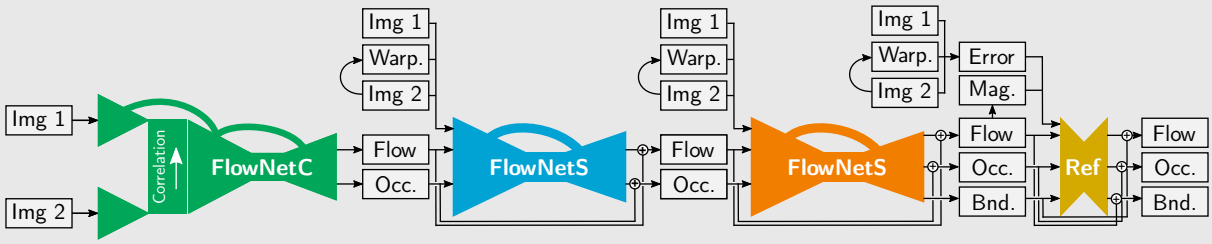
PhD Thesis

Estimating Optical Flow with Convolutional Neural Networks

Florian Eddy Robert Ilg



Dissertation zur Erlangung des
Doktorgrades der Technischen Fakultät der
Albert-Ludwigs-Universität Freiburg

Tag der Promotion / Datum der mündlichen Prüfung:	13.03.2020
Dekanin:	Prof. Dr. Hannah Bast
Erstgutachter und Betereuer:	Prof. Dr. Thomas Brox
Zweitgutachterin:	Prof. Dr. Cordelia Schmidt
Beisitzer:	Prof. Dr. Frank Hutter

Declaration

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work. I hereby also declare, that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Place, Date

Signature

Abstract

Optical flow estimation is a fundamental discipline in computer vision. Applications range from camera stabilization, image compression, action recognition and motion segmentation to structure from motion.

In the past, many attempts have been made to solve the problem by establishing an energy function and optimizing it with discrete or variational methods. The difficult aspects of optical flow, such as occlusions, discontinuities and the aperture problem, are hard to integrate into such energy minimization and pose inevitable limitations.

This thesis presents an orthogonal approach to estimate optical flow with Convolutional Neural Networks (CNNs) and shows that such networks are able to learn a better heuristic than engineered methods.

First, an end-to-end encoder-decoder network named FlowNetS and a Siamese network named FlowNetC with an explicit correlation unit are presented. The approach is then taken further to a network pipeline named FlowNet2, with several refinement stages in which occlusions and motion boundaries are also integrated.

The results show that optical flow estimation with CNNs is possible and that CNNs can perform among state-of-the-art methods while being orders of magnitude faster in runtime. Moreover, in motion boundary and occlusion estimation, CNNs significantly outperform traditional methods and are state of the art. Being able to estimate such high-quality flow in real time has changed the possible use cases and has had a significant impact on applications. Finally, CNNs have the advantage of being able to learn priors for specific scenarios as well as for the aperture problem from training data.

In order to take possible applications even further, a multi-hypothesis network named FlowNetH is introduced and a stack of networks to estimate an uncertainty measure along with the flow is presented. The evaluation shows that the uncertainties are state of the art, too, and that CNNs are able to inform about the reliability of their own flow predictions very well.

The reader is finally left with an outlook of how the approach can be brought to a multi-modal probabilistic setting and how it can be used as a building block for larger systems in future.

Zusammenfassung

Die Schätzung von optischem Fluss ist ein grundlegender Bestandteil der Bildverarbeitung. Die Anwendungen reichen von Kamerastabilisierung über Bildkompression, Handlugserkennung und Bewegungssegmentierung bis zu 3D-Rekonstruktion.

In der Vergangenheit wurden viele Ansätze entwickelt, um das Problem mittels einer Energiefunktion und diskreter oder kontinuierlicher Optimierung zu lösen. Die schwierigen Aspekte von optischem Fluss, wie Verdeckungen, Diskontinuitäten und das Aperturproblem, sind problematisch in solche Ansätze zu integrieren und führen zu Einschränkungen.

Diese Dissertation stellt einen neuartigen Ansatz vor, indem sie Convolutional Neural Networks verwendet. Es wird gezeigt, dass die neuronalen Netze eine bessere Heuristik lernen als die bislang händlich entwickelten Methoden.

Als erstes werden ein End-to-End Encoder-Decoder Netzwerk namens FlowNetS und ein siamesisches Netzwerk namens FlowNetC mit einer expliziten Korrelation vorgestellt. Der Ansatz wird dann weiterentwickelt zu einer Pipeline von Netzwerken, genannt FlowNet2, und in die einzelnen Stufen werden weiterhin Verdeckungen und Bewegungskanten integriert.

Die Resultate zeigen, dass die Schätzung von optischem Fluss mittels neuronaler Netze möglich ist und dass die Ergebnisse vergleichbar mit State-of-the-art-Ansätzen sind, derweil aber mit um Größenordnungen geringerer Laufzeit. Weiterhin sind die Netzwerke deutlich besser darin, Verdeckungen und Bewegungskanten zu erkennen, und setzen in diesen Bereichen einen neuen State of the art. Die Schätzung von solchen hochwertigen Flussfeldern in Echtzeit hat außerdem die möglichen Anwendungen revolutioniert und allgemein große Auswirkungen verursacht. Zuletzt haben die neuronalen Netze auch noch den Vorteil, dass man Priors für spezifische Anwendungsfälle und das damit verbundene Aperturproblem aus Trainingsdaten lernen kann.

Um die möglichen Anwendungen sogar noch weiterzuführen, werden anschließend ein Multi-Hypothesen-Netzwerk namens FlowNetH und eine Pipeline für die Schätzung von Unsicherheiten präsentiert. Die Auswertung zeigt, dass die Unsicherheiten ebenfalls State of the art sind und dass neuronale Netze sehr gut über die Zuverlässigkeit der eigenen Flussschätzung informieren können.

Dem Leser wird abschließend ein Ausblick darüber gegeben, wie man den Ansatz dazu erweitern kann, multimodale Wahrscheinlichkeitsverteilungen zu liefern, um diese zukünftig als Baustein für weitergehende, zuverlässigere Systeme zu verwenden.

To my science Teacher
Steve Fernandes

Contents

1 Introduction **1**
- 1.1 List of Publications . . . 3
- 1.2 Contributions by the Author . . . 4
- 1.3 Contributions to the Field of Computer Vision . . . 6

2 Problem Definition **9**
- 2.1 Optical Flow . . . 9
- 2.2 Disparity . . . 17
- 2.3 Scene Flow . . . 18
- 2.4 Evaluation . . . 19

3 Traditional Approaches **23**
- 3.1 Feature Descriptors . . . 23
- 3.2 Energy-Based Methods . . . 25
- 3.3 Combinatorial Methods . . . 30
- 3.4 Heuristics . . . 31
- 3.5 Occlusion, Depth and Motion Boundary Estimation . . . 34
- 3.6 Uncertainty Estimation . . . 36
- 3.7 Summary . . . 38

4 Convolutional Neural Network Basics **39**
- 4.1 Basic Concepts . . . 39
- 4.2 Training and Convergence . . . 41
- 4.3 Feature Hierarchies . . . 43

5 Training Data **45**
- 5.1 FlyingChairs . . . 46
- 5.2 ChairsSDHom . . . 47
- 5.3 FlyingThings3D . . . 48

6 FlowNet **51**
- 6.1 Network Architectures . . . 52
- 6.2 Analysis . . . 55
- 6.3 Summary . . . 64

7 SceneFlowNet **65**

8 Network Stacks **67**
8.1 Stacking two Networks for Flow Refinement 67
8.2 Stacking Multiple Diverse Networks 69
8.3 Evaluation on Applications 71
8.4 Summary 73

9 Joint Flow, Occlusion and Motion Boundary Estimation **75**
9.1 Estimating Occlusions with CNNs 75
9.2 Joint Estimation of Occlusions and Flow 77
9.3 Flow, Occlusion and Motion Boundary Estimation Stack 80
9.4 Benchmark Results 81
9.5 Application to Motion Segmentation 85
9.6 Summary 86

10 Extending Training with Unlabeled Images **87**
10.1 FusionNet 88
10.2 Augmented FlowNet 90
10.3 Experiments 91
10.4 Benchmark Results 94
10.5 Results on Motion Segmentation 94
10.6 Summary 95

11 Uncertainty Estimation **97**
11.1 Formulation of Uncertainty 99
11.2 Sources of Uncertainty 100
11.3 Bayesian Neural Networks and Frequentist Approximations 101
11.4 Predicting Multiple Hypotheses within a Single Network 106
11.5 Experiments 107
11.6 Summary 115

12 Discussion **117**
12.1 Architecture Choice 117
12.2 Regularization 120
12.3 Comparison of Algorithm Implementations 121

13 Outlook **123**

14 Conclusion **127**

Acknowledgements **130**

Bibliography **131**

Notes **153**

List of Figures

2.1 Illustration of optical flow . . . 10
2.2 Examples of apparent motion . . . 10
2.3 Rigid and nonrigid objects . . . 12
2.4 Rotation leading to nonlinear displacement field . . . 12
2.5 Illustration of occlusions . . . 13
2.6 Occlusions and disocclusions . . . 14
2.7 Homogeneous areas and features . . . 15
2.8 Illustration of the aperture problem . . . 16
2.9 Epipolar geometry . . . 17
2.10 Disparity example . . . 17
2.11 Scene flow definition . . . 18
2.12 Optical flow visualization . . . 19
2.13 Endpoint vs. angular error . . . 20
2.14 Overview of benchmark datasets . . . 22

3.1 Concept of HOG descriptors . . . 24
3.2 Effect of linearization . . . 27
3.3 Energy descent for nonconvex functions . . . 28
3.4 Elastic deformation to register images . . . 29
3.5 DeepMatching aggregation step . . . 31
3.6 Concept of DeepMatching . . . 31
3.7 PCA-Flow basis . . . 32
3.8 PCA-Flow reconstruction . . . 32
3.9 PatchMatch and FlowFields propagation . . . 33
3.10 Flow inconsistency due to occlusion . . . 35
3.11 Sparsification plot . . . 36

4.1 Neural network . . . 40
4.2 Activation functions . . . 40
4.3 Convolution variants . . . 41
4.4 Local minima in high-dimensional spaces . . . 42
4.5 Feature hierarchy learned by a CNN . . . 44

5.1 Two examples from the FlyingChairs dataset . . . 46
5.2 Displacement magnitude histograms . . . 47
5.3 Example images from the ChairsSDHom dataset . . . 48

5.4 Examples from the FlyingThings3D dataset 50

6.1 FlowNet 51
6.2 Siamese networks and stacked images 52
6.3 FlowNetS architecture 53
6.4 FlowNetC architecture 54
6.5 Decoder architecture 55
6.6 Data augmentation examples 57
6.7 Effect of skip connections 58
6.8 Image reconstruction from bottleneck 59
6.9 Effect of ground truth normalization 60
6.10 Visualization of first-layer filters of FlowNetC 60
6.11 Visualization of features after correlation 61
6.12 Performance over network size 62
6.13 Learning-rate schedules 63

7.1 Joint SceneFlowNet from a FlowNet and two DispNets 66

8.1 FlowNet2 network stack 67
8.2 FlowNet2 family endpoint error vs. runtime 71
8.3 FlowNet2 results on Sintel 72
8.4 FlowNet2 results on real images 72

9.1 Occlusion estimation 75
9.2 Joint flow and occlusion refinement stack variants 79
9.3 Flow, occlusion and motion boundary estimation stack 81
9.4 Qualitative results for occlusion estimation 83
9.5 Qualitative results for motion boundary estimation 83

10.1 Network acting as a regularizer on noisy data 88
10.2 Overview of the FusionNet principle 89
10.3 Data domain transfer by using FusionNet 90

11.1 Example of joint estimation of optical flow and its uncertainty 97
11.2 Probabilistic formulation of correspondence 98
11.3 Overview of ensemble generation approaches 103
11.4 Networks for uncertainty prediction 104
11.5 Sparsification plot of FlowNetH-Pred-Merged 109
11.6 Graphic evaluation of uncertainty estimation approaches 110
11.7 Full flow and uncertainty estimation stack 112
11.8 Uncertainty estimation examples from real-world data 113
11.9 Comparing FlowNetH variants to ProbFlow 114

12.1 Results during the refinement pipeline 118
12.2 Evaluation of coarse-to-fine estimation abilities 119
12.3 General and specialized architectures 120
12.4 Evaluation of regularization challenges 120

13.1 From point estimates to mixture distributions 124
13.2 Sampling and fitting framework 124
13.3 Using CNNs to build graphical models 125
13.4 Illustration of Bayes filter 126

List of Tables

2.1 Error regions 21

6.1 Solver results 56
6.2 Augmentation results 57
6.3 Deep supervision results 57
6.4 Results for using skip connections 58
6.5 Dataset schedules 63

7.1 SceneFlowNet compared to single FlowNet and DispNet 66

8.1 Ablation study for stacking two networks 68
8.2 FlowNet 2 stack variants 69
8.3 Motion segmentation and action recognition results by using different methods 74

9.1 Estimation of only occlusions from different inputs with a FlowNetS . . 76
9.2 Joint estimation of flow and occlusions with a FlowNetC 77
9.3 Results of joint flow and occlusion refinement stacks 80
9.4 Evaluation of estimated disparity occlusions from our DispNet3 82
9.5 Evaluation of estimated flow occlusions from our FlowNet3 82
9.6 Evaluation of estimated motion boundaries from our FlowNet3 82
9.7 Benchmark results for disparity estimation 84
9.8 Benchmark results for optical flow estimation 85
9.9 Motion segmentation results by using our occlusions 86

10.1 Comparison of FusionNet to the state of the art 92
10.2 Influence of the proxy ground truth on the augmented FlowNetC 93
10.3 Comparison of augmented FlowNet stacks to UnFlow 94
10.4 Benchmark results for FusionNet and AugmentedFlowNet 96
10.5 AugmentedFlowNet motion segmentation results 96

11.1 Improved FlowNetC settings 108
11.2 Quantitative evaluation of uncertainty estimation approaches 110
11.3 Benchmark results for FlowNetH variants 114

Chapter 1

Introduction

Concepts of Nature

Looking at human history, humans have ever been striving for improving their lives through technological innovations. While in the beginning, these innovations where as banal as knives and fire, technological advancement has reached an extremely high level today. The basis for such a development is the understanding of the concepts of nature so as to employ them in constructing sophisticated mechanisms.

