.

come from significantly different domains. The self-supervised learning uses videos obtained from moving through North American cities. In contrast, none of the target dataset images were collected in the same geographic locations. For instance, CityScapes includes data from driving in German cities. Thus, in addition to a shift in task, the fine-tuning of the network for segmentation must also deal with a *domain shift* in the input.

CityScapes [19] and KITTI [32] make available video sequences that give temporal context to every image in the dataset. None of these extra frames are labeled, but we

can leverage them in the following way. Before training the network on segmentation, we fine-tune it, using the same self-supervised relative depth prediction task described in Section 3.3.1, on these videos. Our intuition is that this may induce some of the modifications in the network that reflect the changing distribution of the input. Then, we proceed as before to train the fine-tuned representation on the semantic segmentation data. Specifically, we fine-tune different FCNs variants based on VGG16 sequentially, *i.e.*, from FCN32s to FCN16s, and finally to FCN8s. For FCN32s, the training procedure is identical to AlexNet FCN32s described earlier. FCN16s and FCN8s are trained for the same number of epochs as FCN32s, where the learning rate is set to 0.00002 and 0.00001, respectively, and kept constant during training.

The effectiveness of our unsupervised domain adaptation for semantic segmentation can be found in Table 3.2. The last two rows of demonstrate that such fine-tuning can consistently improve the performance of a self-supervised model over all FCN variants on both CityScapes and KITTI, validating its effectiveness as a domain adaptation approach. Interestingly, we can see that while fine-tuning is helpful for FCN32s on CamVid initially, it does not help much for FCN16s and FCN8s. Perhaps this is due to the domain gap between CamVid and CityScapes/KITTI. Qualitative semantic segmentation results can be found in Fig. 3.4.

3.4.4 Joint Semantic Reasoning

Joint semantic reasoning is important for urban scene understanding, especially with respect to tasks such as autonomous driving [136]. We investigate the effectiveness of our pre-trained model and the unsupervised strategy of domain adaptation using the MultiNet architecture [136] for joint road segmentation and car detection.[5] MultiNet consists of a single encoder, using the VGG16 as backbone, and two sibling

[5]We use the author's released code https://github.com/MarvinTeichmann/MultiNet. As the scene classification data is not publicly available, we only study road segmentation and car detection here.

Table 3.3: Results of joint semantic reasoning, including road segmentation and car detection.

pre-training	Road Segmentation		Car Detection (AP)		
	F_1	AP	Easy	Medium	Hard
ImageNet	**96.33**	**92.26**	**95.59**	**86.43**	**72.28**
scratch	93.78	91.37	89.37	79.93	66.02
Ours CD	94.74	92.13	92.84	84.73	69.47
Ours CD+K	**95.66**	**92.14**	**94.31**	**85.72**	**70.50**

decoders for each task. For road segmentation, the decoder contains three upsampling layers, forming an FCN8s. The car detection decoder directly regresses the coordinates of objects. Following [136], the entire network is jointly trained using the Adam optimizer, using a learning rate of 0.00005 and weight decay of 0.0005 for 200K steps. We refer readers to [136] for more technical details.

We replace the ImageNet-trained VGG16 network with a randomly initialized one and our own VGG16 pre-trained on CityDriving using relative depth. For the road segmentation task, there are 241 training and 48 validation images. For car detection, there are 7K training images 481 validation images. Detailed comparisons on the validation set can be found in Table 3.3. We use the F_1 measure and Average Precision (AP) scores for road segmentation evaluation and AP scores for car detection. AP scores for different car categories are reported separately. We can clearly see that our pre-trained model (Ours CD) consistently outperforms the randomly initialized model (scratch in Table 3.3). Furthermore, by using the domain adaptation strategy via fine-tuning on the KITTI raw videos (Ours CD+K), we can further close the gap between an ImageNet pre-trained model. Remarkably, after fine-tuning, the F_1 score of road segmentation and AP scores for easy and medium categories of our pre-trained model are pretty close to the ImageNet counterpart's. (See last row of Table 3.3.)

3.4.5 Monocular Absolute Depth Estimation

For the monocular absolute depth estimation, we adopt the U-Net architecture [112] similar to [35, 169], which consists of a fully convolutional encoder and another fully convolutional decoder with skip connections. In order to use an ImageNet pre-trained model, we replace the encoder with the VGG16 and ResNet50 architectures. We use the training and validation set of [35], containing 22.6K and 888 images, respectively. We evaluate our model on the Eigen split [27, 35], consisting of 697 images, where ground-truth absolute depth values are captured using LiDAR at sparse pixels. Unlike [35], which uses stereo image pairs as supervision to train the network, or [169], which uses neighboring video frames as supervision to train the network (*yet camera intrinsic parameters are required*), we use the absolute sparse LiDAR depth values to fine-tune our network. The entire network (either VGG16 or ResNet50 version) is trained for 300 epochs using the Adam optimizer with a weight decay of 0.0005. The initial learning rate is 0.0001 and decreased by factor of 10 at the 200th epoch.

Detailed comparisons can be found in Table 3.4. We can observe that our pre-trained models consistently outperforms ImageNet counterparts, as well as randomly initialized models, using either VGG16 or ResNet50 architectures. It is worth noting, however, that converting relative depth to absolute depth is non-trivial. Computing relative depth (*i.e.*, percentile from absolute depth) is a non-linear mapping. The inverse transformation from relative depth to absolute depth is not unique. Following [169], we multiply our relative depth by a factor as the ratio between relative depth and absolute depth, we get pretty bad results (RMSE of 11.08 vs 4.903), showing this task is non-trivial.

Moreover, pre-training as domain adaptation also improves the performance of our pre-trained model. After fine-tuning our pre-trained model using KITTI's raw videos (Ours CD+K), our ResNet50 model achieves better results than most of the

Table 3.4: Monocular depth estimation on the KITTI dataset using the split of Eigen *et al.* [27] (range of 0-80m). For model details, Arch.=Architecture, A=AlexNet, V=VGG16, and R=ResNet50. For training data, Class.=classification, I=ImageNet, CD=CityDriving, K=KITTI, CS=CityScapes. **pp** indicates test-time augmentation by horizontally flipping the input image.

Method	Arch.	Training Data				Error Metrics				Accuracy Metrics		
		Class.	Stereo	Video	GT	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
[27]	A	I	-	-	K	0.203	1.548	6.307	0.282	0.702	0.890	0.958
[78]	A	I	-	-	K	0.202	1.614	6.523	0.275	0.678	0.895	0.965
[35]+**pp**	R	-	CS+K	-	-	0.114	0.898	4.935	0.206	0.861	0.949	0.976
[169]	V	-	-	CS+K	-	0.198	1.836	6.565	0.275	0.718	0.901	0.960
[69]	R	I	K	-	K	**0.113**	**0.741**	**4.621**	**0.189**	**0.862**	**0.960**	**0.986**
Ours	V	I	-	-	K	0.157	1.115	5.546	0.233	0.768	0.922	0.974
Ours	V	-	-	-	K	0.163	1.241	5.649	0.238	0.765	0.918	0.970
Ours	V	-	-	CD	K	0.154	1.117	5.499	0.228	0.775	0.928	0.976
Ours	V	-	-	CD+K	K	**0.148**	**1.056**	**5.317**	**0.221**	**0.791**	**0.932**	**0.977**
Ours	R	I	-	-	K	0.128	0.933	5.073	0.203	0.827	0.945	0.980
Ours	R	-	-	-	K	0.131	0.937	5.032	0.203	0.827	0.946	0.981
Ours	R	-	-	CD	K	0.128	0.901	**4.898**	0.198	0.834	0.948	0.983
Ours	R	-	-	CD+K	K	**0.125**	**0.881**	4.903	**0.195**	**0.840**	**0.951**	**0.983**

previous methods [27, 78, 35, 169]. The results are also on par with the state-of-the-art method [69].

3.5 Conclusions and Discussions

We have proposed a new proxy task for self-supervised learning of visual representations. It requires only access to unlabeled videos taken by a moving camera. Representations are learned by optimizing prediction of relative depth, recovered from estimated motion flow, from individual (single) frames. We show this task to be a powerful proxy task, which is competitive with recently proposed alternatives as a means of pre-training representations on unlabeled data. We also demonstrate a novel application of such pre-training, aimed at domain adaptation. When given videos taken by cars driven in cities, self-supervised pre-training primes the downstream ur-

ban scene understanding networks, leading to improved accuracy after fine-tuning on a small amount of manually labeled data.

Our work offers novel insights about one of the most important questions in vision today: how can we leverage unlabeled data, and in particular massive amounts of unlabeled video, to improve recognition systems. While a comprehensive picture of self-supervision methods and the role they play in this pursuit is yet to emerge, our results suggest that learning to predict relative depth is an important piece of this picture.

While the gap of the performance between self-supervised methods and their ImageNet counterparts is quickly shrinking, none of current self-supervised methods performs better than ImageNet pre-trained models on tasks involving semantics (*e.g.*, semantic segmentation and object detection). This makes pre-training on ImageNet still practically critical for many computer vision tasks. Despite this fact, this does not mean self-supervised methods are unimportant or unnecessary. The value of self-supervised methods lies in the fact that the training data can easily be scaled up without tedious and expensive human effort.

On other tasks, better performance of self-supervised methods than ImageNet counterparts has been achieved, including our monocular depth estimation and surface normal prediction [148]. Moreover, it has been shown that combining different self-supervised methods can lead to better performance [23, 148]. All of these make it very promising that representations learned using self-supervised methods may surpass what ImageNet provides us today.

CHAPTER 4

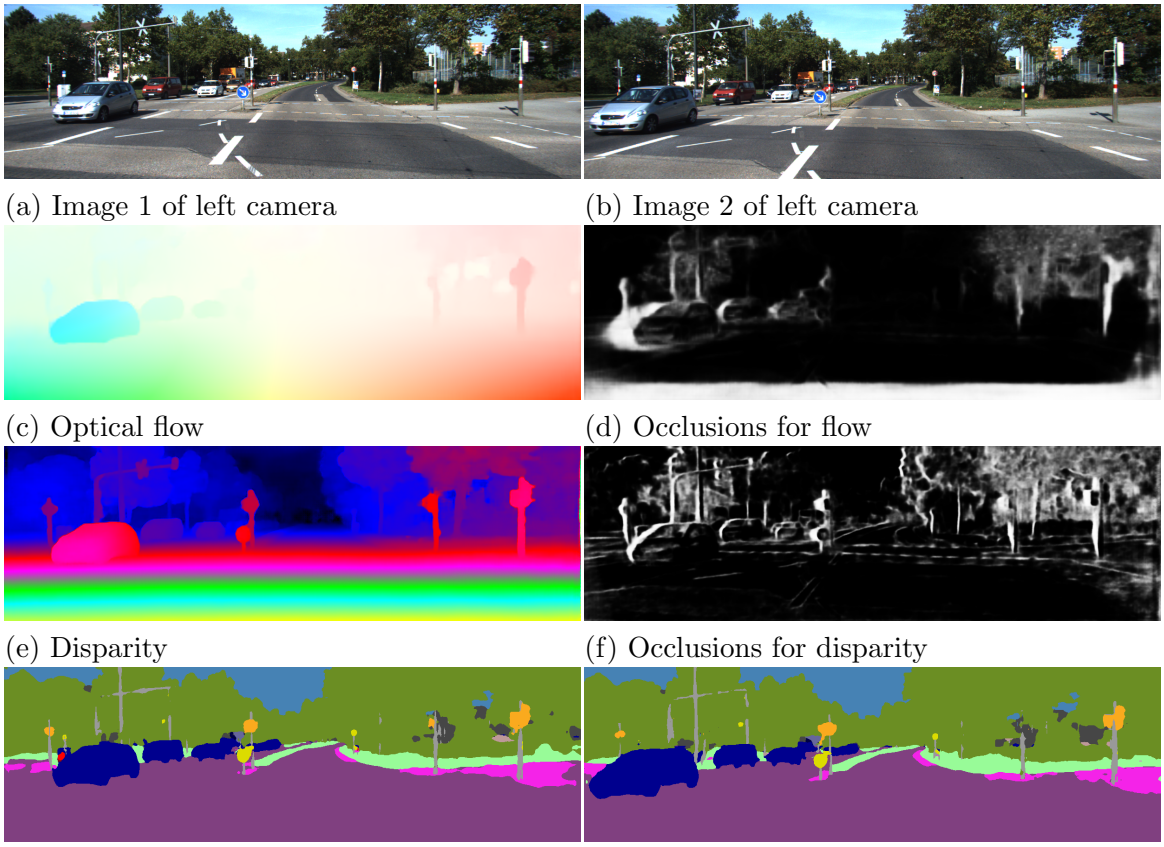
SENSE: A SHARED ENCODER FOR SCENE FLOW ESTIMATION

4.1 Overview

Scene flow estimation aims at recovering the 3D structure (disparity) and motion of a scene from image sequences captured by two or more cameras [140]. It generalizes the classical problems of optical flow estimation for monocular image sequences and disparity prediction for stereo image pairs. There has been steady and impressive progress on scene flow estimation, as evidenced by results on the KITTI benchmark [92]. State-of-the-art scene flow methods outperform the best disparity (stereo) and optical flow methods by a significant margin, demonstrating the benefit of additional information in the stereo video sequences. However, the top-performing scene flow methods [8, 143] are based on the energy minimization framework [41] and are thus computationally expensive for real-time applications, such as 3D motion capture [28] and autonomous driving [52].

Recently, a flurry of convolutional neural network (CNN)-based methods have been developed for the sub-problems of stereo and optical flow. These methods achieve state-of-the-art performance and run in real-time. However, while stereo and flow are closely-related, the top-performing networks for stereo and flow adopt significantly different architectures. Further, existing networks for scene flow stack sub-networks for stereo and optical flow together [90, 49], which does not fully exploit the structure of the two tightly-coupled problems.

As both stereo and flow rely on pixel features to establish correspondences, will the same features work for these two or more related tasks? To answer this question,



(a) Image 1 of left camera

(b) Image 2 of left camera

(c) Optical flow

(d) Occlusions for flow

(e) Disparity

(f) Occlusions for disparity

(g) Segmentation of (a)

(h) Segmentation of (b)

Figure 4.1: Given stereo videos, we train compact networks for several holistic scene understanding problems by sharing features.

we take a modular approach and build a Shared Encoder Network for Scene-flow Estimation (SENSE). Specifically, we share a feature encoder among four closely-related tasks: optical flow, stereo, occlusion, and semantic segmentation. Sharing features makes the network compact and also leads to better feature representation via multi-task learning.

The interactions among closely-related tasks further constrain the network training, ameliorating the issue of sparse ground-truth annotations for scene flow estimation. Unlike many other vision tasks, it is inherently difficult to collect ground-truth optical flow and stereo for real-world data. Training data-hungry deep CNNs often relies on synthetic data [17, 26, 90], which lacks the fine details and diversity ubiquitous in the real world. To narrow the domain gap, fine-tuning on real-world data is necessary, but the scarcity of annotated real-world data has been a serious bottleneck for learning CNN models for scene flow.

To address the data scarcity issue, we introduce a semi-supervised loss for SENSE by adding distillation and self-supervised loss terms to the supervised losses. First, no existing dataset provides ground truth annotations for all the four tasks we address. For example, the KITTI benchmark has no ground truth annotations for occlusion and semantic segmentation.[1] Thus, we train separate models for tasks with missing ground truth annotations using other annotated data, and use the pre-trained models to "supervise" our network on the real data via a distillation loss [39]. Second, we use self-supervision loss terms that encourage corresponding visible pixels to have similar pixel values and semantic classes, according to either optical flow or stereo. The self-supervision loss terms tightly couple the four tasks together and are critical for improvement in regions without ground truth, such as sky regions.

[1]Segmentation is only available for left images of KITTI 2015 [2].

Experiments on both synthetic and real-world benchmark datasets demonstrate that SENSE achieves state-of-the-art results for optical flow, while maintaining the same run-time efficiency as specialized networks for flow. It also compares favorably against state of the art on disparity and scene flow estimation, while having a much smaller memory footprint. Ablation studies confirm the utility of our design choices, and show that our proposed distillation and self-supervised loss terms help mitigate issues with partially labeled data.

To summarize, we make the following contributions:

- We introduce a modular network design for holistic scene understanding, called SENSE, to integrate optical flow, stereo, occlusion, and semantic segmentation.
- SENSE shares an encoder among these four tasks, which makes networks compact and also induces better feature representation via multi-task learning.
- SENSE can better handle partially labeled data by exploiting interactions among tasks in a semi-supervised approach; it leads to qualitatively better results in regions without ground-truth annotations.
- SENSE achieves state-of-the-art flow results while running as fast as specialized flow networks. It compares favorably against state of the art on stereo and scene flow, while consuming much less memory.

4.2 Related Work

A comprehensive survey of holistic scene understanding is beyond our scope and we review the most relevant work.

Energy minimization for scene flow estimation. Scene flow was first introduced by Vedula *et al.* [140] as the dense 3D motion of all points in an observed scene from several calibrated cameras. Several classical methods adopt energy minimization approaches, such as joint recovery of flow and stereo [44] and decoupled inference

of stereo and flow for efficiency [150]. Compared with optical flow and stereo, the solution space of scene flow is of higher dimension and thus more challenging. Vogel *et al.* [142] reduce the solution space by assuming a scene flow of piecewise rigid moving planes over superpixels. Their work first tackles scene flow from a holistic perspective and outperforms contemporary stereo and optical flow methods by a large margin on the KITTI benchmark [32].

Joint scene understanding. Motion and segmentation are chicken-and-egg problems: knowing one simplifies the other. While the layered approach has long been regarded as an elegant solution to these two problems [145], existing solutions tend to get stuck in local minima [130]. In the motion segmentation literature, most methods start from an estimate of optical flow as input, and segment the scene by jointly estimating (either implicitly or explicitly) camera motion, object motion, and scene appearance, e.g. [10, 137]. Lv *et al.* [86] show that motion can be segmented directly from two images, without first calculating optical flow. Taylor *et al.* [134] demonstrate that occlusion can also be a useful cue.

Exploiting advances in semantic segmentation, Sevilla *et al.* [120] show that semantic information is good enough to initialize the layered segmentation and thereby improves optical flow. Bai *et al.* [3] use instance-level segmentation to deal with a small number of traffic participants. Hur and Roth [46] jointly estimate optical flow and temporally consistent semantic segmentation and obtain gains on both tasks. The object scene flow algorithm [92] segments a scene into independently moving regions and enforces superpixels within each region to have similar 3D motion. The "objects" in their model are assumed to be planar and initialized via bottom-up motion estimation. Behl *et al.* [8], Ren *et al.* [108], and Ma *et al.* [87] all show that instance segmentation helps scene flow estimation in the autonomous setting. While assuming a rigid motion for each individual instance works well for cars, this assump-

tion tends to fail in general scenes, such as Sintel, on which our holistic approach achieves state-of-the-art performance.

The top-performing energy-based approaches are too computationally expensive for real-time applications. Here we present a compact CNN model to holistically reason about geometry (disparity), motion (flow), and semantics, which runs much faster than energy-based approaches.