Probably the most important human sense is vision, and hence, a very important question for technological advancement is how human visual abilities can be implemented by technology. In order to pursue this endeavour, the field of computer vision has evolved over time [Sze10]. The motivation that drives many researchers in this field is the goal to finally imitate the performance of the human visual system with by a machine [Bro05].

In many phenomena of nature, principles can be represented by clear physical laws, e.g. the free fall of a ball or the required support for a building. In unison with the human pursuit of understanding concepts, the field of computer vision has long been focusing on finding such laws in vision, too [Sze10]. However, almost all objects in nature are unique, so uncovering such laws often seems to be impossible. For example, no mathematical laws are known until the present day for identifying a human in an image. Instead, there are only statistics on what humans typically look like.

Machine Learning

The field of machine learning takes a contrary approach and does not try to model these laws, but instead seeks to build a machine to infer such laws from data [Bis06]. The far too complex dependencies in this case are not modelled by a human, but left to be determined by an algorithm from given training data. This may be contrary to the long-pursued practice in which humans do not seek to understand the laws anymore, but build a machine to do this work for them: Machine-learning approaches are designed "not to memorize the data but to learn the underlying generator" [Bis06]. Such learned models have shown very good generalization in recent history [KSH12, IS15, HZRS16, HLvdMW17]. Another important observation is that the performance of algorithms heavily depends on the representation of data [GBC16]. The main reason why an analysis of the image contents is so difficult lies in the large amount of unrelated data provided by a camera, and there is a priori no relation between pixels and objects [Bro05]. For example, if the representation is by *features*, such as foot, arm, torso and head instead of the image pixels, person detection becomes an easy task.

The breakthrough of machine learning came with deep learning, with which it became possible to learn a complete hierarchy of hidden feature representations [GBC16] in large networks [KSH12]. Keys to this success were the simple ReLU nonlinearity [NH10, KSH12] and the computation power of GPUs [KSH12]. Convolutional neural networks (CNNs) [LB98] have since then shown superior performance in almost every computer vision discipline [KSH12, SYLK18, UZU+17, ISB18, EPF14, RDGF16, DLHT16, RFB15].

Optical Flow

One of these disciplines is optical flow, which is fundamental to computer vision. Applications range from camera stabilization, image and video compression, action recognition and motion segmentation to structure from motion. While in the past, optical flow has been mostly addressed by discrete and variable methods, this thesis presents the initial breakthrough solution with CNNs and continues by elaborating the approach to competitive results in flow, disparity, occlusion, motion boundary and uncertainty estimation.

Optical flow estimation itself is a difficult optimization problem, contains many hyperparameters and suffers from the aperture problem [Sze10]. The latter makes the problem ill-defined, and in such cases, the desired solution can only be obtained with prior knowledge. Such prior knowledge needs to contain typical object connectivities and motion patterns. Again, it is infeasible to derive corresponding laws for all possible types of objects. Learning these rules from actual training data is where CNNs show their ultimate strength.

Before this work, CNNs were mainly successfully applied to classification [KSH12] and other cases of semantic segmentation [SLD17] as well as depth from single image [EPF14]. Correspondence estimation itself is very different from those problems in that it requires a matching algorithm. This work shows that CNNs are capable of also learning an approximation of such an algorithm end-to-end, thereby being orders of magnitude faster. In general, this proves that much better rules for inferring optical flow exist than have so far been discovered by humans with handcrafted methods [HS81, BM11, BTS15, WRHS13].

Uncertainty Estimation

While the exact rules learned by a CNN cannot be understood, it is most important that the learned rules model the underlying data and provide the best solution for the task to be solved. However, one can then argue about how this generalizes – that the CNN is a black box and that there is no reliability measure for its output. Part of this thesis will therefore be focusing on showing that reliability measures (*uncertainties*) can be obtained very well along with the flow predictions. The results show that the uncertainties from CNNs actually outperform traditional methods and that one can trust them even more than the engineered approaches.

Occlusions and Motion Boundaries

Other important modalities to flow are occlusions and motion boundaries, which are both very strong cues for motion segmentation. Referring to the classic work of Black and Jepson [BF00a], "motion boundaries may be useful for navigation, structure from motion, video compression, perceptual organization and object recognition". For traditional methods, they provide a significant problem: they require discrete decisions and cannot easily be integrated into convex energy minimization [HR17, PRCBP16]. Instead, occlusions are regarded as outliers to the data term and motion boundaries

are regarded as outliers to the smoothness term [BM11]. This causes problems if the amount of occlusions and motion boundaries in the image is large. They are therefore usually estimated in a disjoint post-processing step. Since they already complicate the flow estimation itself, estimating them from the flow is thus not very reliable. It is interesting to see how CNNs perform on these modalities, and this thesis shows that CNNs outperform traditional methods for occlusions and motion boundaries by far, too.

Outlook The work concludes that CNNs can learn an extremely efficient heuristic for optical flow (with an immense impact for practical applications), are among the state of the art in flow estimation as well as in occlusion, motion boundary and uncertainty estimation. The reader is finally left with an outlook on how the approach can be brought to a multi-modal probabilistic setting and how it be used as a building block for larger systems.

1.1 List of Publications

This thesis summarizes the following publications (equal contributions indicated with *):

[DFI+15] **FlowNet: Learning Optical Flow with Convolutional Networks**
Alexey Dosovitskiy*, Philipp Fischer*, Eddy Ilg*,
P. Häusser, C. Hazırbaş, V. Golkov, P. Smagt, D. Cremers and Thomas Brox
IEEE International Conference on Computer Vision (ICCV), 2015

[MIH+16] **A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation**
Nikolaus Mayer*, Eddy Ilg*, Philip Häusser*, Philipp Fischer*,
D. Cremers, A. Dosovitskiy and Thomas Brox
IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016

[IMS+17] **FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks**
Eddy Ilg,
N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy and Thomas Brox
IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017

[UZU+17] **DeMoN: Depth and Motion Network for Learning Monocular Stereo**
Benjamin Ummenhofer*, Huizhong Zhou*,
J. Uhrig, N. Mayer, Eddy Ilg, A. Dosovitskiy and Thomas Brox
IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017

[MIB17] **End-to-End Learning of Video Super-Resolution with Motion Compensation**
Osama Makansi,
Eddy Ilg and Thomas Brox
German Conference on Pattern Recognition (GCPR) 2017

[KBI+19] **Lucid Data Dreaming for Multiple Object Tracking**
Anna Khoreva,
Rodrigo Benenson, Eddy Ilg, Thomas Brox and Bernt Schiele
International Journal of Computer Vision, 2019

[MIF+18] **What Makes Good Synthetic Training Data for Learning Disparity and Optical Flow Estimation?**
Nikolaus Mayer,
Eddy Ilg, P. Fischer, C. Hazirbas, D. Cremers, A. Dosovitskiy and Thomas Brox
International Journal of Computer Vision, 2018

[IÇG+18] **Uncertainty Estimates for Optical Flow with Multi-Hypotheses Networks**
Eddy Ilg*, Özgün Çiçek*, Silvio Galesso*,
A. Klein, O. Makansi, F. Hutter and Thomas Brox
European Conference on Computer Vision (ECCV), 2018

[ISB18] **Occlusions, Motion and Depth Boundaries with a Generic Network for Optical Flow, Disparity, or Scene Flow Estimation**
Eddy Ilg*, Tonmoy Saikia*,
and Thomas Brox
European Conference on Computer Vision (ECCV), 2018

[MIÇB19] **Overcoming Limitations of Mixture Density Networks: A Sampling and Fitting Framework for Multimodal Future Prediction**
Osama Makansi, Eddy Ilg, Özgün Çiçek,
and Thomas Brox
IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019

[MIB18] **FusionNet and AugmentedFlowNet: Selective Proxy Ground Truth for Training on Unlabeled Image**
Osama Makansi*, Eddy Ilg*,
and Thomas Brox
arXiv:1808.06389 (20 Aug 2018)

1.2 Contributions by the Author

FlowNet This work presents deep networks for optical flow, disparity and scene flow estimation. The pioneering work, which has first shown that estimation of optical flow with a CNN is possible, was termed FlowNet [DFI+15] and was joint work together with Alexey Dosovitskiy and Philipp Fischer. The author of this thesis contributed with expertise in optical flow, to the concept of FlowNetC and the design of the correlation layer. Furthermore, the author implemented the variational refinement step.

Scene Flow In a follow-up work together with Nikolaus Mayer, Philip Häusser and Philipp Fischer, more sophisticated training datasets were provided and the networks were also applied to disparity and scene flow estimation. The author contributed the scene flow network.

FlowNet2 While [DFI+15] has first proven the completely novel approach, it did not reach a performance comparable to state-of-the-art methods and performed poorly in real-world applications. A major contribution by the author is the work of FlowNet 2.0 [IMS+17] (abbreviated as *FlowNet2*), proposing the concept of refinement in multiple steps by using a stack of several networks. That work was the sole idea of the author of this thesis. He contributed the concept, developed the differentiable warping operation and training schedules for the stack construction, and proposed the dataset priors for the ChairsSDHom dataset. This provided the first CNN to outperform or be on-par with traditional methods while being orders of magnitude faster. The revolutional accuracy/run-time trade-off enabled the previously impractical use of optical flow in numerous practical applications (see also Section 1.3). The analysis of these schedules and datasets further contributed to [MIF+18], and the concept of network stacks and refinement also contributed to [UZU+17].

Occlusions and Motion Boundaries Further work together with Tonmoy Saikia [ISB18] extended FlowNet2 to additionally estimate occlusions and motion boundaries, and applied the same stacks also to disparity. All the estimated quantities outperformed existing methods by far in terms of accuracy as well as run-time and achieved state-of-the-art results on various benchmarks [ISB18]. The author contributed the loss functions for motion boundaries and occlusions as well as the the network setups.

FusionNet / Augmented FlowNet A limitation of the CNN approach was still the need for large amounts of annotated training data, which could only be generated synthetically in a laborious process. The work with Osama Makansi termed FusionNet and AugmentedFlowNet [MIB18] has shown that optical flow computed from different existing methods can successfully be fused to create surrogate ground truth, enabling the successful training of FlowNet2 on large amounts of additional unlabeled data. This concept was developed together with Osama Makansi. The author also contributed the hinge loss for an unconstrained assessment metric.

FlowNetH The work together with Özgün Çiçek and Silvio Galesso investigated the ability to make not only a prediction of the optical flow, but to also provide an uncertainty estimate along with it [IÇG+18]. A major contribution by the author of this thesis is a CNN that predicts multiple hypotheses from a single network and in a single forward pass (named FlowNetH) from which the uncertainty can then be inferred. This approach gave rise to predict high-quality uncertainty without the need of time-consuming sampling.

Mixture Densities Since the uncertainty estimation for optical flow presented unimodal probability distributions as output, further work with Osama Makansi and Özgün Çiçek extended the approach to obtain well-calibrated mixture distributions [MIÇB19]. While this is in principle similar to Mixture Density Networks, the approach uses FlowNetH in an evolving manner with an algorithm named EWTA to internally ensure diversity. The author contributed mainly to the idea of EWTA which was developed jointly with Osama Makansi and Özgün Çiçek.

Minor The author also contributed to the DeMoN work [UZU+17] by transferring some of the concepts of FlowNet2 to depth-and-motion estimation. Other minor contributions include an end-to-end network for multi-frame video super-resolution [MIB17] that

has achieved state-of-the-art results (to which the author of this thesis contributed the network architecture) and integrating FlowNet2 into object tracking [KBI+19].

1.3 Contributions to the Field of Computer Vision

Flow Estimation with CNNs

The work of FlowNet was the first work to estimate optical flow with end-to-end trainable CNNs. The advantages of superior speed/accuracy trade-off and the ability to learn priors have led to a paradigm shift. Since then, many works have proposed variants and extensions for supervised and unsupervised learning for flow [YHD16, RB17, RYN+17, SZB17, ZLNH17, LHY17a, MHR18, SYLK18, HTL18] and disparity [MIH+16, KMD+17, PSR+17, GAB17, LFG+18].

Speed / Accuracy

The work of FlowNet2 presented results that are on par with state of the art while being orders of magnitude faster than any conventional method (see Figure 8.2). The results highlight crisp motion boundaries, the retrieval of fine structures and a robustness to compression artifacts. With these features, the work changed the possible applications of optical flow and has since been used for:

- Video super-resolution [PPSHS18, ZLX18, PPSHS18],
- video frame prediction and interpolation [FAR18],
- video style transfer [RDB18, GGZY18],
- video-to-video synthesis [WLZ+18],
- facial expression recognition [LZXW18],
- action detection and recognition [SKH+19, LLR18, HC16, ENT18, TXD+19, LC18, KDD18, Tka18, TLZ+19],
- video object detection and segmentation [ZDYW18, LL18a, LSV+18],
- video classification [CAA19, CAA17],
- semantic forecasting [TBL18],
- object tracking [ZCZ+18, KBI+19, VSF+18, XWW18],
- video and motion segmentation [NS18, VVB17, XFL+18, XFYL18, HWK+18, HHC+18, XBZ18, SEG+18, FAFM15, BWL18, LL18b, XXFH19, CCL+18, SAMR18],
- traffic flow analysis [TDDT+18],
- future person localization [YMYS18],
- vehicle velocity estimation [KMF18],
- visual odometry [ZLL18, ZSP+18] and
- autonomous driving and robot navigation [STB+18, HMLL18].