End-to-end learning of optical flow and disparity. Recently CNN based methods have made significant progress on optical flow and disparity, two sub-problems of scene flow estimation. Dosovitskiy *et al.* [26] first introduce two CNN models, FlowNetS and FlowNetC, for optical flow and bring about a paradigm shift to optical flow and disparity estimation. Ilg *et al.* [48] propose several technical improvements, such as dataset scheduling and stacking basic models into a big one, *i.e.*, FlowNet2. FlowNet2 has near real-time performance and obtains competitive results against hand-designed methods. Ilg *et al.* [49] stack networks for flow, disparity together for the joint task of scene flow estimation. However, there is no information sharing between the networks for flow and disparity. Ranjan and Black [104] introduce a spatial pyramid network that performs on par with FlowNetC but has more than 100 times fewer parameters, due to the use of two classical principles: pyramids and warping. Sun *et al.* [131] develop a compact yet effective network, called PWC-Net, which makes frequent use of three principles to construct the network: pyramids of learnable features, warping operations, and cost volume processing. PWC-Net obtains state-of-the-art performance on two major optical flow benchmarks.

The FlowNet work also inspired new CNN models for stereo estimation [65, 18, 159]. Kendall *et al.* [65] concatenate features to construct the cost volume, followed by 3D convolutions. The 3D convolution becomes commonly-used for stereo but is computationally expensive in speed and memory. Chang and Chen [18] introduce a pyramid pooling module to exploit context information for establishing correspon-

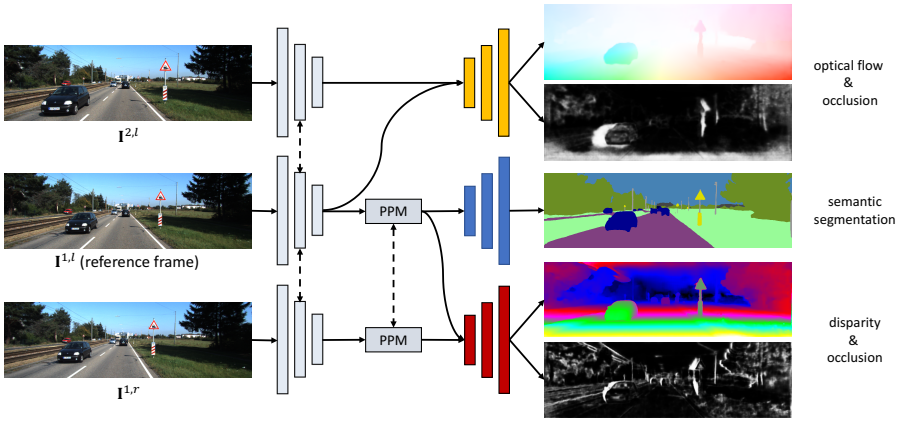


Figure 4.2: Illustration of network design. Dashed arrows indicate shared weights. We have a single encoder for all input images and all different tasks and keep different decoders for different tasks. On the right, from top to bottom are: optical flow, forward occlusion mask, semantic segmentation, disparity, and disparity occlusion. The PPM (Pyramid Pooling Module) is not helpful for optical flow estimation. But thanks to the modular network design, we can flexibly configure the network.

dences in ambiguous regions. Yang *et al.* [159] incorporate semantic cues to tackle textureless regions. Yin *et al.*cast optical flow and disparity estimations as probabilistic distribution matching problems [160] to provide uncertainty estimation. They do not exploit the shared encoder of the two tasks as we do.

Existing scene flow networks [49, 87, 90] stack independent networks for disparity and flow together. We are interested in exploiting the interactions among multiple related tasks to design a compact and effective network for holistic scene understanding. Our holistic scene flow network performs favorably against state of the art while being faster for inference and consuming less memory. In particular, we show the benefit of sharing the feature encoder between different tasks, such as flow and disparity.

Self-supervised learning from videos. Supervised learning often uses synthetic data, as it is hard to obtain ground truth optical flow and disparity for real-world videos. Recently self-supervised learning methods have been proposed to learn scene flow by minimizing the data matching cost [171] or interpolation errors [61, 79].

However, the self-supervised methods have not yet achieved the performance of their supervised counterparts.

4.3 Semi-Supervised Scene Flow Estimation

We follow the problem setup of the KITTI scene flow benchmark [92], as illustrated in Fig. 4.2. The inputs are two stereo image pairs over time $\left(\mathbf{I}^{1,l}, \mathbf{I}^{2,l}, \mathbf{I}^{1,r}, \mathbf{I}^{2,r}\right)$, where the first number in the superscript indicates the time step and the second symbol denotes the left or right camera. To save space, we will omit the superscript if the context is clear. We want to estimate optical flow $\mathbf{F}^{1,l}$ from the first left image to the second left image and disparity $\mathbf{D}^{1,l}$ and $\mathbf{D}^{2,l}$ from the left image to the right image at the first and second frames, respectively. We also consider occlusion between two consecutive frames $\mathbf{O}_F^{1,l}$ and between the two sets of stereo images $\mathbf{O}_D^{1,l}$ and $\mathbf{O}_D^{2,l}$, as well as semantic segmentation for the reference (first left) image, *i.e.*, $\mathbf{S}^{1,l}$. These extra outputs introduce interactions between different tasks to impose more constraints in the network training. Further, we hypothesize that sharing features among these closely-related tasks induces better feature representations.

We will first introduce our modular network design in Section 4.3.1, which shares an encoder among different tasks and supports flexible configurations during training. We will then explain our semi-supervised loss function in Section 4.3.2, which enables learning with partially labeled data.

4.3.1 Modular Network Design

To enable feature sharing among different tasks and allow flexible configurations during training, we design the network in a modular way. Specifically, we build our network on top of PWC-Net [131], a compact network for optical flow estimation. PWC-Net consists of an encoder and a decoder, where the encoder takes the input images and extracts features at different hierarchies of the network. The decoder

is specially designed with domain knowledge of optical flow. The encoder-decoder structure allows us to design a network in a modular way, with a single shared encoder and several decoders for different tasks.

Shared encoder. The original encoder of PWC-Net, however, is not well-suited to multiple tasks because of its small capacity. More than 80% of the parameters of PWC-Net are concentrated in the decoder, which uses DenseNet [43] blocks at each pyramid level. The encoder consists of plain convolutional layers and uses fewer than 20% of the parameters. While sufficient for optical flow, the encoder does not work well enough for disparity estimation. To make the encoder versatile for different tasks, we make the following modifications. First, we reduce the number of feature pyramid levels from 6 to 5, which reduces the number of parameters by nearly 50%. It also allows us to borrow the widely-used 5-level ResNet-like encoder architecture [18, 37], which has been proven to be effective in a variety of vision tasks. Specifically, we replace plain CNN layers with residual blocks [37] and add Batch Normalization layers [50] in both encoder and decoder. With these modifications, the new model has slightly fewer parameters but gives better disparity estimation results and also better flow (Table 4.1).

Decoder for disparity. Next we explain how to adapt PWC-Net to disparity estimation between two stereo images. Disparity is a special case of optical flow computation, with correspondences lying on a horizontal line. As a result, we need only to build a 1D cost volume for disparity, while the decoder of the original PWC-Net constructs a 2D cost volume for optical flow. Specifically, for optical flow, a feature at $p=(x, y)$ in the first feature map is compared to features at $q \in [x-k, x+k]\times[y-k, y+k]$ in the warped second feature map. For disparity, we need only to search for correspondences by comparing p in the left feature map to $q \in [x-k, x+k]\times y$ in the warped right feature map. We use $k=4$ for both optical flow and disparity estimations. Across the

feature pyramids, our decoder for disparity adopts the same warping and refinement process as PWC-Net.

To further improve disparity estimation accuracy, we investigate more design choices. First, we use the Pyramid Pooling Module (PPM) [168] to aggregate the learned features of input images across multiple levels. Second, the decoder outputs a disparity map one fourth the size of the input resolution, which tends to have blurred disparity boundaries. As a remedy, we add a simple hourglass module widely used in disparity estimation [18]. It takes a twice upsampled disparity, a feature map of the first image, and a warped feature map of the second image to predict a residual disparity that is added to the upsampled disparity. Both the PPM and hourglass modifications lead to significant improvements in disparity estimation. They are not helpful for optical flow estimation though, indicating that the original PWC-Net is well designed for optical flow. The modular design allows us to flexibly configure networks that work for different tasks, as shown in Fig. 4.2.

Decoder for segmentation. To introduce more constraints to network training, we also consider semantic segmentation. It encourages the encoder to learn some semantic information, which may help optical flow and disparity estimations. For semantic segmentation decoder, we use the UPerNet [154] for its simplicity.

Occlusion estimation. For occlusion predictions, we add sibling branches to optical flow or disparity decoders to perform pixel-wise binary classification, where 1 means fully occluded. Adding such extra modules enables holistic scene understanding that helps us to induce better feature representations in the shared encoder and use extra supervision signals for network training to deal with partially labeled data, which is discussed in Section 4.3.2. Critically, for scene flow estimation, the shared encoder results in a more compact and efficient model. For optical flow and disparity estimations, we can combine modules as needed during training, with no influence on

the inference time. For scene flow estimation, extra modules can be used optionally, depending on configuration. See explanations in Section 4.4.2.

4.3.2 Semi-Supervised Loss

No fully labeled datasets are available to directly train our holistic scene flow network. For example, KITTI has no ground-truth occlusion masks. Even for optical flow and disparity ground-truths, only around 19% of pixels of the KITTI data have annotations due to the difficulty in data capturing. The synthetic SceneFlow dataset [90] has no ground truth for semantic segmentation. To address these issues, we introduce our semi-supervised loss functions, which consist of supervised, distillation, and self-supervised loss terms.

Supervised loss. When corresponding ground-truth annotations are available, we define our supervised loss as

$$\mathcal{L}_{sp} = (\mathcal{L}_F + \mathcal{L}_{O_F}) + (\mathcal{L}_D + \mathcal{L}_{O_D}), \tag{4.1}$$

where $\mathcal{L}_F$ and $\mathcal{L}_{O_F}$ are loss terms for estimating optical flow and its corresponding occlusion. $\mathcal{L}_D$ and $\mathcal{L}_{O_D}$ are the loss terms for estimating disparity and its corresponding occlusion. $\mathcal{L}_F$ is defined across multiple pyramid levels as

$$\mathcal{L}_F = \sum_{i=1}^{N_F} \omega_i \sum_p \rho\left(\mathbf{F}_i(p), \hat{\mathbf{F}}_i(p)\right), \tag{4.2}$$

where ω_i denotes optical flow and disparity weights at pyramid level i, N_F is the number of pyramid levels, and $\rho(\cdot, \cdot)$ is a loss function measuring the similarity between the ground-truth $\mathbf{F}_i(p)$ and estimated optical flow $\hat{\mathbf{F}}_i(p)$ at pixel p. Disparity and occlusion loss functions, $\mathcal{L}_D$, $\mathcal{L}_{O_F}$, and $\mathcal{L}_{O_D}$ are defined in a similar way. We use L_2 and `smooth_l1` [34, 18] loss for optical flow and disparity estimations, respectively. For the occlusions, we use binary cross entropy loss when ground-truth annotations are

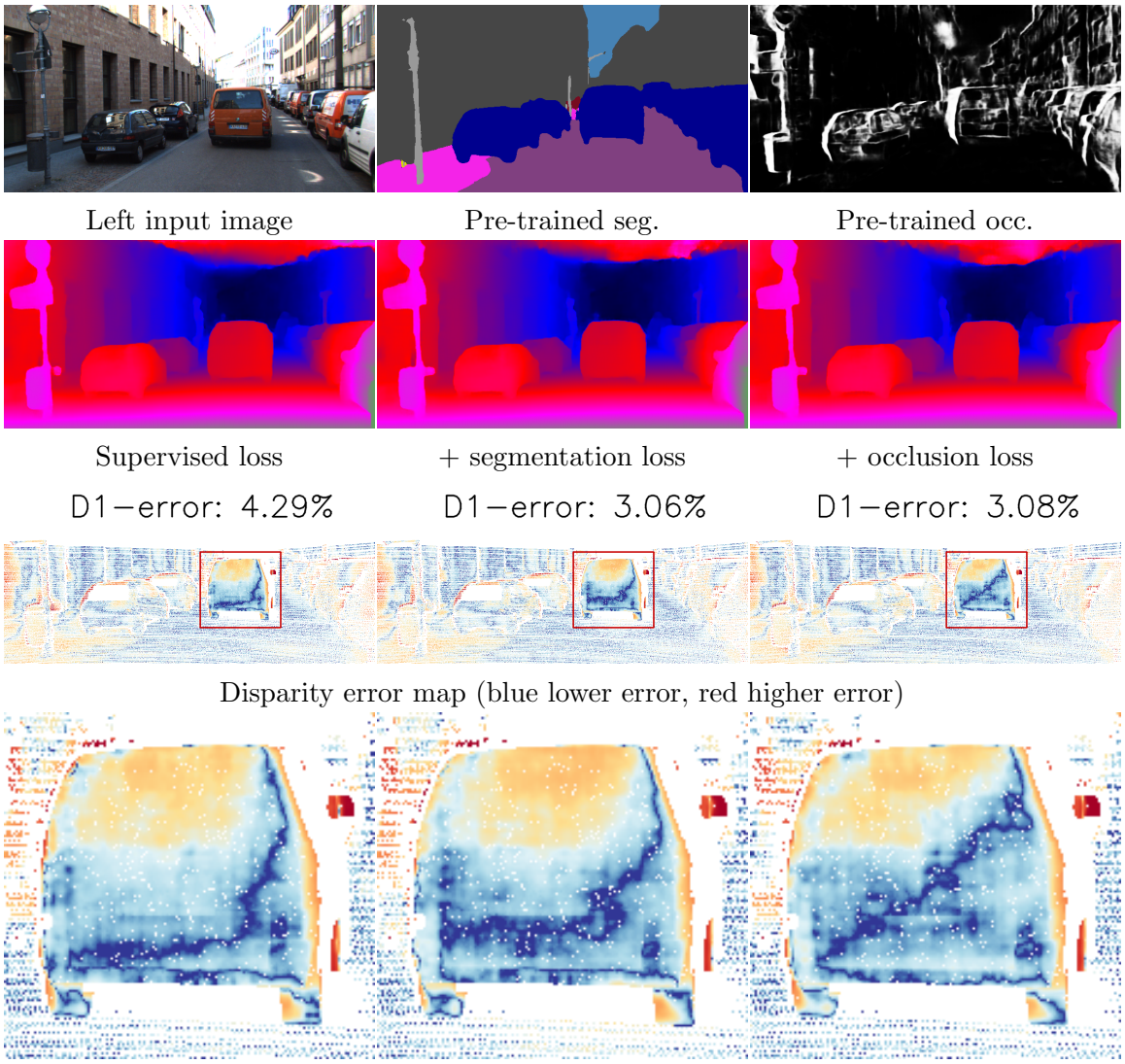


Figure 4.3: Effects of adding distillation losses for semantic segmentation (middle) and occlusion (right) to the supervised loss.

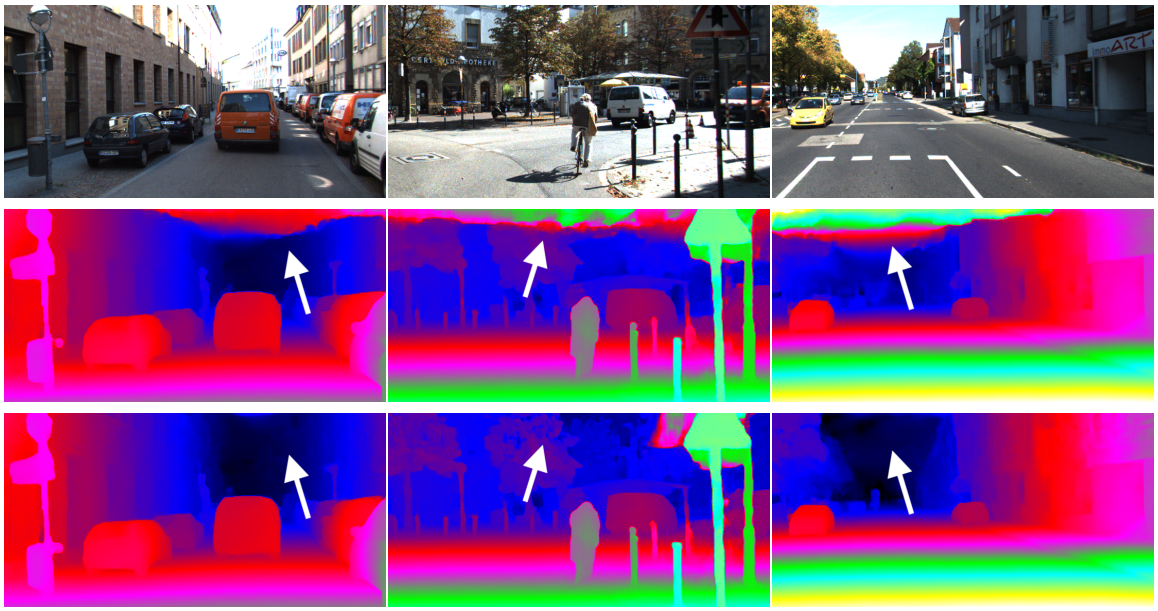


Figure 4.4: Illustration of effectiveness of self-supervised loss. From top to bottom: input images, disparity estimations *without* using self-supervised loss, and disparity estimations *with* using self-supervised loss. We can see self-supervised loss helps greatly reduce artifacts in the sky region.

available (*e.g.*, on FlyingThings3D [90]). For semantic segmentation, only ground-truth annotations of the left images are available for KITTI2015. We empirically found using distillation loss only introduced below yields better accuracy.

Distillation loss. For occlusion estimation and semantic segmentation tasks, ground-truth annotations are not always available. They are important, however, during network training. For instance, on KITTI, supervised loss can only be computed on sparsely annotated pixels. Adding extra supervision for occlusion estimation is helpful for the network to extrapolate optical flow and disparity estimations to regions where ground-truth annotations are missing, yielding visually appealing results.

We find the occlusion estimations provided by a pre-trained model on synthetic data are reasonably good, as shown in Fig. 4.3. As a soft supervision, we encourage the occlusion estimations of the network during training do not deviate much from what it learned in the pre-training stage. Therefore, we simply use the estimations of a pre-

trained network as pseudo ground-truth and `smooth_l1` loss function during training, computed in multiple pyramid levels as $\mathcal{L}_F$ and $\mathcal{L}_D$. Adding extra supervision using distillation loss for occlusion is helpful for reducing artifacts in disparity estimation, as shown in Fig. 4.3.

For semantic segmentation, we use the distillation loss formulation proposed in [39]. Specifically, semantic segmentation distillation loss $\mathcal{L}_{S_d}$ for a single pixel p (omitted here for simplicity) is defined as

$$\mathcal{L}_{S_d} = T \sum_{i=1}^{C} \tilde{y}_i \log \hat{y}_i, \quad \tilde{y}_i = \frac{\exp^{-z_i/T}}{\sum_k \exp^{-z_k/T}}, \tag{4.3}$$

where C is the number of segmentation categories. z_i and $\tilde{y}_i$ come from a more powerful teacher segmentation model, where z_i is the output for the i-th category right before the `softmax` layer, also known as `logit`. $\tilde{y}_i$ is "softened" posterior probability for the i-th category, controlled by the hyper-parameter T [39]. We empirically found $T=1$ works well on a validation set. $\hat{y}_i$ is the estimated posterior probability of our model. The distillation is aggregated over all pixels in training images.