Occlusions and Motion Boundaries

Together with optical flow, occlusions and motion boundaries are a chicken-and-egg problem [HR17, PRCBP16]. The extension of FlowNet 2.0 to additionally estimate these quantities shows that CNNs can perform much better on solving this problem than engineered methods [ISB18]. Occlusions are ∼50% and motion boundaries ∼15% better than state of the art. While this work was recently published, it is expected that the results will boost action recognition and motion segmentation in the future.

Uncertainty Estimates

For all applications, it is furthermore vital to know how reliable the actual prediction is. Previous research has not investigated this question much – in fact only one of the state-of-the-art methods provides uncertainty estimates [WKR17]. The recent CNNs presented here outperform this work by a large margin [IÇG+18]. Despite their black-box nature, this shows that CNNs can be informative about their own limitations and can reliably be used for optical flow estimation in practice.

Summary

In summary, the work has introduced CNNs for optical flow estimation to computer vision, showing that they are able to perform among state of the art while being able to run in real time and being able to learn better priors, providing the best solution for occlusions and motion boundaries and being ready for real-world applications by informing about their prediction's quality.

Chapter 2

Problem Definition

Motivation In our everyday lives, we interact with a three-dimensional world. It is natural for us to know that certain objects have a certain distance, size and speed. We require this information to infer the scene layout and the actions of others, upon which we decide and coordinate our own actions. These can be tasks as simple as preparing a meal or getting dressed – while others are more critical, such as driving a car or crossing a road.

Sensors However, finding out the 3D motion of objects occurring in our surrounding is far from easy. In the first place, the 3D geometry of a scene can only be measured directly with active sensors, involving infrared or acoustic signals, and even then only under well-defined conditions. Sensors that work under general conditions are passive and cannot measure depth information. This is also the case for our visual system and for cameras: the perceived information is only a 2D projection of the scene. Figure 2.1 shows an example illustrating this.

2.1 Optical Flow

Displacement Vector Let $\boldsymbol{I}_1(\boldsymbol{x})$ and $\boldsymbol{I}_2(\boldsymbol{x})$ denote two different images and $\boldsymbol{x} = (x, y)$ a position within an image. A point $\boldsymbol{P}_1$ observed by a camera (shown in Figure 2.1) of an object visible in $\boldsymbol{I}_1$ is projected to image coordinates $\boldsymbol{x}_1$.

When either the object or the camera moves and a second image is taken, the 3D object is displaced to $\boldsymbol{P}_2 = \boldsymbol{P}_1 + \boldsymbol{W}$ in the camera coordinate system. If the point is still visible in $\boldsymbol{I}_2$, its coordinates can again be projected and $\boldsymbol{x}_2 = \boldsymbol{x}_1 + \boldsymbol{w}$. The points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are said to *correspond* to each other, and the vector $\boldsymbol{w} = (u, v)$ is called *displacement vector*.

Object and Apparent Motion w resembles the *object motion* [HS81], which is not necessarily congruent with the *apparent motion*. To illustrate this, some examples are shown in Figure 2.2. While the ball is rolling, the specular highlight appears to move differently from the ball in the scene. In the second example, the shadow of the car appears to be moving, although it does not resemble a moving object itself.

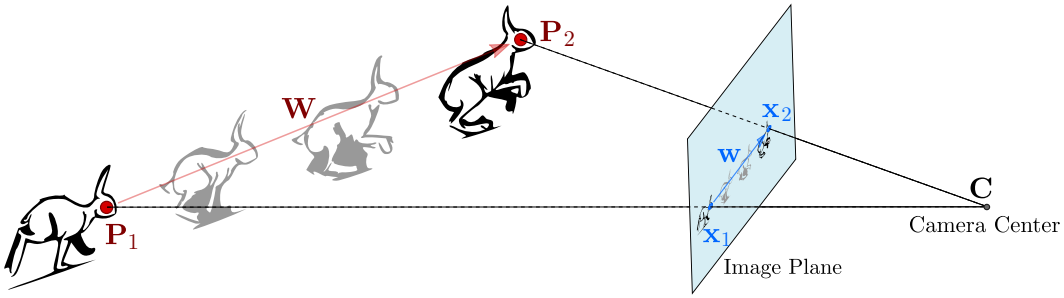
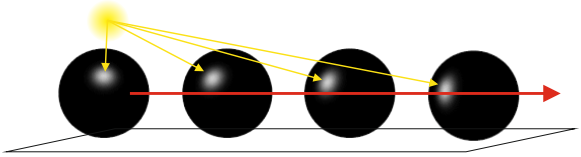


Figure 2.1: **Illustration of optical flow.** The rabbit moves in 3D space. The illustrated point $\boldsymbol{P}_1$ undergoes a change in 3D coordinates to $\boldsymbol{P}_2 = \boldsymbol{P}_1 + \boldsymbol{W}$ (shown in red). A camera with camera center $\boldsymbol{C}$ observes the rabbit by the 2D projection to its image plane and sees the 2D displacement $\boldsymbol{x}_2 = \boldsymbol{x}_1 + \boldsymbol{w}$ (shown in blue).



(a) A black ball with a reflective surface is rolling through a scene, indicated by the red arrow. The specular highlight follows some apparent motion caused by the light source, which neither corresponds to the ball's nor its surface motion. Source: [SN19].

(b) Although it is not a moving object, the shadow of the car causes an apparent motion. Source: [MG15].

Figure 2.2: **Examples of apparent motion.** The figures illustrate that apparent motion does not necessarily correspond to object motion.

Early literature defines *optical flow* as apparent motion [HS81, BR78, WS85]. In the first example, this corresponds to the motion of the specular highlight instead of the ball, while in the second example, the apparent motion is actually undefined. The reason for this is that the shadow is transparent and one can partially observe the underlying scene and partially the shadow.

The majority of applications actually requires knowledge about the object motion instead of apparent motion, as the object motion gives direct information about the scene. Benchmarks therefore truly measure how well optical flow algorithms estimate the object motion instead of the apparent motion [BSL+09, BWSB12, GLU12, MG15]. To this end, in this work, we define optical flow as the object motion, corresponding to the dense field $\boldsymbol{w}(\boldsymbol{x})$:

(2.1.1) The low-level vision task referred to as optical flow estimation is to infer the 2D object motion $\boldsymbol{w}(\boldsymbol{x})$ given only the two images $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$[a].

[a] Note that optical flow can also be estimated from more than two frames; this is a separate task and is referred to as *multi-frame* optical flow estimation.

2.1.1 Motion Boundaries

In the following, the properties of $\boldsymbol{w}(\boldsymbol{x})$ will be investigated further. Figure 2.3a shows a moving car. The first observation that can be made is that the boundaries of moving objects cause discontinuities in the optical flow field (shown in red in the figure). In this thesis, these discontinuities will be referred to as *motion boundaries*[1].

Motion, Depth and Object Boundaries

Motion boundaries frequently occur at the presence of object boundaries or depth discontinuities (the latter will here be referred to as *depth boundaries*). For example, the object boundaries of the car in Figure 2.3a separate it from the background, and since the car is moving, these object boundaries also result in motion boundaries. Furthermore, the car has a different distance to the camera than the background, and thus, the resulting depth boundaries can also be seen as a cause for the motion boundaries (a small exception is where the wheels touch the ground). However, it is important to note that while depth and object boundaries are correlated to motion boundaries, they do not necessarily imply them. In order to see this, one can simply consider a scene in which nothing moves: while there are many object and depth boundaries, motion boundaries do not exist.

(2.1.2) $\boldsymbol{w}(\boldsymbol{x})$ contains discontinuities. These are referred to as motion boundaries and often correspond to object and depth boundaries, but are not necessarily implied by them. Furthermore, motion boundaries in general do not allow a segmentation, as parts may be connected seamlessly.

2.1.2 Continuous Regions

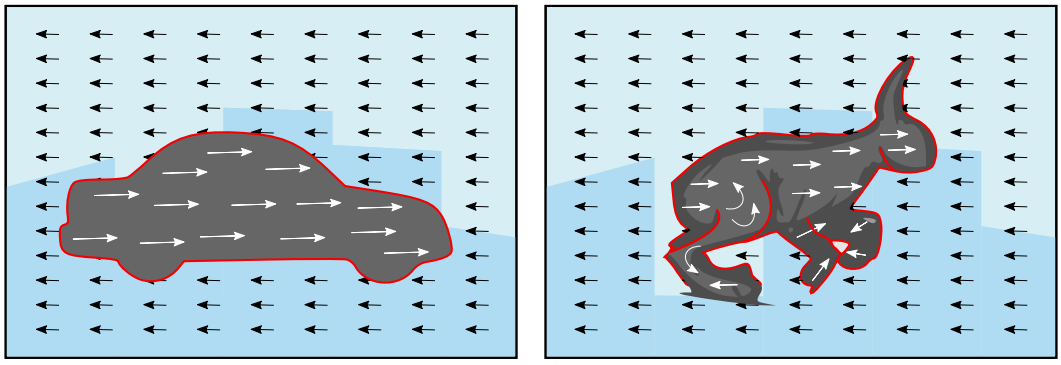
As motion boundaries model the discontinuities of $\boldsymbol{w}(\boldsymbol{x})$, its remaining regions are by definition continuous. Due to this continuity, a few approaches try to model these regions parametrically with affine transformations [SSB12, SWS+13, DB15] or homographies [YL15, HBK+14].

Parametric forms

Figure 2.4 shows a simple example of a planar surface rotating in front of the camera: due to the nonlinear projection of the camera, the resulting displacement field $u(x)$ in x-direction is nonlinear and can thus not be modeled by an affine transformation. In this particular example, the transformation could be correctly described by a homography. However, the general case of nonplanar objects also not be modeled by homographies. Concluding, homographies and affine transformations can only be used as local approximations.

In general, $\boldsymbol{w}(\boldsymbol{x})$ is generated by the 3D motion, the projection matrix and the object geometry. While the motion and projection matrix can in principle be modeled parametrically, the object geometry can take any form and therefore prohibits any general parametric model for $\boldsymbol{w}(\boldsymbol{x})$. Further problems arise if the object is nonrigid. This is illustrated in Figure 2.3b. In this case, the object can also not be described by piecewise parametric functions, as the transition between the pieces is smooth.

[1] in the literature, there is often a confusion between motion and occlusion boundaries [SBM+11, WRHS15]



(a) Schematic illustration of a car moving in front of a background. The car is rigid, and therefore the motion boundaries allow to clearly separate the car from the background.

(b) Schematic illustration of a deformable object moving. The object cannot be clearly segmented into parts. E.g., there is a smooth motion transition between the leg and the torso.

Figure 2.3: **Rigid and nonrigid objects.** Rigid objects can be segmented into layers, while nonrigid objects cannot. Motion boundaries are shown in red.

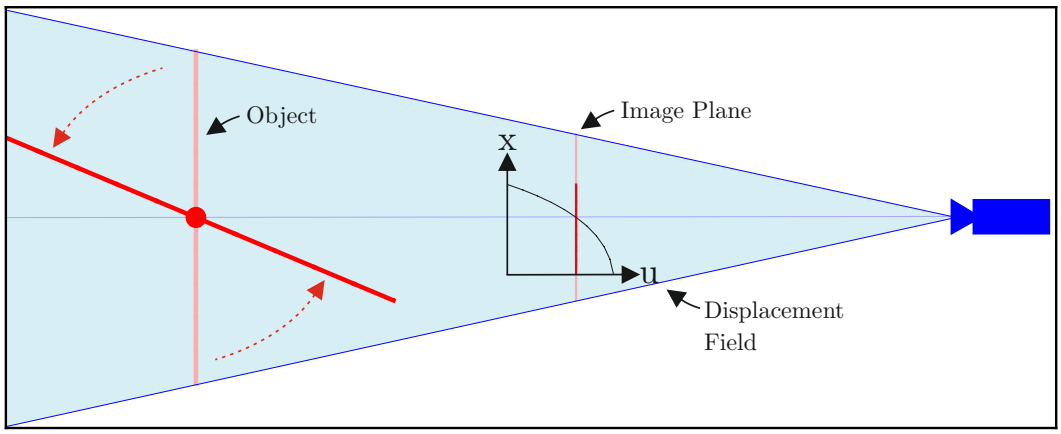


Figure 2.4: **Rotation leading to nonlinear displacement field.** The plane rotates around the indicated axis. The displacement of pixels $u(x)$ in x-direction is overlaid over the image plane. The simple motion of the planar object leads to a nonlinear displacement field.

This gives rise to the conclusion:

(2.1.3) The continuous regions of $\boldsymbol{w}(\boldsymbol{x})$ strongly depend on the unknown object geometry and can in general not be modeled parametrically.

2.1.3 Occlusions

In the 3D world observed by a camera, objects can only change their position over time, but do not suddenly disappear. However, from an observer's perspective, an object may be covered by another object and hence become partially or fully invisible.