Self-supervised loss. To further constrain the network training, we also define self-supervised loss. Optical flow and disparity are defined as correspondence between two input images. We can therefore compare two corresponding pixels defined by either optical flow or disparity as supervision for network training.

The most straightforward metric is to compare values between two corresponding pixels that are visible in both frames, known as photometric consistency. In a single pyramid level, it is defined as $\mathcal{L}_{PC} =$

$$\|\mathbf{I}^l - g(\mathbf{I}^r, \mathbf{D}^l)\|_1 \odot \bar{\mathbf{O}}_D + \|\mathbf{I}^1 - g(\mathbf{I}^2, \mathbf{F}^1)\|_1 \odot \bar{\mathbf{O}}_F, \tag{4.4}$$

where $g(\cdot,\cdot)$ is the differentiable warping function, $\bar{\mathbf{O}} = 1 - \mathbf{O}$, $\odot$ denotes element-wise multiplication followed by summation, and we omit some superscripts when the

context is clear. This loss term reasons about occlusion by modulating the consistency loss using the occlusion map and tightly couples occlusion with optical flow and stereo.

As photometric consistency is not robust to lighting changes, we further introduce semantic consistency, encouraging two corresponding pixels to have similar semantic segmentation posterior probability. Specifically, this semantic consistency is defined as $\mathcal{L}_{SC} =$

$$\|\tilde{\mathbf{y}}^l - g(\tilde{\mathbf{y}}^r, \mathbf{D}^l)\|_1 \odot \bar{\mathbf{O}}_D + \|\tilde{\mathbf{y}}^1 - g(\tilde{\mathbf{y}}^2, \mathbf{F}^1)\|_1 \odot \bar{\mathbf{O}}_F, \tag{4.5}$$

where $\tilde{\mathbf{y}}$ denotes a posterior probability image coming from the teacher segmentation network used in Eq.(4.3). Unlike raw pixel values, the segmentation posterior probability is more robust to lighting changes.

Finally, we consider the structural similarity loss

$$\begin{aligned}\mathcal{L}_{SS} = &\gamma_D \left(1 - SS(\mathbf{I}^l, \mathbf{I}^l \otimes \mathbf{O}_D + g(\mathbf{I}^r, \mathbf{D}^l) \otimes \bar{\mathbf{O}}_D)\right) + \\ &\gamma_F \left(1 - SS(\mathbf{I}^1, \mathbf{I}^1 \otimes \mathbf{O}_F + g(\mathbf{I}^2, \mathbf{F}^1) \otimes \bar{\mathbf{O}}_F)\right),\end{aligned} \tag{4.6}$$

where $\otimes$ indicates element-wise multiplications only. $SS(\cdot, \cdot)$ is a differentiable function that outputs a single scalar value to measure the structural similarity between two input images [167]. Note that for occluded pixels in the warped image, their values are replaced with values of pixels at the same position in the left/first image.

There exist trivial solutions for minimizing Eq.(4.4) and Eq.(4.5) by setting $\mathbf{O}_D$ and $\mathbf{O}_F$ to all ones. We thus add regularization terms

$$\mathcal{L}_{REG} = \beta_D \sum_p \mathbf{O}_D(p) + \beta_F \sum_p \mathbf{O}_F(p), \tag{4.7}$$

Although the self-supervised photometric and structural similarity loss terms have been studied in previous work [55, 35], our definition differs from theirs in that we

model occlusions. On one hand, we avoid defining loss terms in the occluded regions. On the other hand, these self-supervised terms provide modulation for the occlusion estimation as well. Thus, our networks tightly couple these four closely-related tasks together.

Our final semi-supervised loss consists of supervised, distillation, and self-supervised loss terms. More details can be found in the supplementary material.

4.3.3 Rigidity-based Warped Disparity Refinement for Scene Flow Estimation

Determine rigidity area. Given the estimated semantic segmentation labels of the first left frame $\mathbf{S}^{1,l}$, we select pixels as static rigid regions by removing pixels which have a semantic label of vehicle, pedestrian, cyclist, or sky. This step gives a conservative selection of static regions with points not at infinity. The output is a binary mask $\mathbf{B}$ with the label 1 indicating static rigid region. Since the semantic segmentation can be inaccurate at object boundary, we further perform an erosion operation with a size of 10 on the static rigid region mask $\mathbf{B}$.

Estimate rigid flow induced by camera motion. Given the estimated flow $\mathbf{F}^1$ and disparity $\mathbf{D}^1$ of the left frame, we calculate the ego-motion flow induced by the rigid camera motion by minimizing the weighted errors between predicted rigid flow $\mathbf{F}^1_{\mathbf{R}}$ and optical flow $\mathbf{F}^1$ in the background region pixels $\mathbf{x} \in \mathbf{R}^2$:

$$\arg\min_{\xi} \quad \mathbf{r}^T(\xi;\mathbf{x})\mathbf{W}\mathbf{r}(\xi;\mathbf{x}) \tag{4.8}$$

$$\mathbf{r}(\xi;\mathbf{x}) = \mathbf{F}^1(\mathbf{x}) - \mathbf{F}^1_{\mathbf{R}}(\xi;\mathbf{x}) \tag{4.9}$$

$$\mathbf{F}^1_{\mathbf{R}}(\xi;\mathbf{x}) = \mathcal{W}(\xi;\mathbf{x},\mathbf{D}^1(\mathbf{x})) - \mathbf{x} \tag{4.10}$$

where $\mathbf{x} \in \mathbf{R}^2$ denotes the pixels in 2D image space which are within the rigid areas $\mathbf{B}$. $\mathcal{W}(\xi;\mathbf{x},\mathbf{D}^1)$ is the warping function which transforms the pixels $\mathbf{x}$ and

its corresponding disparity $\mathbf{D}^1(\mathbf{x})$ with an estimated transform $\xi \in \mathbf{SE}(3)$. $\mathbf{W}$ is a diagonal weight matrix that depends on residuals using Huber weight function.

We solve equation 4.8 as an iteratively reweighted least-square problem using Gauss-Newton update:

$$\delta\xi = (\mathbf{J}^T\mathbf{W}\mathbf{J})^{-1}\mathbf{J}^T\mathbf{W}\mathbf{r} \tag{4.11}$$

$$\xi = \xi \circ \delta\xi \tag{4.12}$$

where $\circ$ indicates the right composition of $\xi \in \mathbf{SE}(3)$. $\mathbf{J}$ is the Jacobian matrix of $\partial\mathbf{F}^1_{\mathbf{R}}(\xi)/\partial\xi$.

Suppose $\mathbf{K}$ is the intrinsic matrix for a pin-hole camera without distortion, which can be parameterized as (f_x, f_y, c_x, c_y) with f_x, f_y as its focal length and c_x, c_y as its offset along the two axes. The baseline of the stereo pair is b. We define the 3D point $\mathbf{p} = (p_x, p_y, p_z)$ as $\mathbf{p} = (f_x b/\mathbf{D}^1(\mathbf{x}))\mathbf{K}^{-1}\mathbf{x}$. Through chain-rule, we can derive the analytical form of the Jacobian matrix $\mathbf{J}$. To simplify the computation, we use the inverse depth parameterization $\mathbf{p} = (p_u/p_d, p_v/p_d, 1/p_d)$ in which $\mathbf{x} = (p_u, p_v) \in \mathbf{R}^2$ is the pixel of coordinate of $\mathbf{x}$ and p_d is the inverse depth as $p_d = \mathbf{D}^1(\mathbf{x})/(f_x b)$. Thus, we obtain the Jacobian matrix at a pixel $\mathbf{x}$ as:

$$\begin{bmatrix} -p_u p_v f_x & (1+p_u^2) f_x & -p_v f_x & p_d f_x & 0 & -p_u p_d f_x \\ -(1+p_v^2) f_y & p_u p_v f_y & p_u f_y & 0 & p_d f_y & -p_u p_d f_y \end{bmatrix} \tag{4.13}$$

We perform the Gauss-Newton update if the absolute residual error is bigger than 10^{-6} with a maximum of 20 iterations. All operations are implemented in Pytorch and executed in GPU. The running time of the total optimization varies between 0.03s and 0.2s, according to the number of iterations. In average, the optimization step takes 0.1s for KITTI image of resolution 375x1242.

The final optical flow $\mathbf{F}$ is an element-wise linear composition of $\mathbf{F}^1$ and $\mathbf{F}^1_{\mathbf{R}}$ as:

$$\mathbf{F} = (\mathbf{1} - \mathbf{B}) \otimes \mathbf{F}^1 + \mathbf{B} \otimes \mathbf{F}^1_{\mathbf{R}} \tag{4.14}$$

where $\otimes$ indicates element-wise multiplications.

Estimate warped second frame rigid disparity. Given the estimated optimal $\xi^\star$, we define the disparity $\mathbf{D}^{1\to2}_{\mathcal{W},\mathbf{R}}$ of the second frame warped from the first frame following the optimal rigid transform $\xi^\star$:

$$\mathbf{D}^{1\to2}_{\mathcal{W},\mathbf{R}} = \mathcal{W}^{1\to2}_{\mathbf{D}}(\xi^\star; \mathbf{D}^1) \tag{4.15}$$

where $\mathcal{W}^{1\to2}_{\mathbf{D}}(\cdot)$ defines the disparity channel output from the warping function $\mathcal{W}(\cdot)$ through a forward warping. Given the forward optical flow $\mathbf{F}^1$, the warped disparity of the second frame can be computed through an inverse warping $\mathcal{W}^{2\to1}_{\mathbf{D}}$ as:

$$\mathbf{D}^{2\to1}_{\mathcal{W}} = \mathcal{W}^{2\to1}_{\mathbf{D}}(\mathbf{F}^1, \mathbf{D}^2) \tag{4.16}$$

We find that the disparity through forward warping $\mathbf{D}^{1\to2}_{\mathcal{W},\mathbf{R}}$ gives more accurate disparity in static region and can better handle occlusions. The final warped disparity $\mathbf{D}^2_{\mathcal{W}}$ is a element-wise linear composition of $\mathbf{D}^{2\to1}_{\mathcal{W}}$ and $\mathbf{D}^{1\to2}_{\mathcal{W},\mathbf{R}}$ as:

$$\mathbf{D}^2_{\mathcal{W}} = (\mathbf{1} - \mathbf{B}) \otimes \mathbf{D}^{2\to1}_{\mathcal{W}} + \mathbf{B} \otimes \mathbf{D}^{1\to2}_{\mathcal{W},\mathbf{R}} \tag{4.17}$$

Note that both warping function cannot deal with out-of-boundary pixels due to two-view occlusion. This can be resolved by the additional refinement network detailed in the following section.

4.4 Experiments

4.4.1 Implementation Details

We perform pre-training on the synthetic SceneFlow dataset and fine-tuning on Sintel and KITTI, respectively.

Synthetic SceneFlow Dataset. We use the subset of FlyingThings3D used in [49], Monkaa, and Driving for pre-training. We remove images whose maximum optical flow magnitude is greater than 500. We end up using 128,753 samples for training.

Supervised training is performed for the pre-training with ground-truth annotations of optical flow, disparity, and their associated occlusions. The loss function is defined as

$$\mathcal{L}_{sp} = (\mathcal{L}_F + \mathcal{L}_{O_F}) + 0.25 \times (\mathcal{L}_D + \mathcal{L}_{O_D}) \,. \tag{4.18}$$

For Monkaa and Driving, since only optical flow and disparity annotations are available, we only set $\mathcal{L}_{O_D}$ and $\mathcal{L}_{O_F}$ to 0 for training data sampled from Monkaa and Driving.

During training, we use color jittering, including randomly changing gamma value, changing brightness, changing contrast, and adding Gaussian noise, for both optical flow and disparity training. Additionally, we use random crops and vertical flips for stereo training images. The crop size is 256×512. For optical flow training images, we perform extensive data augmentations including random crop, translation, rotation, zooming, squeezing, and horizontal and vertical flip, where the crop size is 384×640. The network is trained for 100 epochs with a batch size of 8 using the Adam optimizer [66]. We use synchronized Batch Normalization [154] to ensure there are enough training samples for estimating Batch Normalization layers' statistics when using multiple GPUs. The initial learning rate is 0.001 and decreased by factor of 10 after 70 epochs.

Sintel. We fine-tune the pre-trained model on Sintel. Sintel training data provides optical flow, disparity, and their corresponding occlusion annotations. We therefore use the same loss function as used for the pre-training.

During training, we apply the same color jittering used for pre-training. Similarly we use random crops and vertical flips for stereo training images with crop size of 384×768. For optical flow training images, we perform extensive data augmentations as well including random crop, translation, rotation, zooming, squeezing, and horizontal and vertical flip, where the crop size is 384×768.

Synchronized Batch Normalization is used with batch size of 8. The model is first trained for 500 epochs using the Adam optimizer with an initial learning rate of 0.0005, which is decreased by factor of 2 after every 100 epochs. The weight decay is 0.0004. After 500-epoch training is finished, we keep fine-tuning the model for another 500 epochs using Adam with an initial learning rate of 0.0002, which is decreased by factor of 2 after every 100 epochs. The weight decay remains 0.0004.

KITTI. On KITTI (including KITTI2012 and KITTI2015), we use both supervised loss and semi-supervised loss. The final loss is defined as

$$\mathcal{L} = \underbrace{\mathcal{L}_F + \mathcal{L}_D}_{\text{supervised loss}} + \underbrace{\alpha_O\big(\mathcal{L}_{O_{Fd}} + \mathcal{L}_{O_{Dd}}\big) + \alpha_{S_d}\mathcal{L}_{S_d}}_{\text{distillation loss}} + \underbrace{\alpha_{PC}\mathcal{L}_{PC} + \alpha_{SC}\mathcal{L}_{SC} + \mathcal{L}_{SS} + \mathcal{L}_{REG}}_{\text{self-supervised loss}}, \tag{4.19}$$

where $\mathcal{L}_{O_{Fd}}$ and $\mathcal{L}_{O_{Dd}}$ are distillation loss for optical flow occlusion and disparity occlusion, respectively. They are defined as `smooth_L1` loss between the pseudo ground-truth (*i.e.*, estimations from a model pre-trained on synthetic SceneFlow dataset) and estimations from the model being trained. On the validation set, we empirically found $\alpha_O = 0.05, \alpha_{S_d} = 1, \alpha_{PC} = 0.5, \alpha_{SC} = 0.5$ work well. For the SSIM loss, we use $\gamma_D = 0.005 \times C_H \times C_W$ and $\gamma_F = 0.01 \times C_H \times {C_W}^2$, where C_H and C_W are

[2]In our definition of SSIM loss, the function $SS(\cdot,\cdot)$ gives a single scalar value.

Table 4.1: Average EPE results on MPI Sintel optical flow dataset. "-ft" means fine-tuning on the MPI Sintel *training* set and the numbers in parentheses are results on the data the methods have been fine-tuned on.

Methods	Training		Test		Time
	Clean	Final	Clean	Final	(s)
FlowFields [4]	-	-	3.75	5.81	28.0
MRFlow [153]	1.83	3.59	**2.53**	5.38	480
FlowFieldsCNN [5]	-	-	3.78	5.36	23.0
DCFlow [155]	-	-	3.54	5.12	8.60
SpyNet-ft [104]	(3.17)	(4.32)	6.64	8.36	0.16
FlowNet2 [48]	2.02	3.14	3.96	6.02	0.12
FlowNet2-ft [48]	(1.45)	(2.01)	4.16	5.74	0.12
LiteFlowNet [45]	(1.64)	(2.23)	4.86	6.09	0.09
PWC-Net [131]	2.55	3.93	-	-	**0.03**
PWC-Net-ft [131]	(1.70)	(2.21)	3.86	5.13	**0.03**
FlowNet3 [49]	2.08	3.94	3.61	6.03	0.07
FlowNet3-ft [49]	(**1.47**)	(2.12)	4.35	5.67	0.07
SENSE	1.91	3.78	-	-	**0.03**
SENSE-ft	(1.54)	(**2.05**)	3.60	**4.86**	**0.03**

crop height and width, respectively. For the regularization term, we empirically set $\beta_F = \beta_D = 0.5$.

During training, we use similar color jittering used in pre-training but with a probability of 0.5. Similarly we use random crops and vertical flips for stereo training images with crop size of 320×768. For optical flow training images, we perform extensive data augmentations as well including random crop, translation, rotation, zooming, squeezing, and horizontal and vertical flip, where the crop size is 320×768.

Synchronized Batch Normalization is used with batch size of 8. The model is fine-tuned for 1,500 epochs using the Adam optimizer with an initial learning rate of 0.001, which is decreased by factor of 2 at epochs of 400, 800, 1,000, 1200, and 1,400. The weight decay is 0.0004. Another round of fine-tuning is followed with an initial learning rate of 0.0002, which is decreased by factor of 2 at epochs of 400, 600, 800, and 900.

Table 4.2: Results on the KITTI optical flow dataset. "-ft" means fine-tuning on the KITTI *training* set and the numbers in the parenthesis are results on the data the methods have been fine-tuned on.

Methods	KITTI 2012			KITTI 2015		
	AEPE	AEPE	Fl-Noc	AEPE	Fl-all	Fl-all
	train	*test*	*test*	*train*	*train*	*test*
FlowFields [4]	-	-	-	-	-	19.80%
MRFlow [153]	-	-	-	-	14.09 %	12.19 %
DCFlow [155]	-	-	-	-	15.09 %	14.83 %
SDF [3]	-	2.3	3.80%	-	-	11.01 %
MirrorFlow [47]	-	2.6	4.38%	-	9.93%	10.29%
SpyNet-ft [104]	(4.13)	4.7	12.31%	-	-	35.07%
FlowNet2 [48]	4.09	-	-	10.06	30.37%	-
FlowNet2-ft [48]	(1.28)	1.8	4.82%	(2.30)	(8.61%)	10.41 %
LiteFlowNet [45]	(1.26)	1.7	-	(2.16)	(8.16%)	10.24 %
PWC-Net [131]	4.14	-	-	10.35	33.67%	-
PWC-Net-ft [131]	(1.45)	1.7	4.22%	(2.16)	(9.80%)	9.60%
FlowNet3 [49]	3:69	-	-	9.33	-	-
FlowNet3-ft [49]	(1.19)	-	3.45%	(**1.79**)	-	8.60%
SENSE	2.55	-	-	6.23	23.29%	-
SENSE-ft	(**1.14**)	**1.5**	**3.00**%	(2.01)	(**9.20**%)	8.38%
SENSE-ft+semi	(1.18)	1.5	3.03%	(2.05)	(9.69%)	**8.16**%

Training semantic segmentation. We jointly train all parts of the entire network, including pre-trained encoder and decoders for optical flow and disparity, as well as a randomly initialized segmentation decoder. We empirically found using a randomly initialized segmentation decoder yields better performance.

For the segmentation distillation loss and semantic consistency loss computation, we first train the teacher segmentation model. We use the ResNet101-UPerNet [154] pre-trained on CityScapes [19] using its training set with fine annotations only, which achieves 75.4% IoU on the validation set. We fine-tune the model on KITTI 2015 [2], where the segmentation annotations, consistent with CityScapes' annotation style, for the left images are provided.

Table 4.3: Results on KITTI stereo datasets (test set).

Methods	KITTI 2012		KITTI 2015				Time
	All	Non-Occ	All		Non-Occ		
	Out-All	Out-Noc	D1-fg	D1-all	D1-fg	D1-all	(s)
Content-CNN [84]	3.07	4.29	8.58	4.54	7.44	4.00	1.0
DispNetC [90]	-	-	4.41	4.34	3.72	4.05	**0.06**
MC-CNN [163]	2.43	3.63	8.88	3.89	7.64	3.33	67
PBCP [119]	2.36	3.45	8.74	3.61	7.71	3.17	68
Displets v2 [36]	2.37	3.09	5.56	3.43	4.95	3.09	265
GC-Net [65]	1.77	2.30	6.16	2.87	5.58	2.61	0.9
PSMNet [18]	**1.49**	**1.89**	4.62	2.32	4.31	2.14	0.41
SegStereo [159]	1.68	2.03	3.70	**2.08**	4.07	2.25	0.6
FlowNet3 [49]	1.82	-	-	2.19	-	-	0.07
SENSE	1.77	2.18	3.13	2.33	2.79	2.13	**0.06**
SENSE+semi	1.73	2.16	**3.01**	2.22	**2.76**	**2.05**	**0.06**

4.4.2 Main Results

Optical flow results. Table 4.1 shows the results for optical flow estimation on the MPI Sintel benchmark dataset. Our approach outperforms CNN-based approaches without or with fine-tuning. On the more photorealistic (final) pass of the test set, which involves more rendering details such as lighting change, shadow, motion blur, etc, our approach outperforms both CNN-based and traditional hand-designed approaches by a large margin.

Table 4.2 shows the results on both KITTI2012 and KITTI2015. Our approach significantly outperforms both hand-designed and CNN-based approaches on KITTI 2012 with and without fine-tuning. On KITTI 2015, our model achieves much lower error rates than CNN-based approaches without pre-training (including ours). After fine-tuning, it outperforms all other approaches.

We note that better optical flow results are reported in an improved version of PWC-Net [132], which uses FlyingChairs followed by FlyingThings3D for pre-training. It also uses much longer learning rate schedules for fine-tuning, so the results are not directly comparable to ours.

Table 4.4: Results on KITTI2015 Scene flow dataset. CNN-based approaches need to deal with refinement of D2, where N and R indicates network and rigidity-based refinement, respectively.

Method	D1-all	D2-all	Fl-all	SF-all	D2 ref.	Time (s)
ISF [8]	4.46	5.95	6.22	8.08	-	600
CSF [85]	5.98	10.06	12.96	15.71	-	80
SGM+FF[117]	13.37	27.80	22.82	33.57	-	29
SceneFF[118]	6.57	10.69	12.88	15.78	-	65
FlowNet3 [49]	2.16	6.45	8.60	11.34	N	0.25
SENSE	2.23	7.37	8.38	11.71	N	0.16
SENSE+semi	2.22	6.57	8.16	11.35	N	0.16
SENSE+semi	2.22	5.89	7.64	9.55	R+N	0.32

Disparity results. For disparity estimation, SENSE significantly outperforms previous CNN-based approaches including DispNetC [90] and GC-Net [65] and achieves comparable accuracy with state-of-the-art approaches like PSMNet [18], SegStereo [159], and FlowNet3 [49]. Notably, our approach performs the best on the foreground region in both all and non-occluded regions on KITTI2015.

Scene flow results. Table 4.4 shows Scene flow results on KITTI 2015. SENSE performs the best in general CNN-based scene flow methods, compared to FlowNet3 [49]. Compared to ISF [8], SENSE is 2K times faster and can handle general nonrigid scene motions.

To remove artifacts introduced by the second frame disparity warping operation, we use a refinement network of a encoder-decoder structure with skip connections. It takes $\mathbf{I}^{1,l}$, $\mathbf{O}_F^{1,l}$, $\mathbf{D}^{1,l}$, and $g(\mathbf{D}^{2,l}, \mathbf{F}^{1,l})$ to generate a residual that is added to the warped disparity. From our holistic outputs, we can refine the background scene flow using a rigidity refinement step. We first determine the static rigid areas according to semantic segmentation outputs. We then calculate the ego-motion flow by minimizing the geometry consistency between optical flow and disparity images using the Gauss-Newton algorithm. Finally, we compute the warped scene flow using the disparity of the reference frame and the ego-motion to substitute the raw scene flow only in

Table 4.5: Effectiveness of different tasks.

Tasks			Results		
flow	disp	seg	flow (F1-occ) ↓	disp (D1-occ) ↓	seg (mIoU) ↑
✓			11.37%	-	-
	✓		-	2.73%	-
		✓	-	-	47.51%
✓	✓		11.59%	2.61%	-
✓		✓	11.39%	-	49.54%
	✓	✓	-	2.62%	49.12%
✓	✓	✓	11.19%	2.59%	48.25%

the rigid background region. This step additionally produces camera motion and better scene flow with minimal costs. Details of refinement steps are provided in supplementary material.

Running time. SENSE is an efficient model. SENSE takes 0.03s to compute optical flow between two images of size 436×1024. For disparity, SENSE is an order of magnitude faster than PSMNet and SegStereo, and slightly faster than FlowNet3. For scene flow using KITTI images, SENSE takes 0.15s to generate one optical flow and two disparity maps. The additional warping refinement network takes 0.01s and the rigidity refinement takes 0.15s.

Model size and memory. SENSE is small in size. It has only 8.8M parameters for the optical flow model, and 8.3M for the disparity model. The scene flow model with shared encoder has 13.4M parameters. In contrast, FlowNet3 has a flow model (117M) and a disparity model (117M), which is 20 times larger. SENSE also has a low GPU memory footprint. FlowNet3 costs 7.4GB while SENSE needs 1.5GB RAM only. Although PSMNet has fewer parameters (5.1M), it costs 4.2GB memory due to 3D convolutions.

4.4.3 Ablation Studies

Performance of different tasks. We report results of different tasks using different combinations of encoder and decoders. Our models are trained using 160 images of

Table 4.6: Ablation study of different loss terms.

Distillation		Self-supervised			Flow	Disp	Seg
seg.	occ.	sem.	pho.	ss	F1-Occ↓	D1-Occ↓	mIoU↑
					11.16%	2.52%	-
✓					10.96%	2.44%	51.48%
	✓				11.07%	2.38%	-
✓	✓				11.17%	2.33%	51.26%
		✓			11.11%	2.38%	-
			✓		11.04%	2.55%	-
				✓	11.16%	2.47%	-
		✓	✓	✓	11.21%	2.58%	-
✓	✓	✓	✓	✓	11.12%	2.49%	50.92%

KITTI 2015 with a half of the aforementioned learning rate schedule. Results are reported on the rest 40 images in Table 4.5. We can see that the shared encoder model performs better than models trained separately.

Semi-supervised loss. To study the effects of distillation and self-supervised loss terms, we perform ablation studies using all images of KITTI 2012 and 160 images of KITTI 2015 for training with a half of full learning rate schedule. The rest 40 ones of KITTI 2015 are used for testing. We finetune the baseline model using sparse flow and disparity annotations only. Table 4.6 shows the quantitative comparisons and Fig. 4.4 highlights the effects qualitatively.

Regarding distillation loss, both segmentation and occlusion distillation loss terms are useful for disparity and optical flow estimation. However, distillation loss is not helpful for reducing the artifacts in sky regions. Thus, the self-supervised loss is essential, as shown in Fig. 4.4, though quantitatively self-supervised loss is not as effective as the distillation loss. Finally, combining all loss terms yields the best optical flow and disparity accuracies. We also test SENSE trained using semi-supervised loss on KITTI, as summarized in Tables 4.2, 4.3, and 4.4. We can see it improves disparity and optical flow accuracy on KITTI 2015 and also leads to better disparity on KITTI 2012.

4.5 Conclusion

We have presented a compact network for four closely-related tasks in holistic scene understanding: Sharing an encoder among these tasks not only makes the network compact but also improves performance by exploiting the interactions among these tasks. It also allows us to introduce distillation and self-supervision losses to deal with partially labeled data. Our holistic network has similar accuracy and running time as specialized networks for optical flow. It performs favorably against state-of-the-art disparity and scene flow methods while being much faster and memory efficient. Our work shows the benefits of synergizing closely-related tasks for holistic scene understanding and we hope the insights will aid new research in this direction.

CHAPTER 5

SUPER SLOMO: HIGH QUALITY ESTIMATION OF MULTIPLE INTERMEDIATE FRAMES FOR VIDEO INTERPOLATION

5.1 Overview

There are many memorable moments in your life that you might want to record with a camera in *slow-motion* because they are hard to see clearly with your eyes: the first time a baby walks, a difficult skateboard trick, a dog catching a ball, *etc.* While it is possible to take 240-fps (frame-per-second) videos with a cell phone, professional high-speed cameras are still required for higher frame rates. In addition, many of the moments we would like to slow down are unpredictable, and as a result, are recorded at standard frame rates. Recording everything at high frame rates is impractical–it requires large memories and is power-intensive for mobile devices.

Thus it is of great interest to generate high-quality slow-motion video from existing videos. In addition to transforming standard videos to higher frame rates, video interpolation can be used to generate smooth view transitions. It also has intriguing new applications in self-supervised learning, serving as a supervisory signal to learn optical flow from unlabeled videos [79, 80].

It is challenging to generate multiple intermediate video frames because the frames have to be coherent, both spatially and temporally. For instance, generating 240-fps videos from standard sequences (30-fps) requires interpolating seven intermediate frames for every two consecutive frames. A successful solution has to not only correctly interpret the motion between two input images (implicitly or explicitly), but

also understand occlusions. Otherwise, it may result in severe artifacts in the interpolated frames, especially around motion boundaries.

Existing methods mainly focus on *single-frame* video interpolation and have achieved impressive performance for this problem setup [79, 80, 96, 97]. However, these methods cannot be directly used to generate arbitrary higher frame-rate videos. While it is an appealing idea to apply a single-frame video interpolation method recursively to generate multiple intermediate frames, this approach has at least two limitations. First, recursive single-frame interpolation cannot be fully parallelized, and is therefore slow, since some frames cannot be computed until other frames are finished (e.g., in seven-frame interpolation, frame 2 depends on 0 and 4, while frame 4 depends on 0 and 8). Errors also accumulates during recursive interpolation. Second, it can only generate $2^i - 1$ intermediate frames (*e.g.*, 3, 7). As a result, one cannot use this approach (efficiently) to generate 1008-fps video from 24-fps, which requires generating 41 intermediate frames.

In this chapter we present a high-quality *variable-length multi-frame* interpolation method that can interpolate a frame at any arbitrary time step between two frames. Our main idea is to warp the input two images to the specific time step and then adaptively fuse the two warped images to generate the intermediate image, where the motion interpretation and occlusion reasoning are modeled in a single end-to-end trainable network. Specifically, we first use a *flow computation* CNN to estimate the bi-directional optical flow between the two input images, which is then linearly fused to approximate the required intermediate optical flow in order to warp input images. This approximation works well in smooth regions but poorly around motion boundaries. We therefore use another *flow interpolation* CNN to refine the flow approximations and predict soft visibility maps. By applying the visibility maps to the warped images before fusion, we exclude the contribution of occluded pixels to the interpolated intermediate frame, reducing artifacts. The parameters of both our

flow computation and interpolation networks are independent of the specific time step being interpolated, which is an input to the flow interpolation network. Thus, our approach can generate as many intermediate frames as needed in parallel,

To train our network, we collect 240-fps videos from YouTube and hand-held cameras [127]. In total, we have 1.1K video clips, consisting of 300K individual video frames with a typical resolution of 1080×720. We then evaluate our trained model on several other independent datasets that require different numbers of interpolations, including the Middlebury [6], UCF101 [123], slowflow dataset [54], and high-frame-rate MPI Sintel [54]. Experimental results demonstrate that our approach significantly outperforms existing methods on all datasets. We also evaluate our unsupervised (self-supervised) optical flow results on the KITTI 2012 optical flow benchmark [32] and obtain better results than the recent method [79].

5.2 Related Work

Video interpolation. The classical approach to video interpolation is based on optical flow [38, 7], and interpolation accuracy is often used to evaluate optical flow algorithms [6, 133]. Such approaches can generate intermediate frames at arbitrary times between two input frames. Our experiments show that state-of-the-art optical flow method [48], coupled with occlusion reasoning [6], can serve as a strong baseline for frame interpolation. However, motion boundaries and severe occlusions are still challenging to existing flow methods [17, 32], and thus the interpolated frames tend to have artifacts around boundaries of moving objects. Furthermore, the intermediate flow computation (*i.e.*, flow interpolation) and occlusion reasoning are based on heuristics and not end-to-end trainable.

Mahajan *et al.* [88] move the image gradients to a given time step and solve a Poisson equation to reconstruct the interpolated frame. This method can also generate multiple intermediate frames, but is computationally expensive because of

the complex optimization problems. Meyer *et al.* [93] propose propagating phase information across oriented multi-scale pyramid levels for video interpolation. While achieving impressive performance, this method still tends to fail for high-frequency contents with large motions.

The success of deep learning in high-level vision tasks has inspired numerous deep models for low-level vision tasks, including frame interpolation. Long *et al.* [80] use frame interpolation as a supervision signal to learn CNN models for optical flow. However, their main target is optical flow and the interpolated frames tend to be blurry. Niklaus *et al.* [96] consider the frame interpolation as a local convolution over the two input frames and use a CNN to learn a spatially-adaptive convolution kernel for each pixel. Their method obtains high-quality results. However, it is both computationally expensive and memory intensive to predict a kernel for every pixel. Niklaus *et al.* [97] improve the efficiency by predicting separable kernels. But the motion that can be handled is limited by the kernel size (up to 51 pixels). Liu *et al.* [79] develop a CNN model for frame interpolation that has an explicit sub-network for motion estimation. Their method obtains not only good interpolation results but also promising unsupervised flow estimation results on KITTI 2012. However, as discussed previously, these CNN-based single-frame interpolation methods [96, 97, 79] are not well-suited for multi-frame interpolation.

Wang *et al.* [146] investigate to generate intermediate frames for a light field video using video frames taken from another standard camera as references. In contrast, our method aims at producing intermediate frames for a plain video and does not need reference images.

Learning optical flow. State-of-the-art optical flow methods [153, 155] adopt the variational approach introduce by Horn and Schunck [41]. Feature matching is often adopted to deal with small and fast-moving objects [14, 109]. However, this approach

requires the optimization of a complex objective function and is often computationally expensive. Learning is often limited to a few parameters [74, 114, 128].

Recently, CNN-based models are becoming increasingly popular for learning optical flow between input images. Dosovitskiy *et al.*[26] develop two network architectures, FlowNetS and FlowNetC, and show the feasibility of learning the mapping from two input images to optical flow using CNN models. Ilg *et al.* [48] further use the FlowNetS and FlowNetC as building blocks to design a larger network, FlowNet2, to achieve much better performance. Two recent methods have also been proposed [104, 131] to build the classical principles of optical flow into the network architecture, achieving comparable or even better results and requiring less computation than FlowNet2 [48].

In addition to the supervised setting, learning optical flow using CNNs in an unsupervised way has also been explored. The main idea is to use the predicted flow to warp one of the input images to another. The reconstruction error serves as a supervision signal to train the network. Instead of merely considering two frames [162], a memory module is proposed to keep the temporal information of a video sequence [102]. Similar to our work, Liang *et al.* [75] train optical flow via video frame extrapolation, but their training uses the flow estimated by the EpicFlow method [109] as an additional supervision signal.

5.3 Proposed Approach

In this section, we first introduce optical flow-based intermediate frame synthesis in section 5.3.1. We then explain details of our flow computation and flow interpolation networks in section 5.3.2. In section 5.3.3, we define the loss function used to train our networks.

5.3.1 Intermediate Frame Synthesis

Given two input images I_0 and I_1 and a time $t \in (0, 1)$, our goal is to predict the intermediate image $\hat{I}_t$ at time $T = t$. A straightforward way is to accomplish this is to train a neural network [80] to directly output the RGB pixels of $\hat{I}_t$. In order to do this, however, the network has to learn to interpret not only the motion pattens but also the appearance of the two input images. Due to the rich RGB color space, it is hard to generate high-quality intermediate images in this way. Inspired by [6] and recent advances in single intermediate video frame interpolation [96, 97, 79], we propose fusing the warped input images at time $T = t$.

Let $F_{t\to0}$ and $F_{t\to1}$ denote the optical flow from I_t to I_0 and I_t to I_1, respectively. If these two flow fields are known, we can synthesize the intermediate image $\hat{I}_t$ as follows:

$$\hat{I}_t = \alpha_0 \odot g(I_0, F_{t\to0}) + (1 - \alpha_0) \odot g(I_1, F_{t\to1}), \tag{5.1}$$

where $g(\cdot, \cdot)$ is a *backward warping* function, which can be implemented using bilinear interpolation [170, 79] and is differentiable. The parameter α_0 controls the contribution of the two input images and depend on two factors: temporal consistency and occlusion reasoning. $\odot$ denotes element-wise multiplication, implying content-aware weighting of input images. For temporal consistency, the closer the time step $T = t$ is to $T = 0$, the more contribution I_0 makes to $\hat{I}_t$; a similar property holds for I_1. On the other hand, an important property of the video frame interpolation problem is that if a pixel p is visible at $T = t$, it is most likely *at least visible in one of the input images*,[1] which means the occlusion problem can be addressed. We therefore introduce *visibility maps* $V_{t\leftarrow0}$ and $V_{t\leftarrow1}$. $V_{t\leftarrow0}(p) \in [0, 1]$ denotes whether the pixel p

[1]It is a rare case but it may happen that an object appears and disappears between I_0 and I_1.

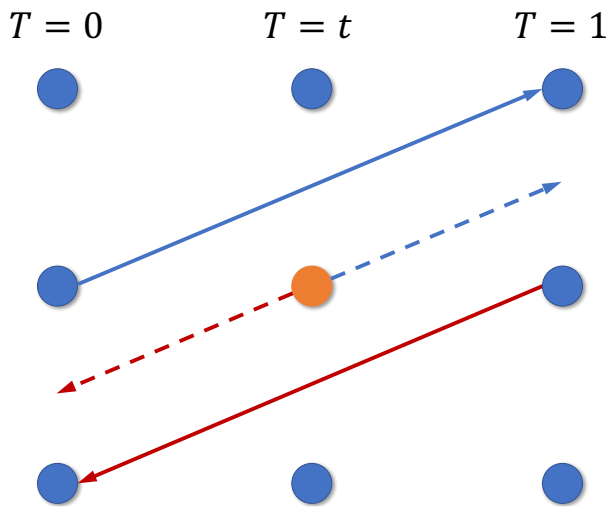


Figure 5.1: Illustration of intermediate optical flow approximation. The orange pixel borrows optical flow from pixels at the same position in the first and second images.

remains visible (0 is fully occluded) when moving from $T = 0$ to $T = t$. Combining the temporal consistency and occlusion reasoning, we have

$$\hat{I}_t = \frac{1}{Z} \odot \big((1-t)V_{t\leftarrow 0} \odot g(I_0, F_{t\to 0}) + tV_{t\leftarrow 1} \odot g(I_1, F_{t\to 1})\big),$$

where $Z = (1-t)V_{t\to 0} + tV_{t\to 1}$ is a normalization factor.