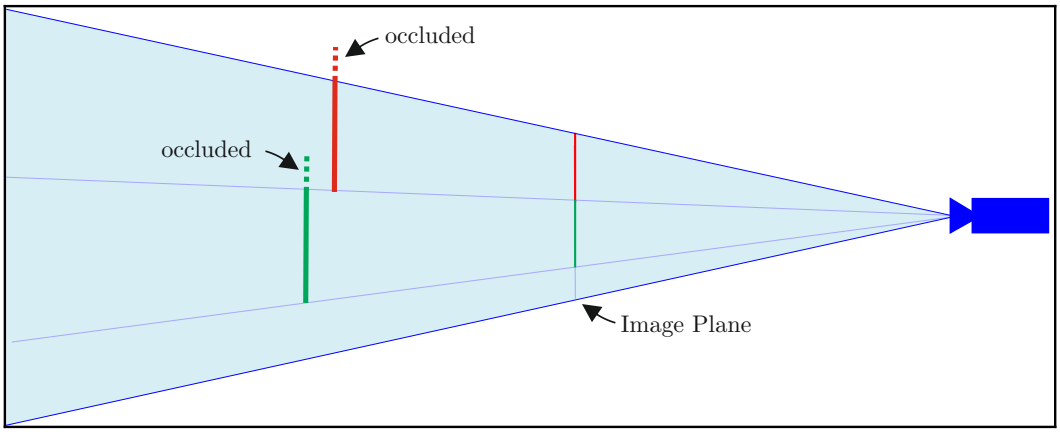


Figure 2.5: **Illustration of occlusions.** The red object is partially out of the field of view. The green object is occluded by the red object. Dotted lines indicate occluded regions not visible in the image.

Another reason that an object becomes partially or fully invisible is when it moves out of the field of view. Examples are given in Figure 2.5. Both cases can cause a pixel that is visible in $\boldsymbol{I}_1$ to be not visible in $\boldsymbol{I}_2$. For such cases, the pixels are said to be *occluded*, meaning that a correspondence in the second image is impossible to obtain from the observed data and without knowing the objects.

Occlusion and Disocclusion

In fact, whenever an object moves, it will cover some of the background and cause an occlusion. At the same time, it will also uncover some of the background and cause a *disocclusion*. For this reason, any type of motion of more than one object always implies the presence of occlusions and disocclusions, as illustrated in Figure 2.6. If more than one foreground object with different motions exist, the mutual occlusion relationships become more complex. In general, the size of an occluded area depends on the magnitude of the difference of the object's displacements (in the general case where both objects move), i.e., small displacement differences cause small occlusion areas and large displacement differences cause large occlusion areas (see also Figure 2.6).

Ground truth

It is important to note that for occluded pixels, a ground-truth optical flow still exists, that is the flow if the displaced object is projected to the second image, regardless of the occlusion. In the absence of such ground truth, it is sometimes possible to interpolate the optical flow in occluded areas by analyzing the surroundings. In the example from Figure 2.6 where parts of the background become occluded, the flow in the occluded area can be approximately interpolated from the surrounding background[2]. Since there is no correspondence data for the pixels in the occluded area, such an interpolation works well if the background in this area does not move or has no depth structure, i.e. if the flow field in this area should be smooth. If the background moves and has depth structure, it is impossible to approximate the flow field in the occluded area from the given information. Note that the background

[2] Note that one needs to know that the foreground object moves in front of the background to do this (deducible by that the foreground object is still completely visible in the second image).

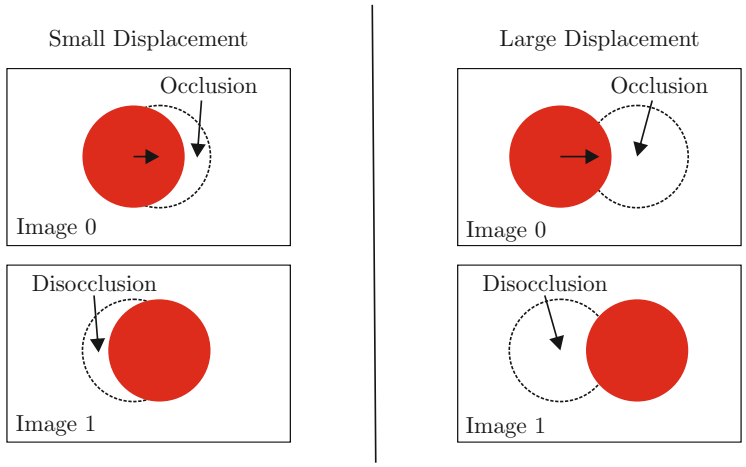


Figure 2.6: **Occlusions and disocclusions.** The left shows a small and the right a large displacement case. The red object moves to the right. By doing so, it will occlude part of the background (top) and disocclude another part of the background (bottom).

could also be any other object and that the explained relationships apply in the general case:

(2.1.4) If the camera or at least one object move, some pixels in an image become occluded. In this case, no correspondence can be found in the second image and $\boldsymbol{w}(\boldsymbol{x})$ can only be interpolated from the surroundings.

2.1.4 Prior Information and the Aperture Problem

Brightness Constancy

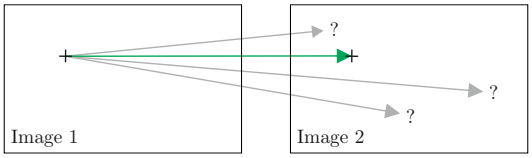
If we assume that the lighting conditions do not change and that there are no occlusions, this implies that apparent motion equals object motion and the observed pixel has the same appearance in both images. This is commonly formulated as the *brightness constancy* criterion:

(2.1.5)
$$\boldsymbol{I}_1(\boldsymbol{x}) = \boldsymbol{I}_2(\boldsymbol{x} + \boldsymbol{w}(\boldsymbol{x})).$$

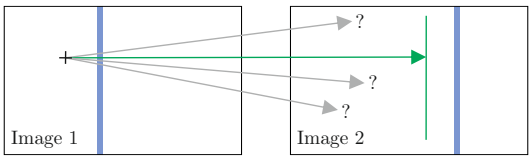
Uniqueness

However, the opposite is not true: pixels that have the same appearance do not necessarily imply a correspondence. This is due to the fact that many pixels may have the same appearance, which gives rise to one of the major difficulties in optical flow estimation. The worst-case scenario would be if $\boldsymbol{I}_1$ is homogeneous, i.e. only shows a single color value. In this case, the appearance of all pixels is the same, i.e. it is not possible to infer any correspondence, and any motion could be possible. This is illustrated in Figure 2.7a. Apart from the high and often infeasible computational complexity of optical flow, this means that even a brute force algorithm traversing the whole solution space cannot find the correct solution and priors are required instead:

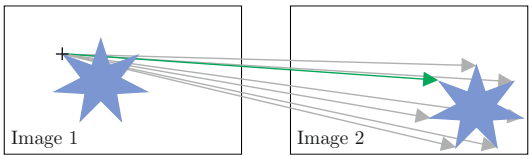
(2.1.6) Estimation of $\boldsymbol{w}(\boldsymbol{x})$ is not well-defined and priors are required. Correctness of priors highly influences the estimation quality.



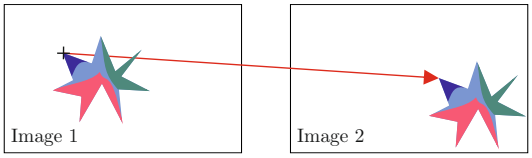
(a) In both images, there is no structure. For the location marked in the first image, the correspondence could be any location in the second image. The most plausible prior for a human is that nothing moves in the scene.



(b) We extend the example by a vertical line on the background, moving horizontally. For the location close to the line, a possible interpretation is that this location moves similar to the line; in fact, the most plausible prior for a human would be that it also moves horizontally.



(c) The object is now replaced by a star. The edges narrow down the correspondences to distinct locations. The most plausible prior would be that the star does not rotate, but in general, any of the indicated correspondences would be possible.



(d) There are now unique edge and color features on the object that exactly determine the correspondence.

Figure 2.7: **Homogeneous areas and features.** Possible correspondences are shown with gray arrows. Green arrows indicate the most plausible prior of a human, and red arrows indicate a uniquely determined correspondence.

Priors In the homogeneous example from Figure 2.7a, the most plausible prior for a human would be that nothing is moving (zero motion). This prior allows for obtaining a plausible $\boldsymbol{w}(\boldsymbol{x})$. However, note that priors fill the definition gap through assumptions that are true in the majority but can also be false: While in the largest number of cases, the zero motion is a plausible explanation for Figure 2.7a, in some cases, it might not be true and even human judgement would be wrong.

We extend the example to a line moving right, as illustrated in Figure 2.7b. Now a human would judge the image content to be moving to the right even in the homogeneous areas, i.e. due to the line feature, a motion in the neighboring areas is assumed. In the homogeneous areas, this solution is still only determined by the

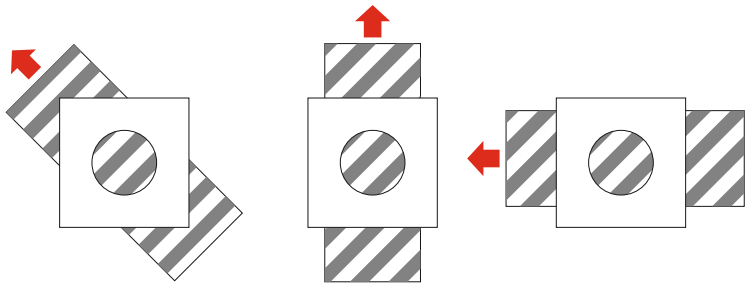


Figure 2.8: **Illustration of the aperture problem.** Three different background objects are shown that are moving in the directions top-left, up and left. The background object is covered by an aperture. When looking through the aperture, none of the three motions can be distinguished and are perceived identically. The true motion can only be determined when considering the whole context (i.e. seeing the whole background object). Thus, the plausible motion strongly depends on the aperture or the extent of the considered pixel compound.

prior and might be wrong but is the most plausible one. Note also that it is in general still unknown if the whole image content moves up or down, since the line gives a feature to explain only horizontal motion. Another, completely different possible interpretation is that the blue line is part of a static background covered by a moving white foreground with a slit. The true solution cannot be determined because white and blue areas are featureless. This insight gives rise to the following simple statement:

(2.1.7) Good estimation of $\boldsymbol{w}(\boldsymbol{x})$ requires unique features.

Pixel Compounds

In order to obtain a feature, a compound of multiple pixels needs to be considered. Examples are given in Figures 2.7c and 2.7d. The star is moving and a human can clearly tell its motion. Although the color inside the star is still homogeneous, the presence of many edge features increases the confidence that the star comprises an object and moves consistently. In summary, motion cannot be determined locally but only by considering a neighborhood, here referred to as a *compound* of pixels. This introduces another very difficult problem: determining the correct size of a compound. The *aperture problem* depicted in Figure 2.8 illustrates that different compounds are connected to different motion priors. If a compound has a certain boundary feature (aperture), the assumed prior motion can significantly differ. To summarize:

(2.1.8) Features consist of pixel compounds. Plausible sizes and connectivities for such compounds are determined by prior information and are not easy to obtain. The influence of the selected compound to the solution is illustrated by the aperture problem.

This presents another problem of optical flow estimation: while the basic complexity of optical flow is already high, a search for all possible compounds is infeasible.

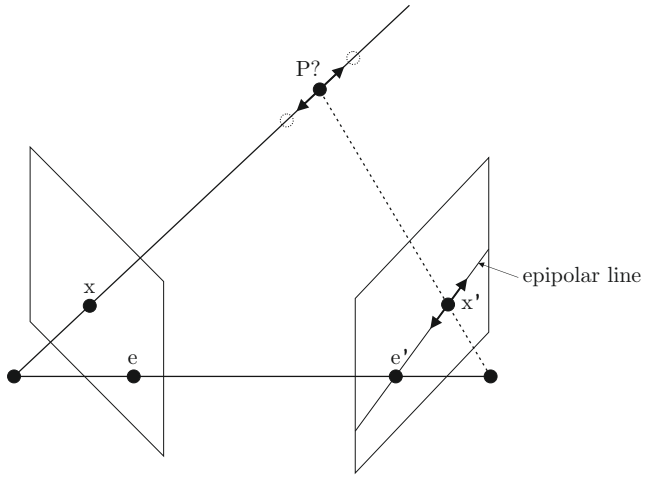


Figure 2.9: **Epipolar geometry.** The left camera observes a point **x**. The source 3D point **P** must lie on a line, which projects to the epipolar line in the right camera. By finding the 2D correspondence $\mathbf{x}'$, the point **P** can be uniquely determined.

Figure 2.10: **Disparity example.** The example shows the left and the right images from a stereo camera and the ground-truth horizontal displacement, referred to as disparity (data from [BWSB12]; the visualization actually shows the negative disparity; low intensity represents a high disparity and vice versa).

Prior Selection

This section has shown that prior information providing assumptions about object size and articulation play a very significant role. Such priors are actually necessary to overcome the ill-definedness of optical flow estimation and to integrate semantic knowledge depending on the image content. Such knowledge cannot be engineered well or only in a very limited form. While traditional techniques only integrate the very simplest prior of local smoothness (presented in Section 3.2), in this thesis, we argue that this is not sufficient. The key contribution is a learning approach that is able to learn sophisticated priors from data.

2.2 Disparity

Section 2.1 described that optical flow arises due to object or camera motion. The optical flow then represents the correspondences in the 2D projection. Assuming now that a scene is static and that the camera motion is known or fixed, such correspondences can be used for 3D reconstruction with the principle of triangulation. This is illustrated in Figure 2.9. In order to determine the depth of a point **P**, its correspondence $\mathbf{x}'$ needs to be determined. This correspondence is restricted to lying on the *epipolar line*. For this reason, the problem of depth estimation is a constrained optical flow estimation problem with one degree of freedom along the epipolar line.

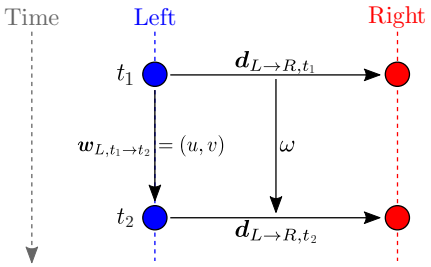


Figure 2.11: **Scene flow definition.** At times t_1 and t_2, the left and right images are recorded. The scene flow consists of the optical flow, the disparity and the change in disparity: $s = (u, v, d, \omega)$ width $d = d_{L\to R,t_1}$.

Stereo Camera

If the camera is furthermore displaced only horizontally, the epipolar lines also become horizontal. In this case, the optical flow can only be in the x-direction, i.e. $\boldsymbol{w} = (u, 0)$ and is referred to as *disparity*, where the disparity is defined as $d = -u$. Instead of displacing the camera, the images for disparity estimation are recorded with two cameras mounted onto a rig (referred to as a *stereo camera*). Since the images are recorded simultaneously and the relative displacement of the cameras does not change, the assumptions that the scene is static and the camera motion is known are valid.

Disparity and Depth

The estimated disparity is then inversely proportional to depth, as shown in Figure 2.10 (the visualization actually shows the negative disparity, which illustrates the depth structure of the scene), i.e. a high disparity corresponds to a close object and a low disparity to a distant object. The exact conversion between depth and disparity can be computed when the camera parameters are known. However, the advantage of the disparity over the depth representation is just that these parameters are not required and that disparity is a representation independent of the specific camera.

2.3 Scene Flow

Using a stereo camera to record images $\boldsymbol{I}_{L,t}$ and $\boldsymbol{I}_{R,t}$ over time allows to compute disparities $\boldsymbol{d}_{L\to R,t_1}$ and $\boldsymbol{d}_{L\to R,t_2}$ as well as the optical flow $\boldsymbol{w}_{L,t_1\to t_2}$. The optical flow (u, v) that represents a change in x- and y-direction can then furthermore be extended by a third component ω, representing the *disparity change*, as illustrated in Figure 2.11. The combination of optical flow and the disparity change is referred to as *scene flow* $\boldsymbol{s} = (u, v, d, \omega)$ width $d = d_{L\to R,t_1}$ [VBR+05, HD07, VSR15].

Redundancy

If the camera parameters are known, scene flow can be converted to the 3D motion of the underlying scene and is therefore fundamental for many robotic applications. If disparities and optical flow are estimated separately, estimating the disparity change is almost redundant, since it can be computed by subtracting the warped second disparity from the first. This holds up to the occluded regions where the disparity change needs to be interpolated similarly to optical flow.

Joint Estimation

However, despite the higher complexity, there can be a large benefit in estimating disparities and optical flow jointly [HD07]. First, if correct disparities were known, optical flow estimation becomes much easier because the additional disparity input

(a) Middlebury [BSL+09] color visualization

(b) Sintel [BWSB12] color visualization

Figure 2.12: **Optical flow visualization.** For the respective variants: left shows an image and middle shows the visualization of the motion to the next image. The direction of the flow field is visualized by hue and the magnitude by saturation. This is illustrated on the right. For the given exemplary flow vector, the resulting color for visualization is shown in the small box.

is highly discriminative for the input pixels. This eases the problem described in Section 2.1.4. Second, if correct optical flow was known, correspondence search for disparity also becomes more robust due to the additional constraint in both image pairs. It is therefore beneficial to jointly estimate optical flow and disparity for scene flow.

2.4 Evaluation

2.4.1 Visualization

For evaluation purposes, optical flow is commonly visualized by using a color circle, as illustrated in Figure 2.12. In order to visualize a flow vector, the vector is placed at the center of the circle and the color value at the head of the vector is used to visualize the pixel. So as to account for different motion magnitude ranges, flow vectors in an image are commonly prescaled by a factor to select the best range of the flow vectors to fall into the circle.

Two common versions exist: the Middlebury [BSL+09] (Figure 2.12a) and the Sintel [BWSB12] (Figure 2.12b) color schemes. If not stated otherwise, in this work, the Sintel visualization will be used.

2.4.2 Error Measures

The evaluation of optical flow is performed by comparing the estimated flow to the ground truth, which is obtained from object motion (contrary to the definition of optical flow in early works as the apparent motion [HS81, BR78, WS85]). The commonly used error metrics will be explained in the following:

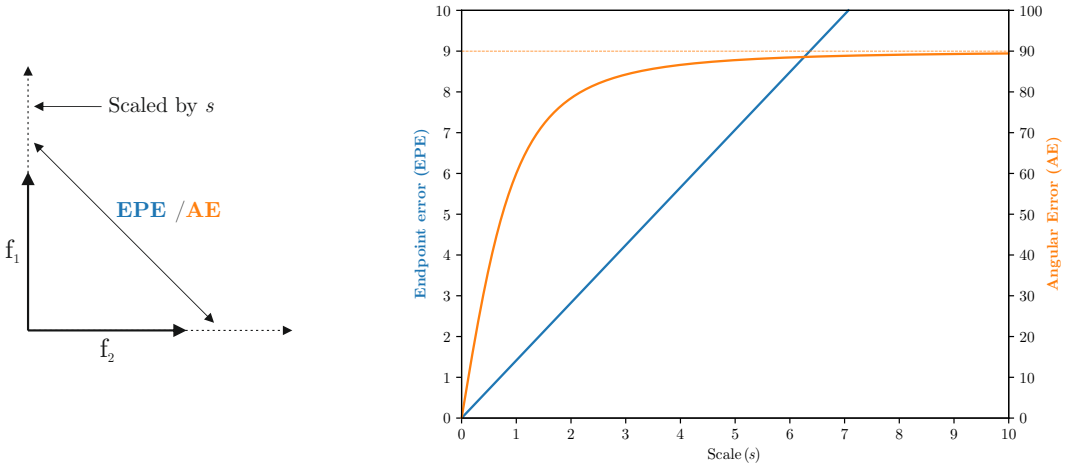


Figure 2.13: **Endpoint vs. angular error.** In order to demonstrate the meaning of the endpoint and the angular error, we compute the error between two unit orthogonal flow vectors (left) when scaling both with coefficients ranging from 0 to 10. The EPE grows linearly with the vectors while the AE grows fast for small vectors and saturates to 90° for large vectors.

Endpoint Error

The **endpoint error** is the most common metric and simply computes the magnitude of the difference between estimation and ground truth [ON06, BSL+09]:

$$\text{EPE} = ||\boldsymbol{w} - \boldsymbol{w}^{gt}|| = \sqrt{(u - u^{\text{gt}})^2 + (v - v^{\text{gt}})^2} \tag{2.4.1}$$

Note that for small flows, errors are usually small, and for large flows, errors are usually large. When computing the average over a dataset, image regions with large flows therefore tend to contribute to the average error significantly stronger and dominate the error.

Angular Error

The **angular error** computes the angle between the predicted and the ground-truth vector by extending them to 3D vectors [BFB94]:

$$\text{AE} = \measuredangle \left(\begin{pmatrix} u \\ v \\ 1 \end{pmatrix}, \begin{pmatrix} u^{\text{gt}} \\ v^{\text{gt}} \\ 1 \end{pmatrix} \right) \tag{2.4.2}$$

The extension to 3D vectors is done so as to avoid division by zero and to account for the magnitude of the vectors. Errors in large flows are penalized less [BSL+09], and for large flow vectors, the measure converges to the angular difference (see Figure 2.13).

Thresholds

Other benchmarks compute the endpoint error and then count the number of pixels that fall above a certain **threshold** [GLU12, MG15]: tE measures the percentage of pixels that have a higher error than E.

Regions

Measuring the errors in distinguished **regions** of the image allows to further assessing the properties of estimated flow fields, see Table 2.1 [BWSB12].

non-occ	error in regions that are visible in the second image
occ	error in regions that are not visible in the second image
d0-10, d10-60, d60-140, d140+	error in regions with different distances to motion boundaries
s0-10, s10-40, s40+	error in regions of different motion magnitudes

Table 2.1: **Error regions.** In order to assess the properties of estimated flow fields, [BWSB12] proposes to measure errors in different regions. "+" indicates an open range.

2.4.3 Benchmark Datasets

In order to assess the performance of algorithms, it is vital to have test cases that reflect real-world scenarios well. Since optical flow is a secondary feature, there is no sensor that can directly capture optical flow. This makes it a challenging task to obtain good test datasets. Existing approaches are described in the following:

Rigid Real-World Scenes

In a **rigid real-world scene**, the scene geometry can be captured by using structured-light or range scanning [BSL+09]. With a registration technique, the transformation between two instants of time can be computed. The scene geometry and the transformation together allow for inferring the optical flow.

Nonrigid Real-World Scenes

For **nonrigid real-world scenes**, the optical flow for the ground truth needs to be computed with an algorithm itself. One can increase the accuracy significantly by using hidden fluorescent texture [BSL+09] or high-frame-rate cameras [JGW+17]. The first requires that the scene is modified and explicitly prepared, while the second requires special lighting conditions.

Synthetically Generated Data

Rendering **synthetic scenes** gives the most control and the highest accuracy of the obtained optical flow fields. The difficulties are that creating photo-realistic image data is not trivial and that it is not obvious which aspects of the real world are relevant and must be modeled, since it is not possible to model all aspects perfectly [MIF+18].

The four common available benchmark datasets that serve for optical flow evaluation and their properties are shown in Figure 2.14. For a complete survey please refer to [MIF+18].

(a) The **Middlebury** dataset [BSL+09] comprises a mix of real and synthetic scenes with small displacements. 8 training and 8 test image pairs are provided. Invalid regions are left white in the flow visualization.

(b) The **Sintel** dataset [BWSB12] is rendered from a synthetic movie and comprises 23 training and 13 test datasets with 1,064 and 564 images respectively. The dataset models challenging cases with homogeneous areas and large motions. An additional final version with atmospheric effects and motion blur is also available (as given in the example). Ground truth for occluded areas is provided.

(c) The **KITTI 2012** dataset [GLU12] comprises 194 training and 195 different test image pairs. Pedestrians and scene are assumed to be static and are recorded by using a laser-range scanner, resulting in a sparse ground truth. Invalid regions are left white in the flow visualization. Fast-moving objects are excluded from the ground truth, such as the car to the right.

(d) The **KITTI 2015** dataset [MG15] comprises 200 test and 200 training image pairs. The dataset is recorded similar to the 2012 version, except that, when available, CAD models are fitted to moving cars (see dense ground truth for cars). In the example, the white truck in the background is excluded due to a missing model, and the pedestrian at the traffic light on the right is also excluded due to motion.

Figure 2.14: **Overview of benchmark datasets.**

Chapter 3

Traditional Approaches

Difficulties Chapter 2 indicates that optical flow estimation is a hard problem consisting of discontinuities that are not always implied by image or depth boundaries (Statement 2.1.2), continuous regions that can in general not be modeled parametrically (Statement 2.1.3), pixels that disappear (Statement 2.1.4) and the need for prior assumptions (Statement 2.1.6). Optical flow estimation has been a research field for almost 40 years [LK81], yet it is still referred to as an "unsolved problem" [FBK15].

The following presents the major concepts that have evolved and shows their respective advantages and disadvantages. Only the core concepts will be presented here that give an intuition of the problem and the findings of the past. Some of these intuitions are very important and serve as a motivation for the alternative approach taken in this thesis. For details of the traditional methods we refer the reader to the summary in [FBK15] and the respective works from the authors as cited.

3.1 Feature Descriptors

The past chapter explained that a single pixel is not enough to establish a correspondence and that it is necessary to consider compounds of pixels (Statement 2.1.8). Such a compound can be a local patch and can be represented by a *descriptor*:

(3.1.1) Descriptors summarize a local region of an image in a discriminative feature vector. Despite a compact representation, the goal of the feature vector is to be invariant to any appearance transformations, such as deformations and lighting conditions.

The simplest variant is to use the color values of the pixels in the region. However, such a simple aggregation of color values is vulnerable to any appearance transformations. The challenge of descriptors is thus to extract the important information from a patch, which can uniquely identify the patch and discriminate it from others.

HOG The most common variant is the Histogram of Orientations (HOG) descriptor [DT05], which is illustrated in Figure 3.1. Using gradients instead of color values makes the descriptor invariant to additive brightness changes. In order to allow for deformations,

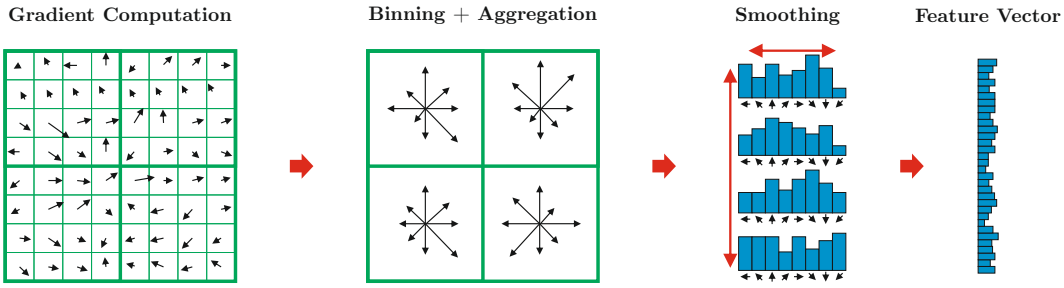


Figure 3.1: **Concept of HOG descriptors.** In the first step, image gradients are computed. They are then spatially aggregated into bins. In order to increase invariance and robustness, the resulting histograms are then smoothed in spatial and in bin direction. Finally, the histograms are aggregated to a feature vector.

the gradients are then binned into eight possible directions and spatially aggregated. Since values close to the bin boundaries may cause some abrupt changes, the next step is to perform smoothing in spatial and in bin direction. Finally, the bins are aggregated to a feature vector. The common input patch size is 16x16 pixels, which results in a 128-dimensional vector.