5.3.2 Arbitrary-time Flow Interpolation

Since we have no access to the target intermediate image I_t, it is hard to compute the flow fields $F_{t\to 0}$ and $F_{t\to 1}$. To address this issue, we can approximately synthesize the intermediate optical flow $F_{t\to 0}$ and $F_{t\to 1}$ using the optical flow between the two input images $F_{0\to 1}$ and $F_{1\to 0}$.

Consider the toy example shown in Fig. 5.1, where each column corresponds to a certain time step and each dot represents a pixel. For the orange dot p at $T = t$, we are interested in synthesizing its optical flow to its corresponding pixel at $T = 1$ (the blue dashed arrow). One simple way is to borrow the optical flow *from the same grid*

positions at $T = 0$ and $T = 1$ (blue and red solid arrows), assuming that the optical flow field is locally smooth. Specifically, $F_{t\to1}(p)$ can be approximated as

$$\hat{F}_{t\to1}(p) = (1-t)F_{0\to1}(p) \tag{5.2}$$

or

$$\hat{F}_{t\to1}(p) = -(1-t)F_{1\to0}(p), \tag{5.3}$$

where we take the direction of the optical flow between the two input images in the same or opposite directions and scale the magnitude accordingly ($(1-t)$ in (5.3)). Similar to the temporal consistency for RGB image synthesis, we can approximate the intermediate optical flow by combining the bi-directional input optical flow as follows (in vector form).

$$\begin{aligned}\hat{F}_{t\to0} &= -(1-t)tF_{0\to1} + t^2F_{1\to0}\\ \hat{F}_{t\to1} &= (1-t)^2F_{0\to1} - t(1-t)F_{1\to0}.\end{aligned} \tag{5.4}$$

This approximation works well in smooth regions but poorly around motion boundaries, because the motion near motion boundaries is not locally smooth. To reduce artifacts around motion boundaries, which may cause poor image synthesis, we propose learning to refine the initial approximation. Inspired by the cascaded architecture for optical flow estimation in [48], we train a flow interpolation sub-network. This sub-network takes the input images I_0 and I_1, the optical flows between them $F_{0\to1}$ and $F_{0\to1}$, the flow approximations $\hat{F}_{t\to0}$ and $\hat{F}_{0\to1}$, and two warped input images using the approximated flows $g(I_0, \hat{F}_{t\to0})$ and $g(I_1, \hat{F}_{t\to1})$ as input, and outputs refined intermediate optical flow fields $F_{t\to1}$ and $F_{t\to0}$. Sample interpolation results are displayed in Figure 5.2.

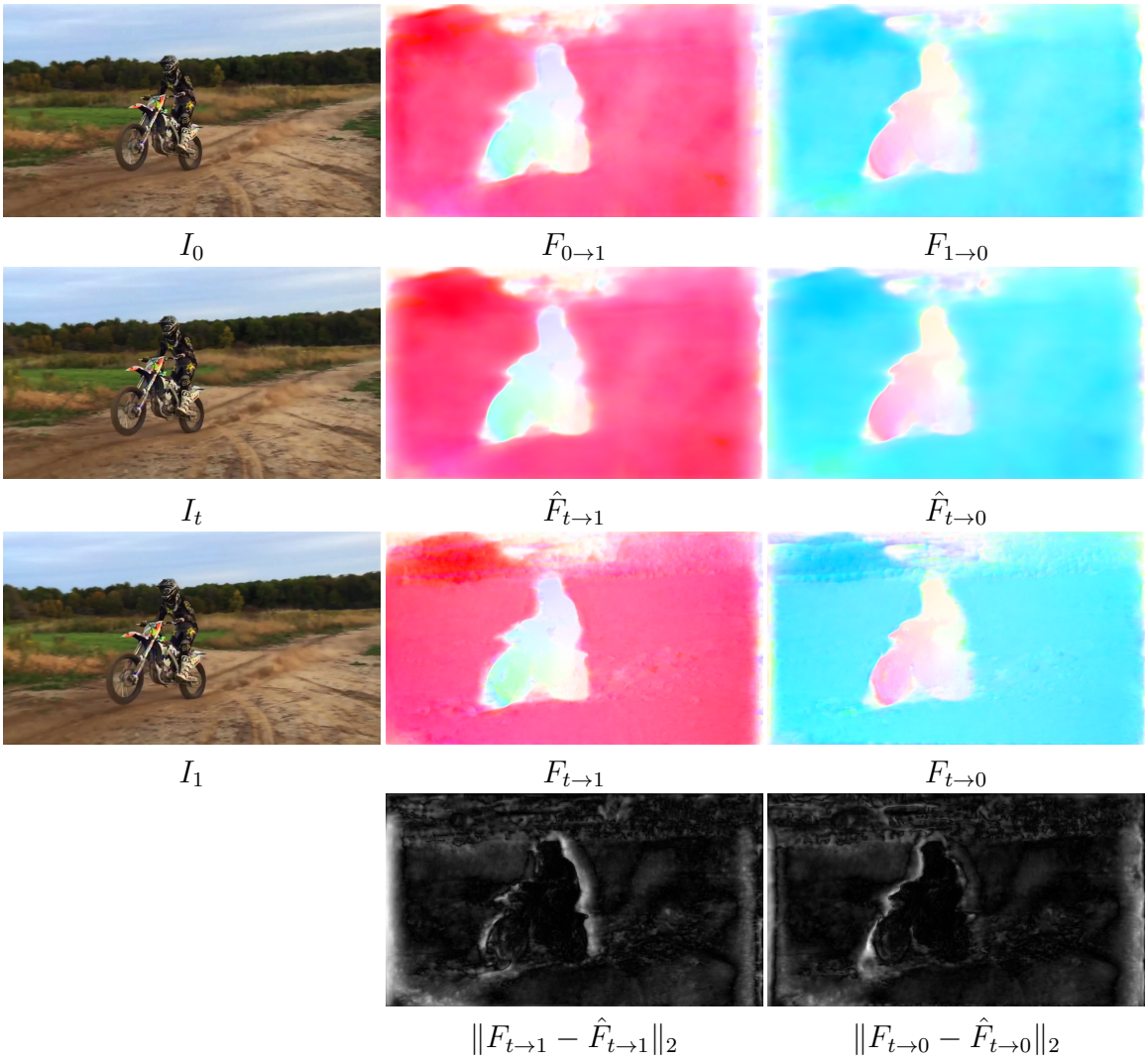


Figure 5.2: Samples of flow interpolation results, where $t = 0.5$. The entire scene is moving toward the left (due to camera translation) and the motorcyclist is independently moving left. The last row shows that the refinement from our flow interpolation CNN is mainly around the motion boundaries (the whiter a pixel, the bigger the refinement).

As discussed in Section 5.3.1, visibility maps are essential to handle occlusions. Thus, We also predict two visibility maps $V_{t\leftarrow0}$ and $V_{t\leftarrow1}$ using the flow interpolation CNN, and enforce them to satisfy the following constraint

$$V_{t\leftarrow0} = 1 - V_{t\leftarrow1}. \tag{5.5}$$

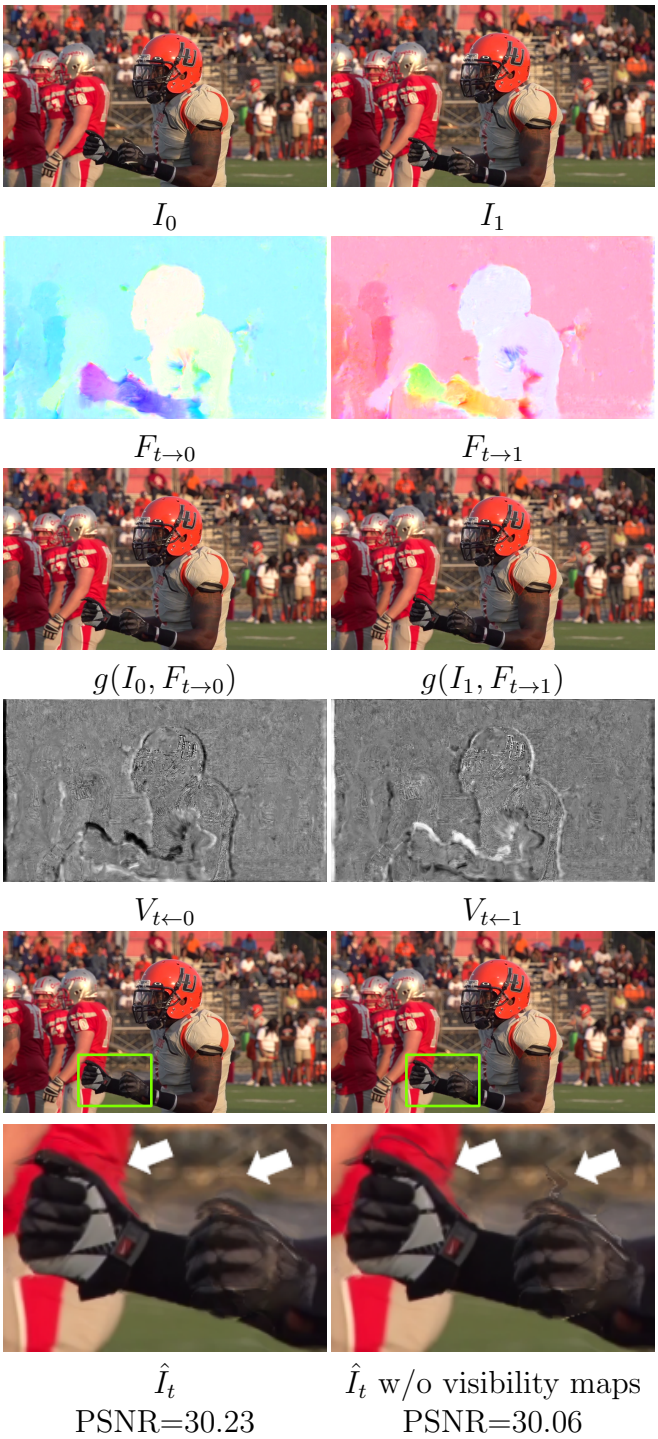


Figure 5.3: Samples of predicted visibility maps (best viewed in color), where $t=0.5$. The arms move downwards from $T=0$ to $T=1$. So the area right above the arm at $T=0$ is visible at t but the area right above the arm at $T=1$ is occluded (*i.e.*, invisible) at t. The visibility maps in the fourth row clearly show this phenomenon. The white area around arms in $V_{t\leftarrow 0}$ indicate such pixels in I_0 contribute most to the synthesized $\hat{I}_t$ while the occluded pixels in I_1 have little contribution. Similar phenomena also happen around motion boundaries (*e.g.*, around bodies of the athletes).

Without such a constraint, the network training diverges. Intuitively, $V_{t\leftarrow 0}(p) = 0$ implies $V_{t\leftarrow 1}(p) = 1$, meaning that the pixel p from I_0 is occluded at $T = t$, we should fully trust I_1 and vice versa. Note that it rarely happens that a pixel at time t is occluded both at time 0 and 1. Since we use soft visibility maps, when the pixel p is visible both in I_0 and I_1, the network learns to adaptively combine the information from two images, similarly to the matting effect [111]. Samples of learned visibility maps are shown in the fourth row of Fig. 5.3.

In order to do flow interpolation, we need to first compute the bi-directional optical flow between the two input images. Recent advances in deep learning for optical flow have demonstrated great potential to leverage deep CNNs to reliably estimate optical flow. In this chapter, we train a flow computation CNN, taking two input images I_0 and I_1, to jointly predict the forward optical flow $F_{0\rightarrow 1}$ and backward optical flow $F_{1\rightarrow 0}$ between them.

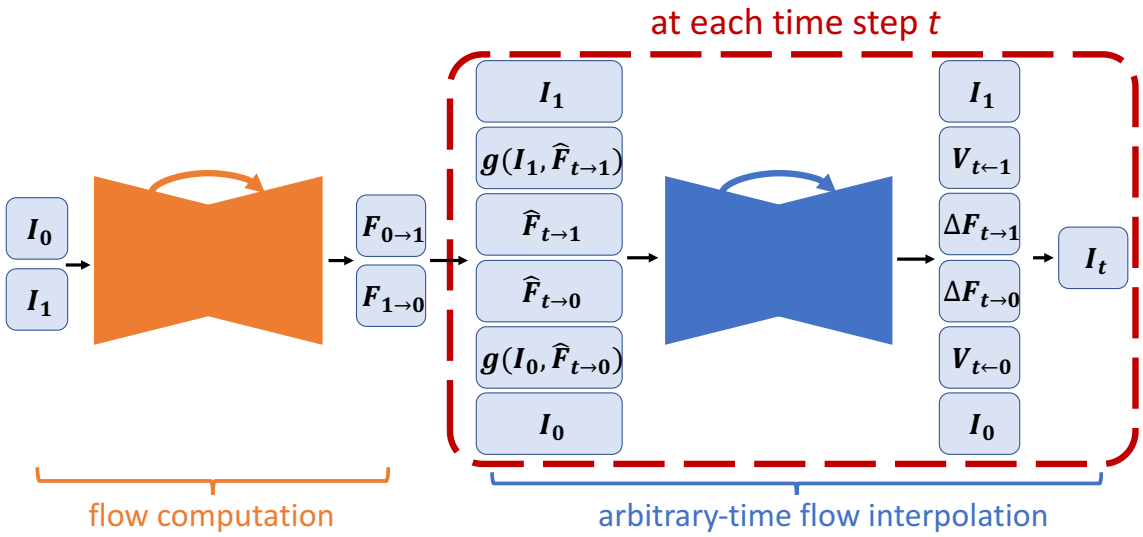


Figure 5.4: Network architecture of our approach.

Our entire network is summarized in Fig. 5.4. For the flow computation and flow interpolation CNNs, we adopt the U-Net architecture [112]. The U-Net is a fully convolutional neural network, consisting of an encoder and a decoder, with skip connections between the encoder and decoder features at the same spatial resolution For

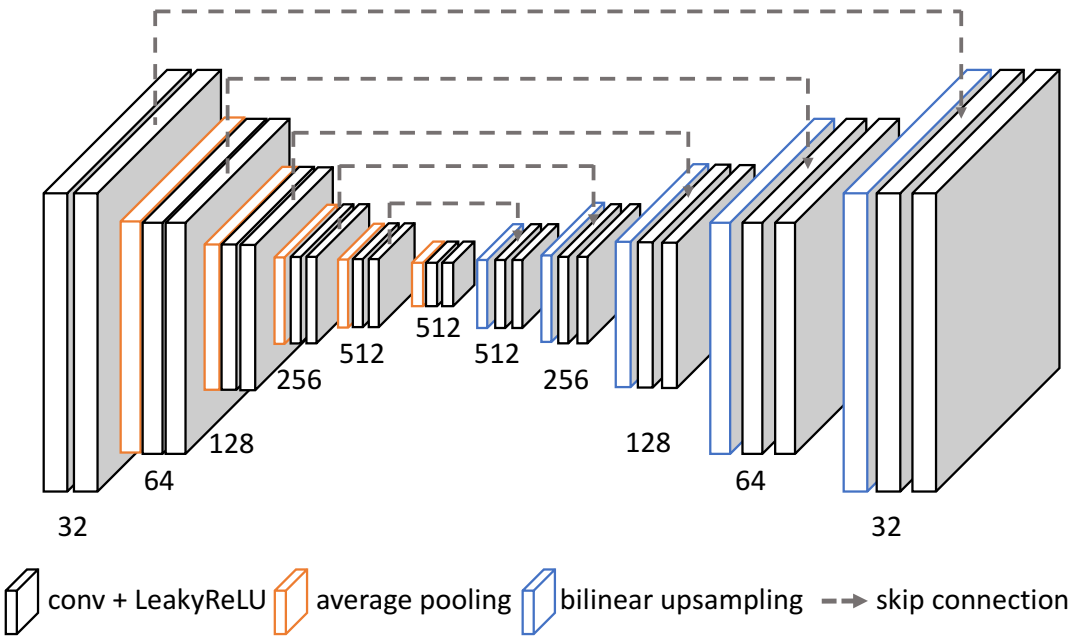


Figure 5.5: Illustration of the architecture of our flow computation and flow interpolation CNNs.

both networks, we have 6 hierarchies in the encoder, consisting of two convolutional and one Leaky ReLU ($\alpha = 0.1$) layers. At the end of each hierarchy except the last one, an average pooling layer with a stride of 2 is used to decrease the spatial dimension. There are 5 hierarchies in the decoder part. At the beginning of each hierarchy, a bilinear upsampling layer is used to increase the spatial dimension by a factor of 2, followed by two convolutional and Leaky ReLU layers.

For the flow computation CNN, it is crucial to have large filters in the first few layers of the encoder to capture long-range motion. We therefore use 7×7 kernels in the first two convoluional layers and 5×5 in the second hierarchy. For layers in the rest of entire network, we use 3×3 convolutional kernels. The detailed configuration of the network is described in Fig. 5.5.

We found concatenating output of the encoders in two networks together as input to the decoder of the flow interpolation network yields slightly better results.

Moreover, instead of directly predicting the intermediate optical flow in the flow interpolation network, we found it performs slightly better to predict intermediate optical flow residuals. In specific, the flow interpolation network predicts $\Delta F_{t\to 0}$ and $\Delta F_{t\to 1}$. We then have

$$\begin{aligned} F_{t\to 0} &= \hat{F}_{t\to 0} + \Delta F_{t\to 0} \\ F_{t\to 1} &= \hat{F}_{t\to 1} + \Delta F_{t\to 1} \end{aligned} \tag{5.6}$$

5.3.3 Training

Given input images I_0 and I_1, a set of intermediate frames $\{I_{t_i}\}_{i=1}^N$ between them, where $t_i \in (0,1)$, and our predictions of intermediate frames $\{\hat{I}_{t_i}\}_{i=1}^N$, our loss function is a linear combination of four terms:

$$l = \lambda_r l_r + \lambda_p l_p + \lambda_w l_w + \lambda_s l_s. \tag{5.7}$$

Reconstruction loss l_r models how good the reconstruction of the intermediate frames is:

$$l_r = \frac{1}{N}\sum_{i=1}^{N} \|\hat{I}_{t_i} - I_{t_i}\|_1. \tag{5.8}$$

Such a reconstruction loss is defined in the RGB space, where pixel values are in the range $[0, 255]$.