SIFT While the HOG descriptor is invariant to slight appearance changes, the Scale Invariant Feature Transform (SIFT) [Low04] adds complete rotation and scale invariance. This is achieved by first aligning to the dominant orientation and by using the characteristic scale to compute the descriptor among the right spatial extent. Other extensions of gradient histograms are GLOH [MS05], SURF [BETVG08] and DAISY [TLF10].

Census Transform A descriptor that is also commonly used in optical flow and has the strongest illumination invariance is the Census transform [ZW94]. In this variant, the center pixel is simply compared to the other pixels in the patch. The stored value is 0 if the outer pixel has a larger value than the center pixel; otherwise, the value is 1. For a detailed study, please refer to [HDW13].

Handcrafted vs. Learned The descriptors mentioned so far are all handcrafted, i.e. humans design heuristics to extract and summarize the important information of a pixel compound. Since the question which information is important depends highly on the data itself, it is reasonable to use a machine learning approach to find the best descriptors. Fisher et al. [FDB14] show that for matching, even the features from a CNN trained for classification outperform the handcrafted SIFT descriptors. Furthermore, descriptors can be trained to be explicitly optimized for the matching task. If ground truth is available, one can use a Siamese network architecture and a triplet loss, so as to minimize the distance to matches and to keep the distance to mismatches sufficiently large [WS09a, ŽL14, LTC17, XRK17].

Unsupervised Learning It is even more interesting to train for discriminative descriptors from unsupervised data. Dosovitskiy et al. [DFS+16] use a large set of surrogate patches and train a network to identify each of the patches under various input transformations. These transformations ensure that the CNN can reduce the patch to the important information regardless of its exact appearance while still being able to discrimi-

nate it from others. Similar approaches are the jigsaw puzzle approach [NF16] and split-brain autoencoders [ZIE17]. In summary:

(3.1.2) Optimal descriptors highly depend on the data. Recent works have shown that learned descriptors significantly outperform the engineered approaches [ŽL14, XRK17, FDB14, DFS$^+$16, NF16, ZIE17].

3.2 Energy-Based Methods

Most traditional methods regard optical flow and disparity estimation as an optimization problem [LK81, HS81, BBPW04, BM11]. This requires the formulation of an objective in terms of an energy function. The two most general assumptions that can be be deduced from the problem definition in Chapter 2 are:

(3.2.1) **Appearance Similarity.** As described by Equation 2.1.5, object motion implies appearance similarity. This is a necessary but not sufficient criterion for correspondence (Section 2.1.4). Appearance can be modeled by grayscale, color and image gradients [HS81, BBPW04, BM11] or with descriptors (see Section 3.1). In the energy function, the appearance similarity is modeled as the *data term* E_D.

(3.2.2) **Local Smoothness.** As described in Section 2.1.4, individual pixels are not enough and even larger structures can still be ambiguous and not sufficient to establish a correspondence. In general, the required compound size is unknown and depends on the image data; it can potentially span the whole image. The most general assumption that can therefore be made is that a pixel exhibits similar motion to its neighbors (with few exceptions among motion boundaries). This is modeled by the *smoothness term* E_S.

Both assumptions are then combined in an energy that is to be minimized with respect to the optical flow field $\boldsymbol{w}$:

(3.2.3) $$E(\boldsymbol{w}) = E_D(\boldsymbol{w}) + \alpha E_S(\boldsymbol{w})\,,$$

where α determines the trade-off between both terms. The concrete implementation of this energy function can be either in a continuous or in a discretized space and past research has investigated all kinds of variations [HS81, BBPW04, BM11, LYT11, XJM12, KK12, SSB12, SWS$^+$13, XRK17]. It is very important to note that occlusions cause fundamental violations to the data, and motion boundaries to the smoothness term. In this model, these violations cannot be resolved and are considered outliers. Implementations must therefore be designed to be as robust as possible to such outliers [BA96, MP98, BBPW04]. However, with more and more of these outliers (especially for large motions), the model becomes inaccurate and will eventually fail.

Discretization In general, a discretization of $\boldsymbol{w}$ means to quantize the flow into predefined labels and allows for treating the problem with combinatorial optimization. Since for optical flow, the label space is very large and general combinatorial optimization is NP-Hard, this is often infeasible or requires overly high runtimes, and the solution can therefore only be approximated. Treating $\boldsymbol{w}$ continuously allows for employing

the calculus of variation (referred to as *variational techniques*) but requires E to be convex for the optimization to be able to descend to the global minimum. In general, variational techniques are faster but limited by the convexity constraint. For this reason, the predominant method for the less complex disparity estimation problem is combinatorial optimization [KZ01a, KSK06, SHLC09, Hir05], while for optical flow, it is variational techniques [HS81, BBPW04].

3.2.1 Variational Methods

The pioneering work of Horn and Schunck [HS81] proposed to treat images as continuous functions and to implement Equation 3.2.3 as follows:

$$E(\boldsymbol{w}) = \int \underbrace{[\boldsymbol{I}_2(\boldsymbol{x}+\boldsymbol{w}) - \boldsymbol{I}_1(\boldsymbol{x})]^2}_{E_D} + \alpha \underbrace{[||\nabla u(\boldsymbol{x})||^2 + ||\nabla v(\boldsymbol{x})||^2]}_{E_S} d\boldsymbol{x}\,. \tag{3.2.4}$$

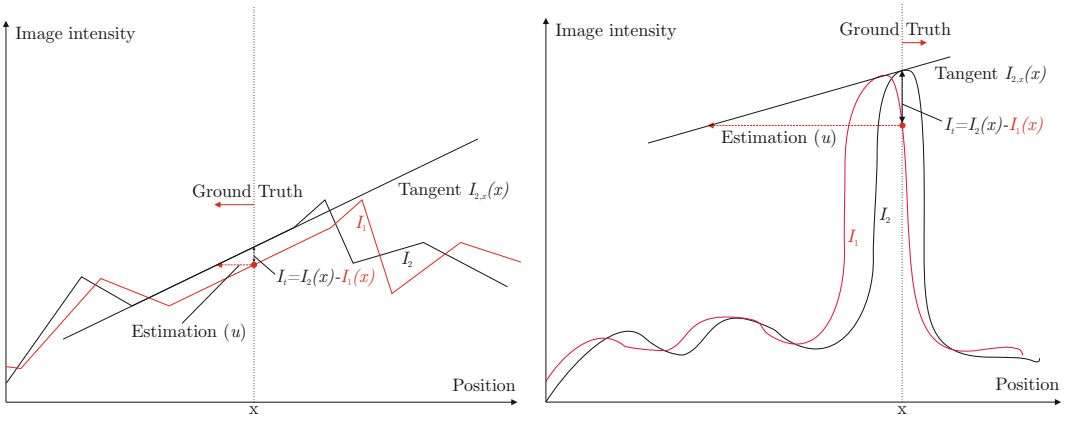
Prior Modeling The first part implements the data term and is the brightness constancy criterion (Equation 2.1.5). The second part implements the smoothness term that locally connects pixels and serves as a regularizer. This regularizer can also be seen as a prior that prefers smooth flow fields (see Statement 2.1.6). The hyper-parameter α controls the influence of this prior. Note that a second prior is the initialization of the optimization with the zero solution.

Linearization A fundamental problem is that the image is a highly nonlinear function. In order to obtain a solution, Horn and Schunck therefore propose a linearization of the first term:

$$\begin{aligned} \boldsymbol{I}_2(\boldsymbol{x}+\boldsymbol{w}) - \boldsymbol{I}_1(\boldsymbol{x}) &= 0 \\ \Leftrightarrow \boldsymbol{I}_{2,x}u + \boldsymbol{I}_{2,y}v + \boldsymbol{I}_2 - \boldsymbol{I}_1(\boldsymbol{x}) &= 0 \\ \Leftrightarrow \boldsymbol{I}_{2,x}u + \boldsymbol{I}_{2,y}v + \boldsymbol{I}_t &= 0, \end{aligned} \tag{3.2.5}$$

where indices x, y, t indicate spatial and temporal derivatives. The third equation is referred to as the *optic flow constraint*. With the linearization in place, the energy function 3.2.4 becomes convex. Obtaining the Euler-Lagrange equations and discretizing leads to a linear system that can be solved to find the global optimum. However, images are highly nonlinear in general. The problems that arise from this linearization are shown in Figure 3.2 (see next page). In general, the linearization limits the approach to small displacements, as images are approximately linear only locally. Depending on the amount of nonlinearity in the image, this can be within the range of one pixel. Furthermore, outliers due to occlusions and motion boundaries can severely influence the solution. In order to weaken the effect of the latter, a robust penalty function Ψ can be used [BA96, MP98, BBPW04], which minimizes the influence of outliers while still maintaining the convexity of Equation 3.2.4.

Gradient Constancy Brox et al. [BBPW04] later extended the data term by gradient constancy. While the gradient term is not invariant to rotations, it adds more invariance to brightness changes. Xu et al. [XJM12] claim that the combination of both terms can often be suboptimal and that one needs to further optimize for a weighting factor between them.



(a) The image shows a piecewise linear 1D image. Following the tangent to obtain $I_{2,x}u = -I_t$ allows for obtaining the ground-truth solution exactly.

(b) The same case for a highly nonlinear 1D image, as is the case for most real images. Following the tangent to obtain $I_{2,x}u = -I_t$ yields a solution that is far off from the ground truth.

Figure 3.2: **Effect of linearization.** The linearization of the Horn and Schunck [HS81] method works for piecewise linear images but fails for nonlinear ones.

GAUSS-NEWTON

In order to overcome the problems associated with linearization, Brox et al. proposed to keep the nonlinearity and to solve Equation 3.2.4 with the Gauss-Newton method. To demonstrate this a simplified version shall be shown here. Not linearizing the data term from Equation 3.2.4 requires finding its minimum by setting the first derivative to 0. Assuming that u would be a scalar, this leads to:

$$\begin{aligned} \frac{\delta}{\delta u}(\boldsymbol{I}_2(\boldsymbol{x}+\boldsymbol{w}) - \boldsymbol{I}_1(\boldsymbol{x}))^2 &= 0 \\ \Leftrightarrow 2(\boldsymbol{I}_2(\boldsymbol{x}+\boldsymbol{w}) - \boldsymbol{I}_1(\boldsymbol{x}))\frac{\delta}{\delta u}\boldsymbol{I}_2(\boldsymbol{x}+\boldsymbol{w}) &= 0. \end{aligned}$$

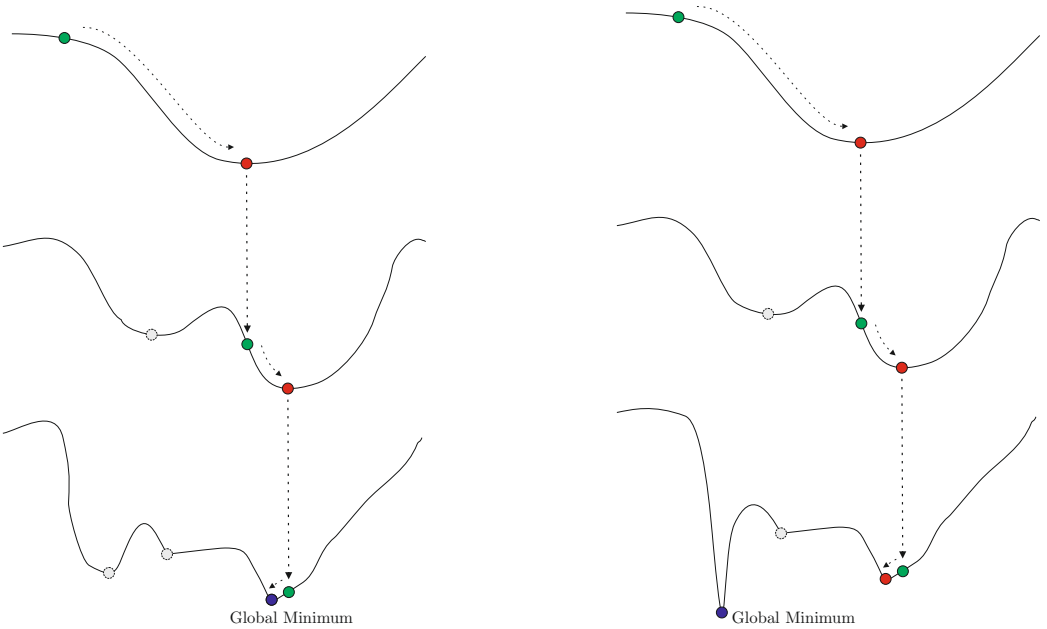
Defining $\boldsymbol{I}_z = \boldsymbol{I}_2(\boldsymbol{x}+\boldsymbol{w}) - \boldsymbol{I}_1(\boldsymbol{x})$ and $\boldsymbol{I}_x = \frac{\delta}{\delta u}\boldsymbol{I}_2(\boldsymbol{x}+\boldsymbol{w})$ gives:

$$\boldsymbol{I}_z\boldsymbol{I}_x = 0. \tag{3.2.6}$$

In order to fulfill equation 3.2.6, it is sufficient that $\boldsymbol{I}_z = 0$. Using Gauss-Newton, we define an iterative scheme and linearize the update step:

$$\begin{aligned} \boldsymbol{I}_z^{k+1} &= \boldsymbol{I}_2(\boldsymbol{x}+\boldsymbol{w}^{k+1}) - \boldsymbol{I}(\boldsymbol{x}) \\ &= \boldsymbol{I}_2(\boldsymbol{x}+\boldsymbol{w}^{k}) + \boldsymbol{I}_x^k du^k + \boldsymbol{I}_y^k dv^k - \boldsymbol{I}(\boldsymbol{x}) \\ &= \boldsymbol{I}_x^k du^k + \boldsymbol{I}_y^k dv^k + \boldsymbol{I}_z^k \end{aligned}$$

The algorithm then iterates $u^{k+1} = u^k + du^k$ and $v^{k+1} = v^k + dv^k$. It is important to note that the linearization is only performed for the update step, which is a much more local criterion than the global linearization from Equation 3.2.5. The linearization always occurs on the second *warped* image, i.e. at the current solution $\boldsymbol{x} + \boldsymbol{w}^k$. Note that the example here is only a sketch and strongly simplified; in general one needs



(a) A case demonstrating successful convergence to the global minimum.