Perceptual loss. Even though we use the L_1 loss to model the reconstruction error of intermediate frames, it might still cause blur in the predictions. We therefore use a perceptual loss [64] to preserve details of the predictions and make interpolated frames sharper, similar to [97]. Specifically, the perceptual loss l_p is defined as

$$l_p = \frac{1}{N}\sum_{i=1}^{N} \|\phi(\hat{I}_t) - \phi(I_t)\|_2, \tag{5.9}$$

where ϕ denote the `conv4_3` features of an ImageNet pre-trained VGG16 model [122]

Warping loss. Besides intermediate predictions, we also introduce the warping loss l_w to model the quality of the computed optical flow, defined as

$$l_w = \|I_0 - g(I_1, F_{0\to1})\|_1 + \|I_1 - g(I_0, F_{1\to0})\|_1 + \frac{1}{N}\sum_{i=1}^{N}\|I_{t_i} - g(I_0, \hat{F}_{t_i\to0})\|_1 + \frac{1}{N}\sum_{i=1}^{N}\|I_{t_i} - g(I_1, \hat{F}_{t_i\to1})\|_1. \tag{5.10}$$

Smoothness loss. Finally, we add a smoothness term [79] to encourage neighboring pixels to have similar flow values:

$$l_s = \|\nabla F_{0\to1}\|_1 + \|\nabla F_{1\to0}\|_1. \tag{5.11}$$

The weights have been set empirically using a validation set as $\lambda_r = 0.8$, $\lambda_p = 0.005$, $\lambda_w = 0.4$, and $\lambda_s = 1$. Every component of our network is differentiable, including warping and flow approximation. Thus our model can be end-to-end trained.

Table 5.1: Statistics of dataset we use to train our network.

	Adobe240-fps [127]	YouTube240-fps
#video clips	118	1,014
#video frames	79,768	296,352
mean #frames per clip	670.3	293.1
resolution	720p	720p

5.4 Experiments

5.4.1 Dataset

To train our network, we use the 240-fps videos from [127], taken with hand-held cameras. We also collect a dataset of 240-fps videos from YouTube. Table 5.1 summarizes the statistics of the two datasets and Fig. 5.6 shows a snapshot of randomly

Adobe240-fps

YouTube240-fps

Figure 5.6: Snapshot of our training data.

sampled video frames. In total, we have 1,132 video clips and 376K individual video frames. There are a great variety of scenes in both datasets, from indoor to outdoor, from static to moving cameras, from daily activities to professional sports, etc.

We train our network using all of our data and test our model on several independent datasets, including the Middlebury benchmark [6], UCF101 [123], slowflow dataset [54], and high-frame-rate Sintel sequences [54]. For Middlebury, we submit our single-frame video interpolation results of eight sequences to its evaluation server. For UCF101, in every triple of frames, the first and third ones are used as input to predict the second frame using 379 sequences provided by [79]. The slowflow dataset contains 46 videos taken with professional high-speed cameras. We use the first and eighth video frames as input, and interpolate intermediate 7 frames, equivalent to

Table 5.2: Effectiveness of multi-frame video interpolation on the *Adobe240-fps* dataset.

	PSNR	SSIM	IE
1 interp	30.26	0.909	8.85
3 interp	31.02	0.917	8.43
7 interp	**31.19**	**0.918**	**8.30**

converting a 30-fps video to a 240-fps one. The original Sintel sequences [17] were rendered at 24 fps. 13 of them were re-rendered at 1008 fps [54]. To convert from 24-fps to 1008-fps using a video frame interpolation approach, one needs to insert 41 in-between frames. However, as discussed in the introduction, it is not directly possible with recursive single-frame interpolation methods [96, 97, 79] to do so. Therefore, we instead predict 31 in-between frames for fair comparisons with previous methods.

Our network is trained using the Adam optimizer [66] for 500 epochs. The learning rate is initialized to be 0.0001 and decreased by a factor of 10 every 200 epochs. During training, all video clips are first divided into shorter ones with 12 frames in each and there is no overlap between any of two clips. For data augmentation, we randomly reverse the direction of entire sequence and select 9 consecutive frames for training. On the image level, each video frame is resized to have a shorter spatial dimension of 360 and a random crop of 352×352 plus horizontal flip are performed.

For evaluation, we report Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) scores between predictions and ground-truth in-between video frames, as well as the interpolation error (IE) [6], which is defined as root-mean-squared (RMS) difference between the ground-truth image and the interpolated image.

Table 5.3: Effectiveness of different components of our model on the *Adobe240-fps* dataset.

	PSNR	SSIM	IE
w/o flow interpolation	30.34	0.908	8.93
w/o vis map	31.16	0.918	8.33
w/o perceptual loss	30.96	0.916	8.50
w/o warping loss	30.52	0.910	8.80
w/o smoothness loss	**31.19**	**0.918**	**8.26**
full model	**31.19**	**0.918**	8.30

5.4.2 Ablation Studies

In this section, we perform ablation studies to analyze our model. For the first two experiments, we randomly sampled 107 videos from Adobe240-fps dataset for training and the remaining 12 ones for testing.

Effectiveness of multi-frame video interpolation. We first test whether jointly predicting several in-between frames improves the video interpolation results. Intuitively, predicting a set of in-between frames together might implicitly enforce the network to generate temporally coherent sequences.

To this end, we train three variants of our model: predicting intermediate single, three, and seven frames, which are all evenly distributed across time steps. At test time, we use each model to predict seven in-between frames. Table 5.2 clearly demonstrates that the more intermediate frames we predict during training, the better the model is.

Impact of different components design. We also investigate the contribution of each component in our model. In particular, we study the impact of flow interpolation by removing the flow refinement from the second U-Net (but keep using the visibility maps). We further study the use of visibility maps as means of occlusion reasoning. We can observe from Table 5.3 that removing each of three components harms performance. Particularly, the flow interpolation plays a crucial role, which

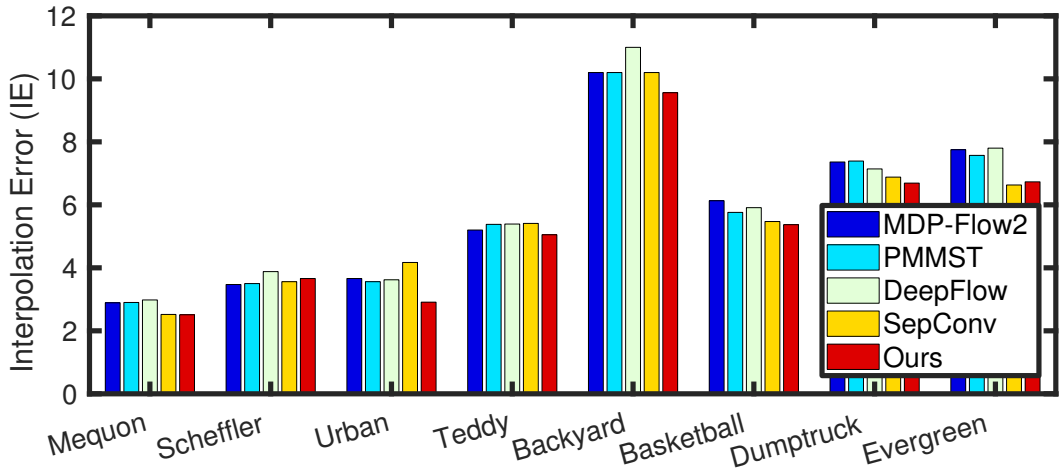


Figure 5.7: Performance comparisons on each sequence of the Middlebury dataset. Numbers are obtained from the Middlebury evaluation server.

Table 5.4: Results on the *UCF101* dataset.

	PSNR	SSIM	IE
Phase-Based [93]	32.35	0.924	8.84
FlowNet2 [6, 48]	32.30	0.930	8.40
DVF [79]	32.46	0.930	8.27
SepConv [97]	33.02	0.935	8.03
Ours (Adobe240-fps)	32.84	0.935	8.04
Ours	**33.14**	**0.938**	**7.80**

verifies our motivation to introduce the second learned network to refine intermediate optical flow approximations. Adding visibility maps improves the interpolation performance slightly. Without it, there are artifacts generated around motion boundaries, as shown in Figure 5.3. Both of these validate our hypothesis that jointly learning motion interpretation and occlusion reasoning helps video interpolation.

We also study different loss terms, where the warping loss is the most important one. Although adding the smoothness terms slightly hurts the performance quantitatively, we fount it is useful to generate visually appealing optical flow between input images.

Table 5.5: Results on the *slowflow* dataset.

	PSNR	SSIM	IE
Phase-Based [93]	31.05	0.858	8.21
FlowNet2 [6, 48]	34.06	**0.924**	**5.35**
SepConv [97]	32.69	0.893	6.79
Ours	**34.19**	**0.924**	6.14

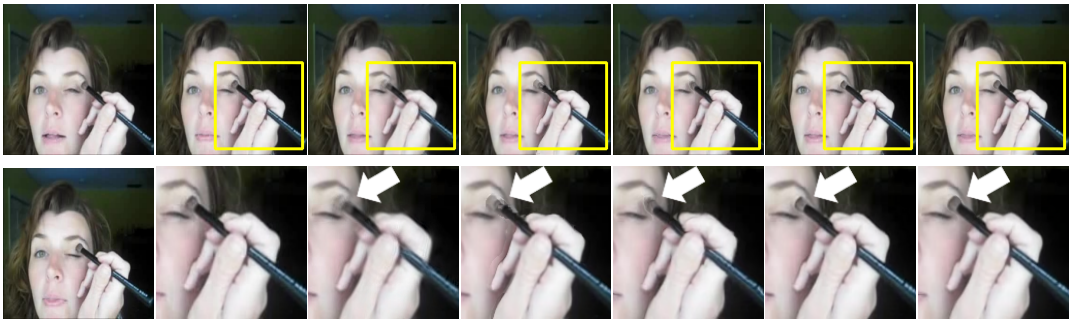
Table 5.6: Results on the high-frame-rate *Sintel* dataset.

	PSNR	SSIM	IE
Phase-Based [93]	28.67	0.840	10.24
FlowNet2 [6, 48]	30.79	0.922	5.78
SepConv [97]	31.51	0.911	6.61
Ours	**32.38**	**0.927**	**5.42**

Impact of the number of training samples. Finally, we investigate the effect of the number of training samples. We compare two models: one trained on the Adobe240-fps dataset only and the other one trained on our full dataset. The performance of these two models on the UCF101 dataset can be found in last two rows Table 5.4. We can see that our model benefits from more training data.

5.4.3 Comparison with state-of-the-art methods

In this section, we compare our approach with state-of-the-art methods including phase-based interpolation [93], separable adaptive convolution (SepConv) [97], and deep voxel flow (DVF) [79]. We also implement a baseline approach using the interpolation algorithm presented in [6], where we use FlowNet2 [48] to compute the bi-directional optical flow results between two input images. FlowNet2 is good at capturing global background motion and recovering sharp motion boundaries for the optical flow. Thus, when coupled with occlusion reasoning [6], FlowNet2 serves as a strong baseline.



(a) (b) (c) (d) (e) (f) (g)

Figure 5.8: Visual results of a sample from UCF101. Our model produces less artifacts around the brush and the hand (best viewed in color). Please see the supplementary material for more image and *video* results. From left to right: (a) two input images, (b) actual in-between, (c) PhaseBased [93], (d) FlowNet2 [6, 48], (e) DVF [79], (f) SepConv [97], and (g) ours.

Single-frame video interpolation. The interpolation error (IE) scores on each sequence form the Middlebury dataset are shown in Figure 5.7. In addition to SepConv, we also compare our model with other three top-performing models on the Middlebury dataset[2], where the interpolation algorithm [6] is coupled with different optical flow methods including MDP-Flow2 [156], PMMST [164], and DeepFlow [151]. Our model achieves the best performance on 6 out of all 8 sequences. Particularly, the `Urban` sequence is generated synthetically and the `Teddy` sequence contains actually two stereo pairs. The performance of our model validates the generalization ability of our approach.

On UCF101, we compute all metrics using the motion masks provided by [79]. The quantitative results are shown in Table 5.4, highlighting the performance of each interpolation model's capacity to deal with challenging motion regions. Our model consistently outperforms both non-neural [93] and CNN-based approaches [97, 79]. Sample interpolation results on a sample from UCF101 can be found at Figure 5.8. More re-

[2]http://vision.middlebury.edu/flow/eval/results/results-i1.php

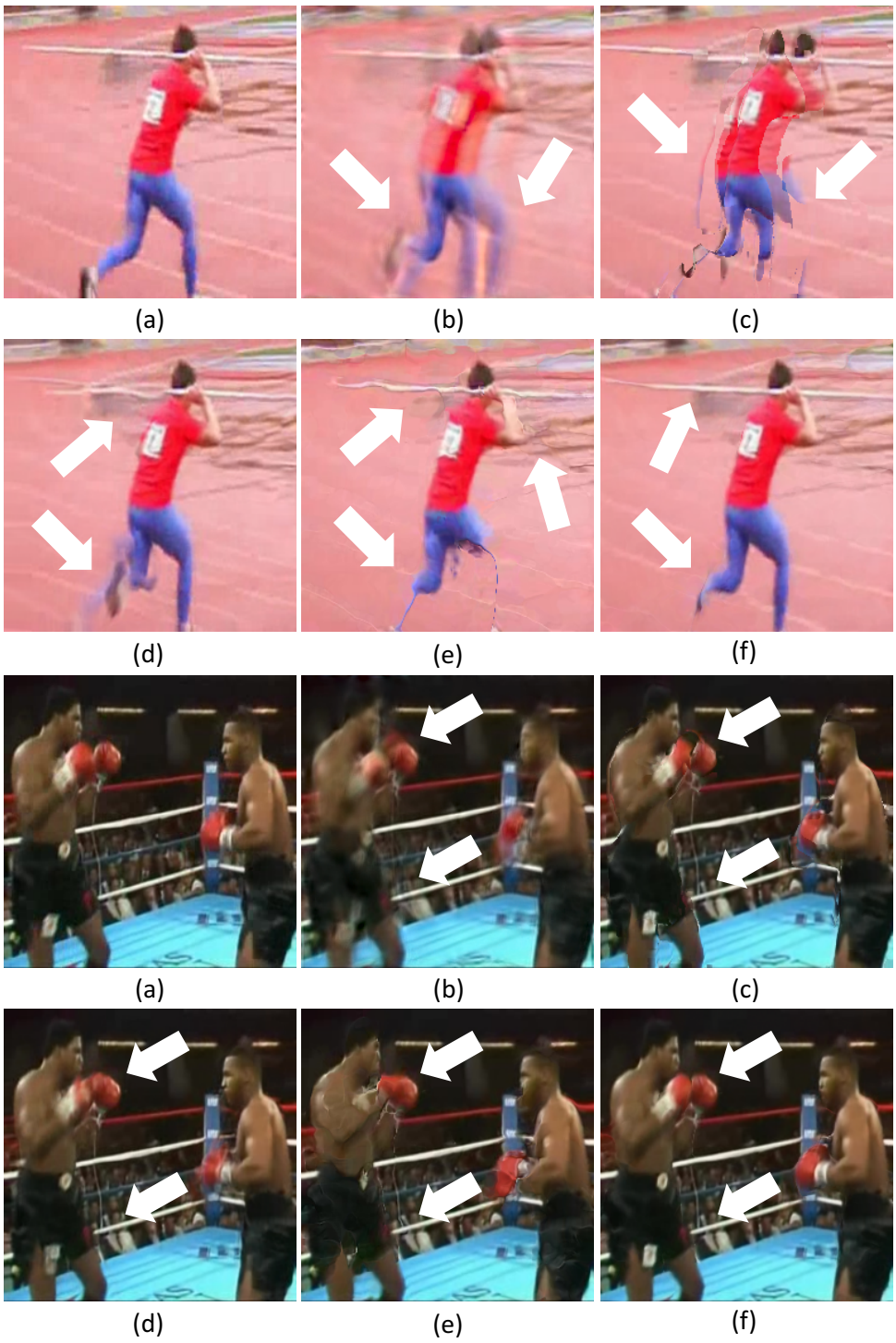


Figure 5.9: Visual comparisons on the UCF101 dataset. (a) Ground truth in-between frame, interpolation results from (b) PhaseBased [93], (c) FlowNet2 [38, 48], (d) SepConv [97], (e) DVF [79], and (f) Ours.

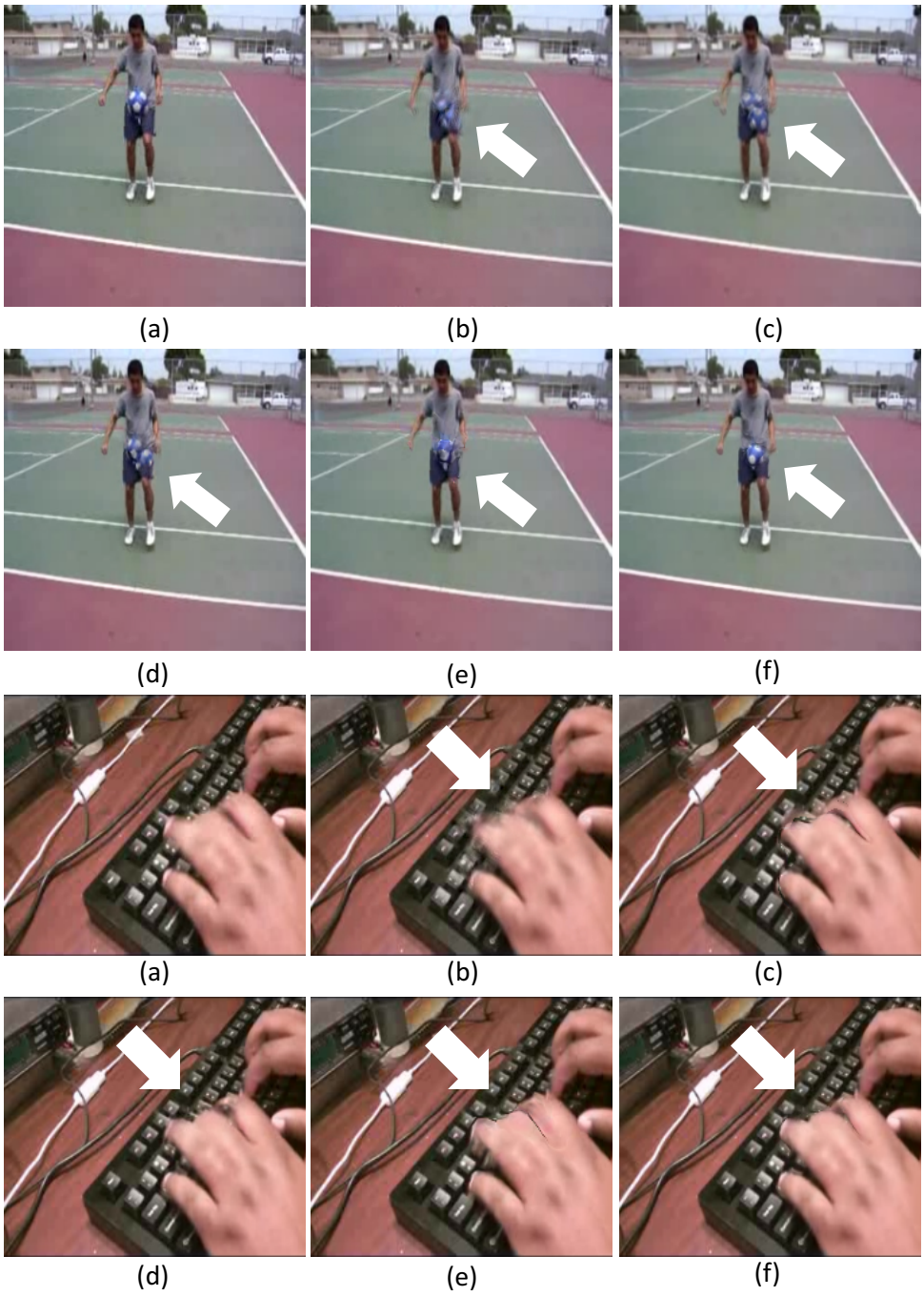


Figure 5.10: Visual comparisons on the UCF101 dataset. (a) Ground truth in-between frame, interpolation results from (b) PhaseBased [93], (c) FlowNet2 [38, 48], (d) SepConv [97], (e) DVF [79], and (f) Ours.