(b) A failure case in which the local minimum is too narrow (frequently appearing with small objects) and the minimum is incorrectly smoothed out in the coarse level, leading to a sub-optimal solution.

Figure 3.3: **Energy descent for nonconvex functions.** Spatial pyramids operate on different levels of smoothing from coarse to fine (top to bottom in the Figure). On each level, a minimum is obtained and used for initialization of the next level. Local minima are indicated in gray.

to consider u and v as continuous functions, add also the smoothness term, integrate over the image domain and use the Euler-Lagrange equations. For a more detailed derivation, the reader is referred to [Bro18].

Coarse-to-Fine

Using the Gauss-Newton method allows for solving the nonlinear Equation 3.2.4. However, as $\boldsymbol{I}$ is nonconvex, the energy also becomes nonconvex which in turn means the method can in general converge to a local minimum instead of the global one. In order to mitigate the danger of local minima, Brox et al. [BM11] employ a spatial pyramid, i.e. they start with strongly smoothed images to allow for a coarse estimation of large structures and then proceed to sharper images for refinement. This concept is illustrated in Figure 3.3. Technically, the smoothing is implemented by downsampling and is commonly also referred to as a *coarse-to-fine* approach [BBPW04].

Limitations

The coarse resolution allows for identifying large displacements of large compounds, while the fine resolution allows for the identification of small displacements of small compounds. This introduces a dependency between displacement and object size: while it allows to solve for large displacements, the object size also needs to be sufficiently large, otherwise it will be smoothed away on the coarse levels of the

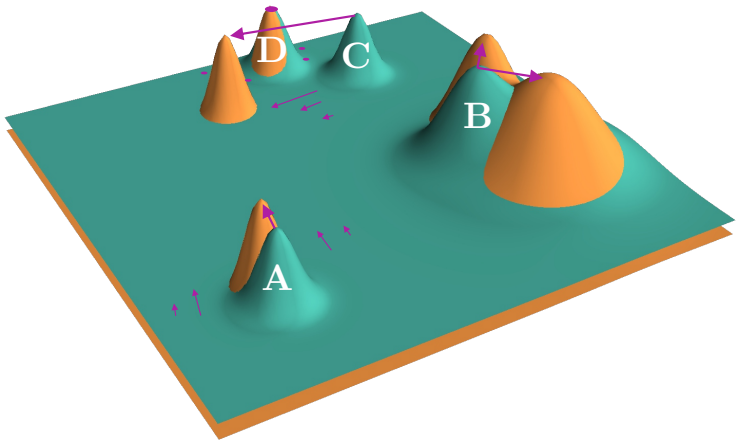


Figure 3.4: **Elastic deformation to register images.** The surfaces represent image intensities. The green image is to be elastically deformed by a flow field w to match the orange image. A) shows a case where the correct deformation can easily be obtained with the Gauss-Newton method. B) shows an ambiguous case where the solution is not uniquely determined. C) shows a case where the displacement is too large to obtain a solution with energy minimization, so that descriptor matching is required. Depending on the smoothing coefficient α, it may not be possible to reach the solution, because close-by D) is already at a local minimum.

spatial pyramid (see Figure 3.3b). The method therefore succeeds in finding large displacements of large objects, but fails for large displacements of small objects. This is especially the case for small body parts that can move very fast [BM11] and leads to the following conclusion:

(3.2.7) Variational methods can solve very well for sub-pixel-accurate displacements. They are doomed to fail where the motion of a small-scale structure is larger than its own scale [BM11].

Descriptor Matching

An approach to mitigate this limitation is to use descriptor matching which can capture pixel compounds that correspond to small objects. In their work called Large Displacement Optical Flow (LDOF), Brox et al. [BM11] propose to use nearest-neighbor descriptor matching and to integrate it into the energy function:

$$E(\boldsymbol{w}) = E_{color}(\boldsymbol{w}) + \gamma E_{gradient}(\boldsymbol{w}) + \alpha E_{smooth}(\boldsymbol{w}) + \beta E_{match}(\boldsymbol{w}, \boldsymbol{w}_1),$$

where the last term enforces similarity of the estimated flow field to previously determined matches w_1. The matches obtained with nearest-neighbor matching do not follow a regularization and can be very noisy. However, since the descriptors are integrated into a variational method here, it is expected that the variational method will perform the necessary regularization; in addition to that, some pre-filtering is performed.

Intuitively, the explained variational techniques can be imagined as treating the images as two surfaces (with the height indicating color intensity) and elastically deforming the second surface by "sliding" it onto the first one. This is illustrated in Figure 3.4. If the initial solution is too far away from the correct one, a local minimum

might lie in-between. If the smoothness term is high, the neighboring solutions can also "pull" the solution over the local minimum (but overly high smoothness terms discourage discontinuities in the flow field and a trade-off needs to be made). The spatial pyramid approach solves this by bringing the solution close to the correct one on a coarser resolution first, provided that the object size is large enough. In summary:

(3.2.8) Variational methods require descriptor matching in order to work for large displacements of small objects. This brings us back to the beginning of obtaining high-quality and well-regularized descriptor matches.

For this reason, the work of Brox et al. [BM11] turned the focus of the field to investigating many descriptor matching techniques [XJM12, WRHS13, LZS13, BTS15, WFR+16, GG16b, XRK17]. Some works even conjecture that descriptor matches are the main step towards the solution and that sub-pixel-accurate results can be obtained by a mere interpolation of descriptor matches [RWHS15].

3.3 Combinatorial Methods

Complexity Combinatorial optimization can handle arbitrary energy functions, but is not feasible for optical flow in a general form due to the large label space. E.g., for a VGA image there are more than 300k possible labels per pixel.

Motion Candidates In order to reduce the possible labels, approaches therefore try to find few possible motion candidates first which then serve as the possible label set. Motion Detail Preserving Flow (MDPFlow) [XJM12] tries to extract possible candidates by finding the dominant motions from SIFT matches in the image, in addition to the nearest match from patch matching. They then use QBPO [RKLS07] to optimize an energy function of the form of Equation 3.2.3.

DiscreteFlow [GG16b] uses a hierarchical search to find the K best neighbors from DAISY [TLF10] descriptors and then proceeds with optimizing a similar energy function using BCD [CK14]. Drayer et al. [DB15] propose a combinatorial refinement of an initial matching field with affine hypotheses.

Layers Other approaches try to introduce depth ordering and layers into optical flow [SSB12, SWS+13]. (Note that due to Statement 2.1.2, the segmentation of optical flow is generally not possible and this can only be an approximation.) Given approximate layers and depth ordering, occlusions can also be inferred and integrated into the energy function, which is then finally optimized by using QBPO [SSB12] or variational EM [SWS+13].

SGM In disparity estimation, large displacements are more predominant and less labels are required. Combinatorial methods are therefore much more popular for this task [KZ01a, KSK06, SHLC09] which can be approximated and implemented very efficiently with Semi-Global Matching (SGM) [Hir05]. The work of Xu et al. [XRK17] named DCFlow recently presented an approach that extends SGM also to optical flow.

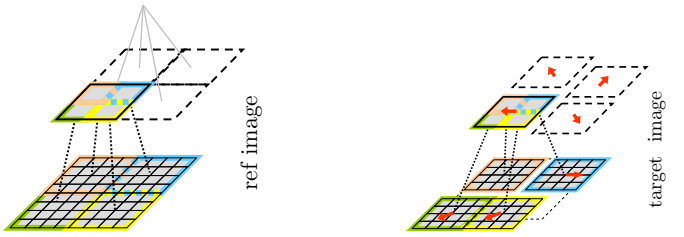


Figure 3.5: **DeepMatching aggregation step.** In order to compute the response of a larger patch, the responses of smaller patches are first max-pooled (right) and then aggregated. Source: [WRHS13].

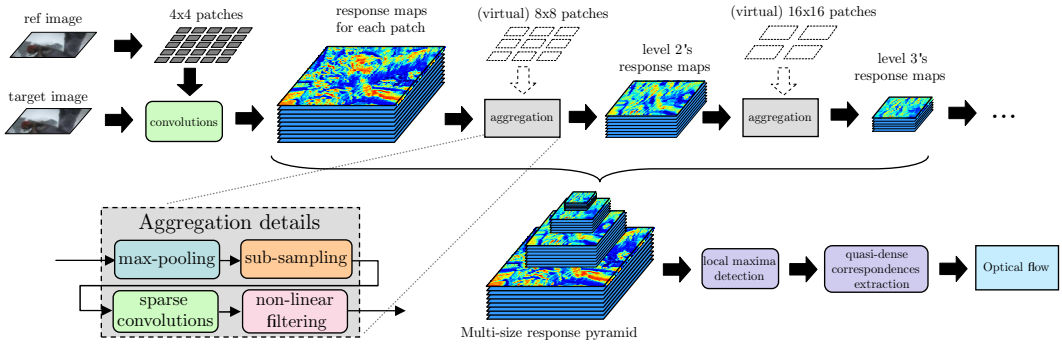


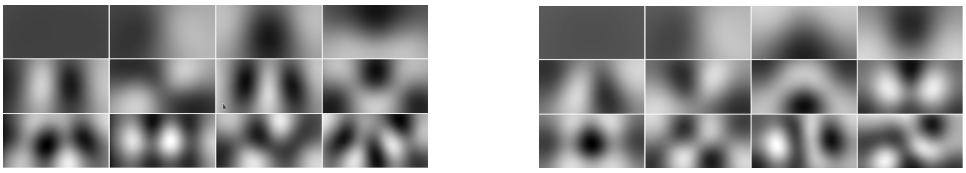
Figure 3.6: **Concept of DeepMatching.** 4x4 patches are used to compute dense HOG descriptor maps in both images. For each patch in the reference image, a response map is created by convolving the patch with the second image. The response maps are then aggregated as illustrated in Figure 3.5. The aggregation happens by max-pooling, sub-sampling and summing up the smaller patch's response values. Repeating the procedure yields a multi-size response pyramid from which maxima are extracted and backtracked to the images in order to obtain correspondences. Source: [WRHS13].

3.4 Heuristics

Section 3.2 explained that variational approaches rely on accurate precomputed descriptor matches. Obtaining such matches with an energy minimization framework for optical flow is not feasible in general. For this reason, many heuristics have been developed and some selected ones will be presented here. It is interesting to compare to these heuristics, as the neural networks presented in this work also resemble a different variant of heuristic.

3.4.1 DeepMatching

DeepMatching by Weinzaepfel et al. [WRHS13] is a heuristic to obtain regularized descriptor matches. The algorithm is motivated by deep networks (although it has no learnable weights and is different in nature). The concept transferred from deep



(a) Horizontal (b) Vertical

Figure 3.7: **PCA-Flow basis.** The first 12 components for the basis learned by PCA-Flow from 120,000 frames. Separate components are learned for horizontal and vertical direction (shown left and right). Source: [WB15].

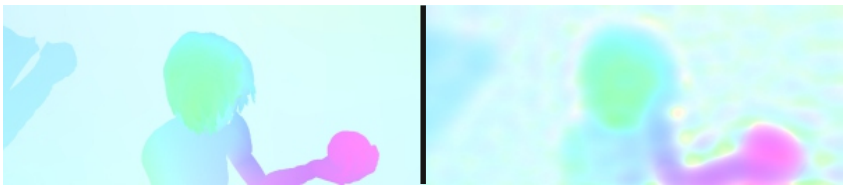


Figure 3.8: **PCA-Flow reconstruction.** The Figure shows the reconstruction of the ground truth (left) through the learned basis (right). Source: [WB15].

networks is the aggregation and pooling. In order to compute the score of a larger feature match, the scores of the composing parts are aggregated and max-pooled. This is illustrated in Figure 3.5.

Aggregation and Pooling

The whole algorithm is illustrated in Figure 3.6. The first step of the algorithm is to compute dense response maps for all descriptors. Taking the maximum of these response maps would yield nearest-neighbor matching. However, DeepMatching proceeds with max-pooling and spatially aggregating them (i.e. summing them up). Repeating this aggregation yields response maps for different resolutions (capturing different receptive fields). Thereby, a large patch has a high response value if the composing small patches also have high response values. The small patches may vary their positions slightly in this process, which allows for some limited deformations.

Fine-to-Coarse

In order to obtain final correspondences, maxima are then extracted from the response pyramid. At different levels, they resemble different object sizes. The maximum is finally tracked back through the pooling operations to obtain the dense image correspondences. Note that the algorithm already takes correspondences as input instead of images. It therefore only serves as a regularization by aggregating consistent regions to larger compounds and can be seen as a fine-to-coarse approach.