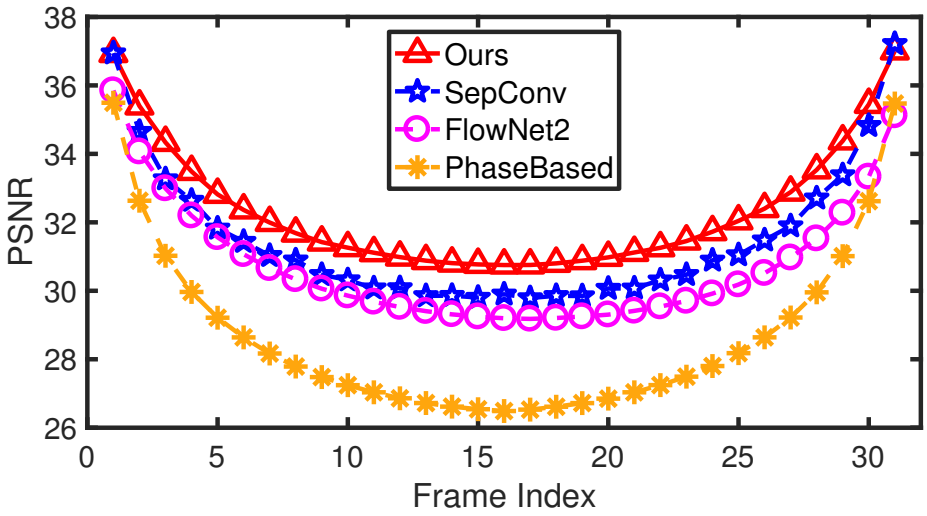


Figure 5.11: PSNR at each time step when generating 31 intermediate frames on the high-frame-rate Sintel dataset.

sults can be found in Figure 5.9 and Figure 5.10. For more visual comparisons, please refer to our supplementary video http://jianghz.me/projects/superslomo/superslomo$_public.mp4$.

Multi-frame video interpolation. For the slowflow dataset, we predict 7 in-between frames. All experiments are performed on the half-resolution images with a spatial dimension of 1280×1024. On this dataset, our approach achieves the best PSNR and SSIM scores and FlowNet2 achieves the best SSIM and L_1 error scores. FlowNet2 is good at capturing global motions and thus produces sharp prediction results on those background regions, which follow a global motion pattern. Detailed visual comparisons can be found in our supplementary material.

On the challenging high-frame-rate Sintel dataset, our approach significantly outperforms all other methods. We also show the PSNR scores at each time step in Figure 5.11. Our approach produces the best predictions for each in-between time step except slightly worse than SepConv at the last time step.

In summary, our approach achieves state-of-the-art results over all datasets, generating single or multiple intermediate frames. It is remarkable, considering the fact our model can be directly applied to different scenarios without any modification.

Table 5.7: Optical flow results on the KITTI 2012 benchmark.

	LDOF[14]	EpicFlow[109]	FlowNetS[26]	DVF[79]	Ours
EPE	12.4	3.8	9.1	14.6	13.0

5.4.4 Unsupervised Optical Flow

Our video frame interpolation approach has an unsupervised (self-supervised) network (the flow computation CNN) that can compute the bi-directional optical flow between two input images. Following [79], we evaluate our unsupervised forward optical flow results on the testing set of KITTI 2012 optical flow benchmark [32]. The average end-point error (EPE) scores of different methods are reported in Table 5.7. Compared with previous unsupervised method DVF [79], our model achieves an average EPE of 13.0, an 11% relative improvement. Very likely this improvement results from the multi-frame video interpolation setting, as DVF [79] has a similar U-Net architecture to ours.

5.5 Conclusion

We have proposed an end-to-end trainable CNN that can produce as many intermediate video frames as needed between two input images. We first use a flow computation CNN to estimate the bidirectional optical flow between the two input frames, and the two flow fields are linearly fused to approximate the intermediate optical flow fields. We then use a flow interpolation CNN to refine the approximated flow fields and predict soft visibility maps for interpolation. We use more than 1.1K 240-fps video clips to train our network to predict seven intermediate frames. Ablation studies on separate validation sets demonstrate the benefit of flow interpolation and visibility map. Our multi-frame approach consistently outperforms state-of-the-art single frame methods on the Middlebury, UCF101, slowflow, and high-frame-rate Sintel datasets. For the unsupervised learning of optical flow, our network outperforms the recent DVF method [79] on the KITTI 2012 benchmark.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, we investigate dynamic scene understanding and synthesis utilizing motion cues contained in videos. In Chapter 3, we present an approach where useful representations of the visual world can be learned from unlabeled videos by having a (virtual) agent wander around. Chapter 4 introduces a holistic approach to estimate the surrounding's motion in order for an agent to safely move in the dynamic world, where multiple tasks are solved simultaneously, leading to more accurate scene understanding. These two approaches reveal the great potential of training visual perception models from unlabeled or partially-labeled data. In particular, we believe that holistic scene understanding is a promising direction, where constraints between different tasks can be utilized to greatly reduce the reliance on manually labeled data.

Finally, in Chapter 5, we show that missing visual content can be re-created in a normal-speed video based on the understanding of motion trajectories of pixels, which enables us to perceive the visual surroundings in a novel manner. This approach is useful for other applications, such as video compression (by compressing the key frames only and later synthesizing the intermediate video frames during the decompression stage), generating videos with motion blur, synthesizing smooth vieo transitions on a head-mounted virtual reality (VR) device to provide users' more authentic experiences, etc.

6.1 Future Work

We briefly mention three future directions, which extends this thesis, toward understanding and synthesizing the dynamic visual world.

Holistic 3D Scene Understanding of Dynamics and Interaction. We live in a 3D visual world, which is constantly in motion. My goal is to build a holistic 3D scene understanding model unifying appearance, geometry, motion, and semantics. Such holistic model will not only allow us to infer various scene dynamics, including depth, ego-motion, object motion, semantic parsing, etc, but also incorporate interactions of agents in the scene. Interactions of agents are crucial for developing intelligence. An important task for kids to learn in kindergarten, for example, is how to play together with others. Investigating agents' interactions is beneficial for applications as well. In an urban traffic scene, An autonomous driving vehicle must predict intent and motion trend of other (autonomous) vehicles, pedestrians, and bicyclists/motorcyclists. Such a 3D holistic scene understanding model will have a huge impact on the advance and safety of next-generation intelligent systems.

Enhancing Photography Experiences in the Mobile Era. Taking photos has never been so easy with smartphones. Recording VLOGs (video blogs) and sharing them on social media is a fashion living style nowadays. It imposes great challenges as well as opportunities for researchers to improve users' photography experiences on cell phones. On the one hand, photos and videos casually shot by non-expert users are usually less visually appealing. Photos or videos may be tilted and look too dark or blurry. On the other hand, the latest smartphones are usually equipped with powerful computational resources and cameras, which allow running of powerful visual perception models to provide instant assistant to users. Following the effort of synthesizing slow-motion videos [62], and from a broader perspective of deep generative models, I am passionate about developing algorithms that enhance users' photogra-

phy experiences to more easily record and share their daily lives, for example, via VLOGs.

Toward Understanding Pre-training of Deep CNNs. In our previsou work, [58] we demonstrated that pre-training of deep CNNs, either self-supervised or supervised, leads to better accuracy on a downstream task than a random initialization. Particularly, when the number of training samples in a downstream task is limited, such pre-training is critical. We know training of deep CNNs often leads to a reasonably good and stable local minimum. Little is known so far, however, about the dynamics of the optimization process given different initialization. I am interested in analyzing the behaviors of such optimizations and furthermore taking inspirations to design better initialization schemes. After all, pre-trained weights are essentially equivalent to a better initialization of the CNNs once the pre-training is finished. It is impactful on broad fields, including machine learning, computer vision, and natural language processing.

BIBLIOGRAPHY

[1] Agrawal, Pulkit, Carreira, João, and Malik, Jitendra. Learning to see by moving. In *CVPR* (2015).

[2] Alhaija, Hassan, Mustikovela, Siva, Mescheder, Lars, Geiger, Andreas, and Rother, Carsten. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *IJCV* (2018).

[3] Bai, Min, Luo, Wenjie, Kundu, Kaustav, and Urtasun, Raquel. Exploiting semantic information and deep matching for optical flow. In *Proc. ECCV* (2016).

[4] Bailer, Christian, Taetz, Bertram, and Stricker, Didier. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *IEEE International Conference on Computer Vision, ICCV* (2015), pp. 4015–4023.

[5] Bailer, Christian, Varanasi, Kiran, and Stricker, Didier. CNN-based patch matching for optical flow with thresholded hinge embedding loss. In *Proc. CVPR* (2017).

[6] Baker, Simon, Scharstein, Daniel, Lewis, J. P., Roth, Stefan, Black, Michael J., and Szeliski, Richard. A database and evaluation methodology for optical flow. *IJCV 92*, 1 (2011), 1–31.

[7] Barron, J.L., Fleet, D.J., and Beauchemin, S.S. Performance of optical flow techniques. *IJCV 12*, 1 (1994), 43–77.

[8] Behl, Aseem, Jafari, Omid Hosseini, Mustikovela, Siva Karthik, Alhaija, Hassan Abu, Rother, Carsten, and Geiger, Andreas. Bounding boxes, segmentations and object coordinates: How important is recognition for 3D scene flow estimation in autonomous driving scenarios? In *Proc. ICCV* (2017).

[9] Bideau, Pia, and Learned-Miller, Erik. It's moving! A probabilistic model for causal motion segmentation in moving camera videos. In *ECCV* (2016).

[10] Bideau, Pia, RoyChowdhury, Aruni, Menon, Rakesh R, and Learned-Miller, Erik. The best of both worlds: Combining cnns and geometric constraints for hierarchical motion segmentation. In *Proc. CVPR* (2018), pp. 508–517.

[11] Black, Michael J., and Anandan, P. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Comput. Vis. Image Underst. 63*, 1 (1996), 75–104.

[12] Brostow, Gabriel J., Fauqueur, Julien, and Cipolla, Roberto. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters* (2008).

[13] Brostow, Gabriel J., Shotton, Jamie, Fauqueur, Julien, and Cipolla, Roberto. Segmentation and recognition using structure from motion point clouds. In *ECCV* (2008), pp. 44–57.

[14] Brox, Thomas, and Malik, Jitendra. Large displacement optical flow: Descriptor matching in variational motion estimation. *PAMI 33*, 3 (2011), 500–513.

[15] Bruhn, Andres, Weickert, Joachim, and Schnörr, Christoph. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision 61* (02 2005), 211–231.

[16] Bruss, Anna R., and Horn, Berthold K. P. Passive navigation. *Comput. Vis. Graph. Image Process. 21*, 1 (1983), 3–20.

[17] Butler, D. J., Wulff, J., Stanley, G. B., and Black, M. J. A naturalistic open source movie for optical flow evaluation. In *Proc. ECCV* (2012).

[18] Chang, Jia-Ren, and Chen, Yong-Sheng. Pyramid stereo matching network. In *Proc. CVPR* (2018).

[19] Cordts, Marius, Omran, Mohamed, Ramos, Sebastian, Rehfeld, Timo, Enzweiler, Markus, Benenson, Rodrigo, Franke, Uwe, Roth, Stefan, and Schiele, Bernt. The cityscapes dataset for semantic urban scene understanding. In *CVPR* (2016).

[20] Dave, Achal, Tokmakov, Pavel, and Ramanan, Deva. Towards segmenting anything that moves. In *IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops* (2019), pp. 1493–1502.

[21] Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, and Fei-Fei, Li. ImageNet: A large-scale hierarchical image database. *CVPR* (2009).

[22] Doersch, Carl, Gupta, Abhinav, and Efros, Alexei A. Unsupervised visual representation learning by context prediction. In *ICCV* (2015).

[23] Doersch, Carl, and Zisserman, Andrew. Multi-task self-supervised visual learning. In *ICCV* (2017).

[24] Dollár, Piotr, and Zitnick, C. Lawrence. Fast edge detection using structured forests. *IEEE Trans. Pattern Anal. Mach. Intell. 37*, 8 (2015), 1558–1570.

[25] Donahue, Jeff, Krähenbühl, Philipp, and Darrell, Trevor. Adversarial feature learning. In *ICLR* (2017).

[26] Dosovitskiy, Alexey, Fischery, Philipp, Ilg, Eddy, Hazirbas, Caner, Golkov, Vladimir, van der Smagt, Patrick, Cremers, Daniel, Brox, Thomas, et al. FlowNet: Learning optical flow with convolutional networks. In *Proc. ICCV* (2015).

[27] Eigen, David, Puhrsch, Christian, and Fergus, Rob. Depth map prediction from a single image using a multi-scale deep network. In *NIPS* (2014).

[28] Furukawa, Yasutaka, and Ponce, Jean. Dense 3d motion capture from synchronized video streams. In *Image and Geometry Processing for 3-D Cinematography* (2010), pp. 193–211.

[29] Gao, Ruohan, Jayaraman, Dinesh, and Grauman, Kristen. Object-centric representation learning from unlabeled videos. In *ACCV* (2016), pp. 248–263.

[30] Garg, Ravi, Kumar, B. G. Vijay, Carneiro, Gustavo, and Reid, Ian D. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *ECCV* (2016), pp. 740–756.

[31] Geiger, Andreas, Lenz, Philip, Stiller, Christoph, and Urtasun, Raquel. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research (IJRR)* (2013).

[32] Geiger, Andreas, Lenz, Philip, and Urtasun, Raquel. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR* (2012).

[33] Gibson, James J. *The Ecological Approach to Visual Perception*. Houghton Mifflin, 1979.

[34] Girshick, Ross B. Fast R-CNN. In *Proc. ICCV* (2015).

[35] Godard, Clément, Mac Aodha, Oisin, and Brostow, Gabriel J. Unsupervised monocular depth estimation with left-right consistency. In *Proc. CVPR* (2017).

[36] Guney, Fatma, and Geiger, Andreas. Displets: Resolving stereo ambiguities using object knowledge. In *Proc. CVPR* (2015).

[37] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proc. CVPR* (2016).

[38] Herbst, Evan, Seitz, Steve, and Baker, Simon. Occlusion reasoning for temporal interpolation using optical flow. Tech. rep., August 2009.

[39] Hinton, Geoffrey E., Vinyals, Oriol, and Dean, Jeffrey. Distilling the knowledge in a neural network. *CoRR abs/1503.02531* (2015).

[40] Horn, Berthold Klaus Paul. *Robot Vision*. MIT Press, Cambridge, MA, USA, 1986.

[41] Horn, B.K.P., and Schunck, B.G. Determining optical flow. *Artificial Intelligence* (1981).

[42] Hu, Yinlin, Li, Yunsong, and Song, Rui. Robust interpolation of correspondences for large displacement optical flow. In *CVPR* (2017).

[43] Huang, Gao, Liu, Zhuang, van der Maaten, Laurens, and Weinberger, Kilian Q. Densely connected convolutional networks. In *Proc. CVPR* (2017).

[44] Huguet, Frédéric, and Devernay, Frédéric. A variational method for scene flow estimation from stereo sequences. In *Proc. ICCV* (2007).

[45] Hui, Tak-Wai, Tang, Xiaoou, and Change Loy, Chen. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *Proc. CVPR* (2018).

[46] Hur, Junhwa, and Roth, Stefan. Joint optical flow and temporally consistent semantic segmentation. In *Proc. ECCV* (2016), Springer.

[47] Hur, Junhwa, and Roth, Stefan. MirrorFlow: Exploiting symmetries in joint optical flow and occlusion estimation. In *Proc. ICCV* (Oct 2017).

[48] Ilg, Eddy, Mayer, Nikolaus, Saikia, Tonmoy, Keuper, Margret, Dosovitskiy, Alexey, and Brox, Thomas. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR* (2017).

[49] Ilg, Eddy, Saikia, Tonmoy, Keuper, Margret, and Brox, Thomas. Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In *Proc. ECCV* (2018).

[50] Ioffe, Sergey, and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML* (2015).

[51] Isola, Phillip, Zoran, Daniel, Krishnan, Dilip, and Adelson, Edward H. Learning visual groups from co-occurrences in space and time. *arXiv preprint arXiv:1511.06811* (2015).

[52] Janai, Joel, Güney, Fatma, Behl, Aseem, and Geiger, Andreas. Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. *arXiv preprint arXiv:1704.05519* (2017).

[53] Janai, Joel, Güney, Fatma, Ranjan, Anurag, Black, Michael J., and Geiger, Andreas. Unsupervised learning of multi-frame optical flow with occlusions. In *ECCV* (2018).

[54] Janai, Joel, Güney, Fatma, Wulff, Jonas, Black, Michael, and Geiger, Andreas. Slow flow: Exploiting high-speed cameras for accurate and diverse optical flow reference data. In *CVPR* (2017).

[55] Jason, J Yu, Harley, Adam W, and Derpanis, Konstantinos G. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *ECCV* (2016), Springer, pp. 3–10.

[56] Jayaraman, Dinesh, and Grauman, Kristen. Learning image representations tied to ego-motion. In *ICCV* (2015), pp. 1413–1421.

[57] Jayaraman, Dinesh, and Grauman, Kristen. Slow and steady feature analysis: higher order temporal coherence in video. In *CVPR* (2016).

[58] Jiang, Huaizu, Larsson, Gustav, Maire, Michael, Shakhnarovich, Greg, and Learned-Miller, Erik. Self-supervised relative depth learning for urban scene understanding. In *European Conference on Computer Vision (ECCV)* (2018).

[59] Jiang, Huaizu, and Learned-Miller, Erik. Face detection with the faster R-CNN. In *IEEE International Conference on Automatic Face & Gesture Recognition (FG)* (2017).

[60] Jiang, Huaizu, Misra, Ishan, Rohrbach, Marcus, Learned-Miller, Erik, and Chen, Xinlei. In defense of grid features for visual question answering. In *CVPR* (2020).

[61] Jiang, Huaizu, Sun, Deqing, Jampani, Varun, Yang, Ming-Hsuan, Learned-Miller, Erik, and Kautz, Jan. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *CVPR* (2018), pp. 9000–9008.

[62] Jiang, Huaizu, Sun, Deqing, Jampani, Varun, Yang, Ming-Hsuan, Learned-Miller, Erik, and Kautz, Jan. SuperSloMo: High quality estimation of multiple intermediate frames for video interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[63] Jin*, SouYoung, Chowdhury*, Aruni Roy, Jiang, Huaizu, Singh, Ashish, Prasad, Aditya, Chakraborty, Deep, and Learned-Miller, Erik. Unsupervised hard example mining from videos for improved object detection. In *European Conference on Computer Vision (ECCV)* (2018).

[64] Johnson, Justin, Alahi, Alexandre, and Fei-Fei, Li. Perceptual losses for real-time style transfer and super-resolution. In *ECCV* (2016).

[65] Kendall, Alex, Martirosyan, Hayk, Dasgupta, Saumitro, Henry, Peter, Kennedy, Ryan, Bachrach, Abraham, and Bry, Adam. End-to-end learning of geometry and context for deep stereo regression. In *Proc. ICCV* (2017).

[66] Kingma, Diederik P., and Ba, Jimmy. Adam: A method for stochastic optimization. In *Proc. ICLR* (2015).

[67] Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *NIPS* (2012), pp. 1106–1114.

[68] Kundu, Abhijit, Vineet, Vibhav, and Koltun, Vladlen. Feature space optimization for semantic video segmentation. In *CVPR* (2016).

[69] Kuznietsov, Yevhen, Stückler, Jörg, and Leibe, Bastian. Semi-supervised deep learning for monocular depth map prediction. In *CVPR* (2017).

[70] Larsson, Gustav, Maire, Michael, and Shakhnarovich, Gregory. Learning representations for automatic colorization. In *ECCV* (2016).

[71] Larsson, Gustav, Maire, Michael, and Shakhnarovich, Gregory. Colorization as a proxy task for visual understanding. In *CVPR* (2017).

[72] Lee, Hsin-Ying, Huang, Jia-Bin, Singh, Maneesh, and Yang, Ming-Hsuan. Unsupervised representation learning by sorting sequences. In *ICCV* (2017).

[73] Li, Yin, Paluri, Manohar, Rehg, James M., and Dollár, Piotr. Unsupervised learning of edges. In *CVPR* (2016), pp. 1619–1627.

[74] Li, Yunpeng, and Huttenlocher, Daniel P. Learning for optical flow using stochastic optimization. In *ECCV* (2008).

[75] Liang, Xiaodan, Lee, Lisa, Dai, Wei, and Xing, Eric P. Dual motion GAN for future-flow embedded video prediction. In *ICCV* (2017).

[76] Liu, Ce, Freeman, William T., Adelson, Edward H., and Weiss, Yair. Human-assisted motion annotation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (2008).

[77] Liu, Ce, Yuen, Jenny, and Torralba, Antonio. SIFT flow: Dense correspondence across scenes and its applications. *IEEE Trans. Pattern Anal. Mach. Intell. 33*, 5 (2011), 978–994.

[78] Liu, Fayao, Shen, Chunhua, Lin, Guosheng, and Reid, Ian D. Learning depth from single monocular images using deep convolutional neural fields. *IEEE TPAMI. 38*, 10 (2016), 2024–2039.

[79] Liu, Ziwei, Yeh, Raymond, Tang, Xiaoou, Liu, Yiming, and Agarwala, Aseem. Video frame synthesis using deep voxel flow. In *Proc. ICCV* (2017).

[80] Long, Gucan, Kneip, Laurent, Alvarez, Jose M., Li, Hongdong, Zhang, Xiaohu, and Yu, Qifeng. Learning image matching by simply watching video. In *ECCV* (2016).

[81] Lotter, William, Kreiman, Gabriel, and Cox, David. Deep predictive coding networks for video prediction and unsupervised learning. In *ICLR* (2017).

[82] Lowe, David G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis. 60*, 2 (2004), 91–110.

[83] Lucas, Bruce D., and Kanade, Takeo. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence, IJCAI* (1981), Patrick J. Hayes, Ed., pp. 674–679.

[84] Luo, Wenjie, Schwing, Alexander G, and Urtasun, Raquel. Efficient deep learning for stereo matching. In *Proc. CVPR* (2016).

[85] Lv, Zhaoyang, Beall, Chris, Alcantarilla, Pablo F., Li, Fuxin, Kira, Zsolt, and Dellaert, Frank. A continuous optimization approach for efficient and accurate scene flow. In *Proc. ECCV* (2016).

[86] Lv, Zhaoyang, Kim, Kihwan, Troccoli, Alejandro, Sun, Deqing, Rehg, James, and Kautz, Jan. Learning rigidity in dynamic scenes with a moving camera for 3d motion field estimation. In *Proc. ECCV* (2018).

[87] Ma, Wei-Chiu, Wang, Shenlong, Hu, Rui, Xiong, Yuwen, and Urtasun, Raquel. Deep rigid instance scene flow. In *Proc. CVPR* (2019).

[88] Mahajan, Dhruv, Huang, Fu-Chung, Matusik, Wojciech, Ramamoorthi, Ravi, and Belhumeur, Peter. Moving gradients: a path-based method for plausible image interpolation. *ACM TOG 28*, 3 (2009), 42.

[89] Mathieu, Michael, Couprie, Camille, and LeCun, Yann. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440* (2015).

[90] Mayer, Nikolaus, Ilg, Eddy, Häusser, Philip, Fischer, Philipp, Cremers, Daniel, Dosovitskiy, Alexey, and Brox, Thomas. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR* (2016).

[91] Meister, Simon, Hur, Junhwa, and Roth, Stefan. UnFlow: Unsupervised learning of optical flow with a bidirectional census loss. In *AAAI* (New Orleans, Louisiana, Feb. 2018).

[92] Menze, Moritz, and Geiger, Andreas. Object scene flow for autonomous vehicles. In *Proc. CVPR* (2015), pp. 3061–3070.

[93] Meyer, Simone, Wang, Oliver, Zimmer, Henning, Grosse, Max, and Sorkine-Hornung, Alexander. Phase-based frame interpolation for video. In *CVPR* (2015).

[94] Misra, Ishan, Zitnick, C Lawrence, and Hebert, Martial. Unsupervised learning using sequential verification for action recognition. In *ECCV* (2016).

[95] Mobahi, Hossein, Collobert, Ronan, and Weston, Jason. Deep learning from temporal coherence in video. In *ICML* (2009).

[96] Niklaus, Simon, Mai, Long, and Liu, Feng. Video frame interpolation via adaptive convolution. In *CVPR* (2017).

[97] Niklaus, Simon, Mai, Long, and Liu, Feng. Video frame interpolation via adaptive separable convolution. In *ICCV* (2017).

[98] Noroozi, Mehdi, and Favaro, Paolo. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV* (2016).

[99] Owens, Andrew, Wu, Jiajun, McDermott, Josh H., Freeman, William T., and Torralba, Antonio. Ambient sound provides supervision for visual learning. In *ECCV* (2016).

[100] Pathak, Deepak, Girshick, Ross B., Dollár, Piotr, Darrell, Trevor, and Hariharan, Bharath. Learning features by watching objects move. In *CVPR* (2017).

[101] Pathak, Deepak, Krähenbühl, Philipp, Donahue, Jeff, Darrell, Trevor, and Efros, Alexei. Context encoders: Feature learning by inpainting. In *CVPR* (2016).

[102] Patraucean, Viorica, Handa, Ankur, and Cipolla, Roberto. Spatio-temporal video autoencoder with differentiable memory. In *ICLR, workshop* (2016).

[103] Radford, Alec, Metz, Luke, and Chintala, Soumith. Unsupervised representation learning with deep convolutional generative adversarial networks.

[104] Ranjan, Anurag, and Black, Michael J. Optical flow estimation using a spatial pyramid network. In *Proc. CVPR* (2017).

[105] Ranjan, Anurag, Jampani, Varun, Balles, Lukas, Kim, Kihwan, Sun, Deqing, Wulff, Jonas, and Black, Michael J. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR* (2019), pp. 12240–12249.

[106] Ranzato, Marc'Aurelio, Szlam, Arthur, Bruna, Joan, Mathieu, Michaël, Collobert, Ronan, and Chopra, Sumit. Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604* (2014).

[107] Ren, Zhe, Yan, Junchi, Ni, Bingbing, Liu, Bin, Yang, Xiaokang, and Zha, Hongyuan. Unsupervised deep learning for optical flow estimation. In *AAAI* (2017).

[108] Ren, Zhile, Sun, Deqing, Kautz, Jan, and Sudderth, Erik. Cascaded scene flow prediction using semantic segmentation. In *3DV* (2017).

[109] Revaud, Jerome, Weinzaepfel, Philippe, Harchaoui, Zaid, and Schmid, Cordelia. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *CVPR* (2015).

[110] Revaud, Jérôme, Weinzaepfel, Philippe, Harchaoui, Zaïd, and Schmid, Cordelia. Deepmatching: Hierarchical deformable dense matching. *Int. J. Comput. Vis. 120*, 3 (2016), 300–323.

[111] Rhemann, Christoph, Rother, Carsten, Wang, Jue, Gelautz, Margrit, Kohli, Pushmeet, and Rott, Pamela. A perceptually motivated online benchmark for image matting. In *CVPR* (2009).

[112] Ronneberger, Olaf, Fischer, Philipp, and Brox, Thomas. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI* (2015).

[113] Ros, Germán, Ramos, Sebastian, Granados, Manuel, Bakhtiary, Amir, Vázquez, David, and López, Antonio Manuel. Vision-based offline-online perception paradigm for autonomous driving. In *WACV* (2015).

[114] Roth, S., and Black, M. J. On the spatial statistics of optical flow. *IJCV 74*, 1 (2007), 33–50.

[115] Roy Chowdhury, Aruni, Chakrabarty, Prithvijit, Singh, Ashish, Jin, SouYoung, Jiang, Huaizu, Cao, Liangliang, and Learned-Miller, Erik G. Automatic adaptation of object detectors to new domains using self-training. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[116] Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C., and Fei-Fei, Li. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV) 115*, 3 (2015), 211–252.

[117] Schuster, René, Bailer, Christian, Wasenmüller, Oliver, and Stricker, Didier. Combining stereo disparity and optical flow for basic scene flow. In *CVTS* (2018).

[118] Schuster, René, Wasenmuller, Oliver, Kuschk, Georg, Bailer, Christian, and Stricker, Didier. Sceneflowfields: Dense interpolation of sparse scene flow correspondences. In *WACV* (2018).

[119] Seki, Akihito, and Pollefeys, Marc. Patch based confidence prediction for dense disparity map. In *BMVC* (2016).

[120] Sevilla-Lara, Laura, Sun, Deqing, Jampani, Varun, and Black, Michael J. Optical flow with semantic segmentation and localized layers. In *Proc. CVPR* (2016).

[121] Shelhamer, Evan, Long, Jonathan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. *TPAMI* (2017).

[122] Simonyan, Karen, and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556* (2014).

[123] Soomro, Khurram, Zamir, Amir Roshan, and Shah, Mubarak. Ucf101: A dataset of 101 human action classes from videos in the wild. *CRCV-TR-12-01* (2012).

[124] Springenberg, Jost Tobias. Unsupervised and semi-supervised learning with categorical generative adversarial networks. In *ICLR* (2016).

[125] Srivastava, Nitish, Mansimov, Elman, and Salakhutdinov, Ruslan. Unsupervised learning of video representations using lstms. In *ICML* (2015).

[126] Su*, Jong-Chyi, Wu*, Chenyun, Jiang, Huaizu, and Maji, Subhransu. Reasoning about fine-grained attribute phrases using reference games. In *International Conference on Computer Vision (ICCV)* (2017).

[127] Su, Shuochen, Delbracio, Mauricio, Wang, Jue, Sapiro, Guillermo, Heidrich, Wolfgang, and Wang, Oliver. Deep video deblurring. In *CVPR* (2017).

[128] Sun, D., Roth, S., Lewis, J. P., and Black, M. J. Learning optical flow. In *ECCV* (2008).

[129] Sun, Deqing, Roth, Stefan, and Black, Michael J. Secrets of optical flow estimation and their principles. In *IEEE Conference on Computer Vision and Pattern Recognition* (2010), pp. 2432–2439.

[130] Sun, Deqing, Sudderth, Erik B, and Black, Michael J. Layered image motion with explicit occlusions, temporal consistency, and depth ordering. In *Advances in Neural Information Processing Systems* (2010), pp. 2226–2234.

[131] Sun, Deqing, Yang, Xiaodong, Liu, Ming-Yu, and Kautz, Jan. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR* (June 2018).

[132] Sun, Deqing, Yang, Xiaodong, Liu, Ming-Yu, and Kautz, Jan. Models matter, so does training: An empirical study of cnns for optical flow estimation. *IEEE TPAMI* (2019).

[133] Szeliski, Richard. Prediction error as a quality metric for motion and stereo. In *ICCV* (1999).

[134] Taylor, Brian, Karasev, Vasiliy, and Soatto, Stefano. Causal video object segmentation from persistence of occlusions. In *Proc. CVPR* (2015), pp. 4268–4276.

[135] Teed, Zachary, and Deng, Jia. RAFT: recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision (ECCV)* (2020).

[136] Teichmann, Marvin, Weber, Michael, Zöllner, J. Marius, Cipolla, Roberto, and Urtasun, Raquel. Multinet: Real-time joint semantic reasoning for autonomous driving. In *CVPR* (2016).

[137] Tokmakov, P., Alahari, K., and Schmid, C. Learning motion patterns in videos. In *Proc. CVPR* (2017).

[138] Tokmakov, Pavel, Alahari, Karteek, and Schmid, Cordelia. Learning motion patterns in videos. In *CVPR* (2017).

[139] Tokmakov, Pavel, Alahari, Karteek, and Schmid, Cordelia. Learning video object segmentation with visual memory. In *ICCV* (2017).

[140] Vedula, Sundar, Baker, Simon, Rander, Peter, Collins, Robert, and Kanade, Takeo. Three-dimensional scene flow. In *Proc. ICCV* (1999).

[141] Vijayanarasimhan, Sudheendra, Ricco, Susanna, Schmid, Cordelia, Sukthankar, Rahul, and Fragkiadaki, Katerina. Sfm-net: Learning of structure and motion from video. In *arXiv* (2017).

[142] Vogel, Christoph, Schindler, Konrad, and Roth, Stefan. Piecewise rigid scene flow. In *Proc. ICCV* (2013), pp. 1377–1384.

[143] Vogel, Christoph, Schindler, Konrad, and Roth, Stefan. 3d scene flow estimation with a piecewise rigid scene model. *IJCV 115*, 1 (2015).

[144] Vondrick, Carl, Pirsiavash, Hamed, and Torralba, Antonio. Generating videos with scene dynamics. In *NIPS* (2016).

[145] Wang, J. Y. A., and Adelson, E. H. Representing moving images with layers. *IEEE Transactions on Image Processing 3*, 5 (Sept. 1994), 625–638.

[146] Wang, Ting-Chun, Zhu, Jun-Yan, Kalantari, Nima Khademi, Efros, Alexei A., and Ramamoorthi, Ravi. Light field video capture using a learning-based hybrid imaging system. *ACM TOG 36*, 4.

[147] Wang, Xiaolong, and Gupta, Abhinav. Unsupervised learning of visual representations using videos. In *ICCV* (2015).

[148] Wang, Xiaolong, He, Kaiming, and Gupta, Abhinav. Transitive invariance for self-supervised visual representation learning. In *ICCV* (2017).

[149] Warren, W., Kay, B., Zosh, W., Duchon, A., and Sahuc, Stephanie. Optic flow is used to control human walking. *Nature Neuroscience 4* (2001), 213–216.

[150] Wedel, Andreas, Rabe, Clemens, Vaudrey, Tobi, Brox, Thomas, Franke, Uwe, and Cremers, Daniel. Efficient dense scene flow from sparse or dense stereo data. In *Proc. ECCV* (2008), Springer, pp. 739–751.

[151] Weinzaepfel, Philippe, Revaud, Jérôme, Harchaoui, Zaïd, and Schmid, Cordelia. Deepflow: Large displacement optical flow with deep matching. In *ICCV* (2013).

[152] Wiskott, Laurenz, and Sejnowski, Terrence J. Slow feature analysis: Unsupervised learning of invariances. *Neural computation 14*, 4 (2002), 715–770.

[153] Wulff, Jonas, Sevilla-Lara, Laura, and Black, Michael J. Optical flow in mostly rigid scenes. In *Proc. CVPR* (2017).

[154] Xiao, Tete, Liu, Yingcheng, Zhou, Bolei, Jiang, Yuning, and Sun, Jian. Unified perceptual parsing for scene understanding. In *Proc. ECCV* (2018).

[155] Xu, Jia, Ranftl, René, and Koltun, Vladlen. Accurate optical flow via direct cost volume processing. In *Proc. CVPR* (2017).

[156] Xu, Li, Jia, Jiaya, and Matsushita, Yasuyuki. Motion detail preserving optical flow estimation. *TPAMI 34*, 9 (2012), 1744–1757.

[157] Xue, Tianfan, Wu, Jiajun, Bouman, Katherine, and Freeman, Bill. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *NIPS* (2016).

[158] Yang, Gengshan, and Ramanan, Deva. Volumetric correspondence networks for optical flow. In *Advances in Neural Information Processing Systems (NeurIPS)* (2019), Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, Eds., pp. 793–803.

[159] Yang, Guorun, Zhao, Hengshuang, Shi, Jianping, Deng, Zhidong, and Jia, Jiaya. SegStereo: Exploiting semantic information for disparity estimation. In *Proc. ECCV* (2018).

[160] Yin, Zhichao, Darrell, Trevor, and Yu, Fisher. Hierarchical discrete distribution decomposition for match density estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019).

[161] Yin, Zhichao, and Shi, Jianping. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR* (2018), pp. 1983–1992.

[162] Yu, Jason J., Harley, Adam W., and Derpanis, Konstantinos G. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *ECCV, workshop* (2016).

[163] Zbontar, Jure, and LeCun, Yann. Stereo matching by training a convolutional neural network to compare image patches. *JMLR* (2016).

[164] Zhang, F., Xu, S., and Zhang, X. High accuracy correspondence field estimation via mst based patch matching.

[165] Zhang, Richard, Isola, Phillip, and Efros, Alexei A. Colorful image colorization. In *ECCV* (2016).

[166] Zhang, Richard, Isola, Phillip, and Efros, Alexei A. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR* (2017).

[167] Zhao, Hang, Gallo, Orazio, Frosio, Iuri, and Kautz, Jan. Loss functions for image restoration with neural networks. *IEEE Trans. Computational Imaging 3*, 1 (2017), 47–57.

[168] Zhao, Hengshuang, Shi, Jianping, Qi, Xiaojuan, Wang, Xiaogang, and Jia, Jiaya. Pyramid scene parsing network. In *Proc. CVPR* (2017).

[169] Zhou, Tinghui, Brown, Matthew, Snavely, Noah, and Lowe, David G. Unsupervised learning of depth and ego-motion from video. In *CVPR* (2017).

[170] Zhou, Tinghui, Tulsiani, Shubham, Sun, Weilun, Malik, Jitendra, and Efros, Alexei A. View synthesis by appearance flow. In *ECCV* (2016).

[171] Zou, Yuliang, Luo, Zelun, and Huang, Jia-Bin. Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In *Proc. ECCV* (2018).