3.4.2 PCA-Flow

PCA-Flow by Wulff et al. [WB15] takes a different approach and represents the flow field for an image pair through a linear combination of basis vectors:

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i \boldsymbol{b}_i$$

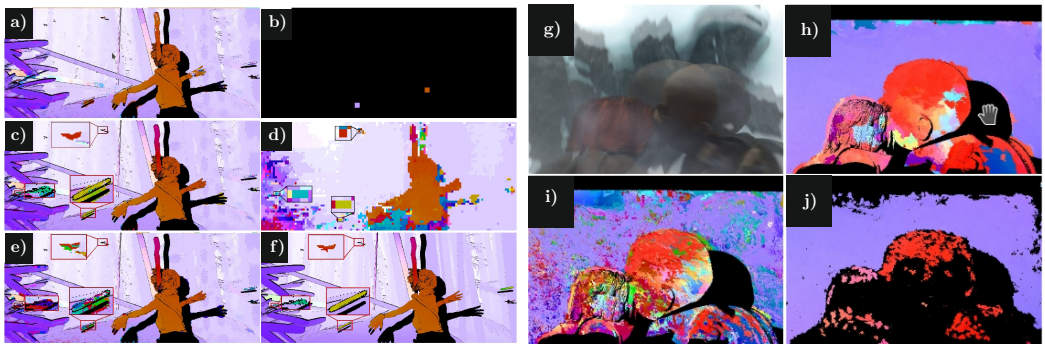


Figure 3.9: **PatchMatch and FlowFields propagation.**
Left: (occlusions masked out with ground truth) **a)** shows the FlowFields solution obtained only from the two seeds shown in b). **c)** shows the FlowFields solution obtained from the k-d tree nearest-neighbor matches shown in d). **e)** shows the FlowFields results without hierarchies (without large distance propagation). **f)** shows the ground truth. Notably, the algorithm can already achieve very good results with only two seeds. Using the full k-d tree initialization with hierarchies gives the best result c).
Right: g) shows overlaid images with large displacement. **i)** shows the matches obtained with approximate nearest neighbors. Notably, the flow field is noisy due to the approximate search. **h)** shows the flow field obtained by FlowFields. There are significantly less outliers, but due to the missing regularization, some outlier regions remain. **j)** shows the flow field after removing outliers with forward-backward consistency checks.
Source: [BTS15].

Learned Basis

The basis itself is learned from data. They use 120,000 frames from Hollywood movies with flow computed by GPUFlow [WTP+09]. The basis is then extracted by using principle component analysis (PCA) and has 250 components for horizontal and vertical direction respectively (500 in total). The first twelve components of the learned basis are shown in Figure 3.7 (see previous page). Interestingly, the basis resembles the basis functions of a Discrete Cosine Transform (DCT).

Given a new image pair, estimating the flow field then becomes the task of finding the coefficients α_i. To this end, they use sparse nearest-neighbor descriptor matching and solve for $\boldsymbol{w}$ with linear least squares. The method relates to this work in that it uses learning and runs in real time.

3.4.3 PatchMatch and FlowFields

PatchMatch by Barnes et al. [BSFG09] constitutes a fast algorithm for computing dense matches. In the original implementation, these descriptors are image patches, but in principle, the algorithm can work with any descriptors. The key idea of the algorithm is to propagate good solutions. The algorithm works as follows:

Propagation Algorithm

1. Initialize each location in the first image with a random correspondence.
2. For each location, check if the correspondences of one of the neighbors are better than the current correspondence; if so, adopt the neighbor's correspondence (propagation step).
3. Search within a small radius of the current correspondence to improve the solution (refinement step). The search radius decreases over time.
4. Repeat from step 2.

The advantages of the algorithm, compared to plain nearest-neighbor matching, are a much faster runtime and implicit regularization through propagation. Note that in principle, the algorithm optimizes the unary matching cost and there is no explicit binary term for the regularization.

FlowFields [BTS15] uses a similar propagation to PatchMatch. However, the algorithm starts with seeds or matches from a k-d tree approximate nearest-neighbor search (see Figure 3.9). The algorithm then uses similar propagation and refinement steps as PatchMatch but additionally introduces propagation across larger distances by working with hierarchies (comparable to an image pyramid) and uses outlier filtering based on forward-backward consistency checks to determine occlusions. The outlier filtering removes most errors arising from the missing regularization (see Figure 3.9j).

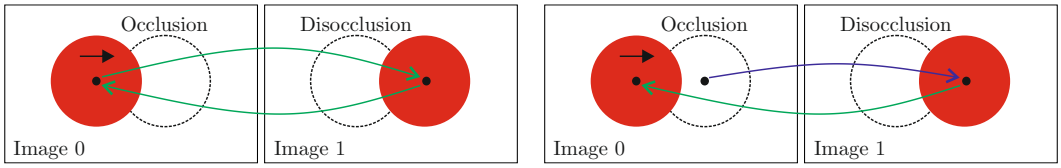
3.5 Occlusion, Depth and Motion Boundary Estimation

(3.5.1) In this work, depth and motion boundary estimation will mainly be described in the case of optical flow. Since disparity is a special case, the term "motion boundary" should be seen interchangeably with "depth boundary" for disparity estimation. In general, if not specifically mentioned otherwise, optical flow principles are also applicable to disparity estimation throughout this work.

Motivation Apart from the fact that occlusion and motion boundary estimation influence the flow estimation itself, they are also important quantities on their own. Referring to the classic work of Black and Jepson [BF00b], "motion boundaries may be useful for navigation, structure from motion, video compression, perceptual organization and object recognition". Moreover, occlusions and motion boundaries are a very strong cue for motion segmentation [ISB18].

Discrete Nature The presence of an occlusion or motion boundary is a discrete phenomenon. Regarding the energy formulation of Equation 3.2.3, occlusions violate the data term and motion boundaries violate the smoothness term. Because of their discrete nature, integrating them into variational frameworks explicitly violates the convex energy assumptions and makes optimization hard.

Most approaches therefore try to estimate motion boundaries and occlusions in a post-processing step. The most common approach to find occlusions is to estimate optical flow in both directions and checking for consistency [ADPS07], as illustrated in Figure 3.10. Other methods use precomputed optical flow and a broad spectrum of visual



(a) Forward and backward flows of a pixel from the foreground object. The point of the object can be determined as non-occluded because the flows are consistent.

(b) Forward and backward flows of a pixel on the static background. In the second image, the pixel is covered by the foreground object, hence the backward flow does not match the forward flow. Therefore, the object can be determined to be occluded.

Figure 3.10: **Flow inconsistency due to occlusion.** The red object moves to the right while the background is static. For the background, forward and backward flows are not consistent, as illustrated in the figure on the left.

features to train a classifier for occlusions [HAB11] or motion boundaries [LSS12a]. Pérez-Rúa et al. [PRCBP16] do not require a dense optical flow field, but motion candidates which are used to determine if a "plausible reconstruction" exists. Motion boundary detectors frequently use image boundary detectors [AMFM11, DZ13] as input in addition to the optical flow.

Chicken-and-Egg Problem

However, since occlusion, motion boundary and correspondence estimation are mutually dependent [HR17, PRCBP16] and the presence of occlusions already negatively influences the correspondence estimation itself, post-processing is suboptimal and leads to unreliable estimates. This is usually referred to as a *chicken-and-egg problem* [HR17, PRCBP16], and one should rather aim for performing the estimation jointly.

Joint Estimation

In discrete methods, such joint estimation can be integrated in different ways. Hur et al. [HR17] establish a joint formulation by estimating forward and backward flows simultaneously as well as by integrating the consistency constraint from Figure 3.10 into the energy function, which is then optimized with BCD. In order to make the optimization feasible, they employ superpixels and a piecewise rigid flow model. Leordeanu et al. [LZS13] train a classifier based on various features, including the current motion estimate, and use it repeatedly during energy minimization of the flow. Notably, this uses occlusion predictions in the optimization but does not constitute a joint optimization. Reasoning about layers is very beneficial to jointly determine occlusions, but it is either overly slow [SSB12] (more than 15 hours per image pair) or also requires precomputed optical flow [SWS+13]. (Moreover according to Statement 2.1.2, it is in general not possible to separate flow fields into layers.) Layered approaches are also used for joint occlusion and disparity estimation [BG05, DYLT07] as well as the mentioned consistency check in the form of uniqueness constraints [GLY95, KZ01b].

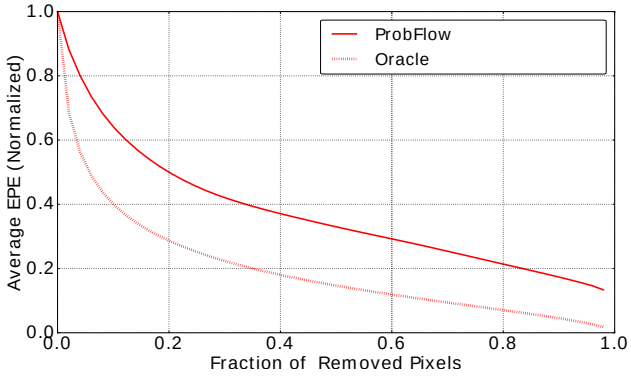


Figure 3.11: **Sparsification plot.** Sparsification plot of ProbFlow [WKR17] for the Sintel train clean dataset. The plot shows the average endpoint error (AEPE) for each fraction of pixels having the highest uncertainties removed. The oracle sparsification shows the lower bound by removing each fraction of pixels ranked by the ground-truth error and corresponds to the best possible ranking. Removing 20 percent of the pixels results in halving the average endpoint error.

3.6 Uncertainty Estimation

Uncertainty and Confidence

Providing a measure of reliability along with predictions is a requirement for real-world applications. Such measures are called *uncertainty* [PSA10, WKR17] in the case of error, and *confidence* [KMG08, BBM09, AHPB13, PTM17, HM12] in the case of reliability estimation. Both measures give equivalent information and an uncertainty estimate can be converted into a confidence by inverting its sign. In special cases, confidence measures are normalized and reflect the probability of a measurement being correct.

Sparsification Plot

The most general evaluation of uncertainties is performed by so-called *sparsification plots* [KMG08, KN11, HM12, AHPB13, WKR17]. The pixels in an image are first ordered from highest to lowest uncertainty. The plot is then created by removing a fraction of the highest-uncertainty pixels and by computing the error of all remaining pixels. The same can be performed with the ground-truth error (which resembles a perfect uncertainty), which is usually referred to as the *oracle*. The quality of the estimated uncertainty map is then determined by how close its sparsification plot is to the oracle. An example is illustrated in Figure 3.11. In this work, we later further simplify the plot by showing the difference between both curves and also introduce an error measure that reflects the area under the difference curve. This will be explained in more detail in Section 11.5.2.

In general, only very few optical flow estimation methods provide uncertainty estimates, while more approaches exist for disparity estimation. The approaches can be differentiated into the categories that will be explained in the following.

3.6.1 Post-hoc uncertainty estimation

Post-hoc methods are decoupled from the flow and disparity estimation process and apply post-processing to estimate uncertainties in a separate step. As such, they ignore information given by the model structure. Very simple methods compute patch similarities [BBM09], use distinctiveness-based measures to evaluate image structure [MT99] or use left-right [EMW04] or forward-backward consistency [ADPS07].

For Optical Flow

All advanced methods usually employ some kind of learning. Kondermann et al. [KKJG07] use PCA to learn typical flow fields within a local neighborhood. They then reconstruct the estimated flow field by using the learned basis to check if the estimated flow field is a "typical" case. One could also say the algorithm compares the estimation to templates that have to be defined with the training data. The reconstruction error serves an uncertainty measure. In their follow-up work [KMG08], Kondermann et al. extend the approach using hypothesis testing on probabilistic motion models. Aodha et al. [AHPB13] follow a simpler approach and define an error threshold on the ground-truth error of estimated flow fields. They then train a classifier from the images and the estimated flow fields to classify as correct or incorrect. The predicted classifier's probability serves as the uncertainty measure. Note that this approach is problematic in case of strong variations in the magnitude of the flow.

For Disparity

For disparity, the state of the art in uncertainty estimation is held by convolutional neural networks [TPBM18]. First outstanding results were achieved by local approaches, taking patches of the disparity map as input and applying a small network to estimate the uncertainty of the center pixel [PTM17, SP16]. Recent work has extended the approach to a fusion of local and global uncertainty estimated with a full encoder-decoder network [TPBM18]. In general, the approaches also follow the classification approach similar to [AHPB13].

3.6.2 Uncertainty Estimation from Model Parameters

Using the final model parameters to infer uncertainty measures was the predominant approach for disparity before CNNs occurred. While this introduces a coupling to the model, the estimation is still performed in a separate step and uncertainties are estimated post-hoc. The simplest confidence measure is the matching cost [EMW04], while more advanced approaches compute local or global properties of the cost curve [Mat92, FVT+93, SS98, ZS01, EMW04, BW06, MAW+07, LAC08]. The intuition is to check how "global" the obtained solution is by analyzing the amount, shape and magnitude of other local minima of the energy function.

In the case of optical flow, Bruhn and Weickert [BW06] used the inverse of the energy functional as a measure of the deviation from the model assumptions, while Kybic and Nieuwenhuis [KN11] performed bootstrap sampling on the data term of an energy-based method in order to obtain meaningful statistics of the flow prediction.

3.6.3 Joint Uncertainty Estimation

Joint estimation of optical flow and disparity requires a fully probabilistic treatment of the problem. Such a formulation is hard to obtain and only very few methods exist. For optical flow, the recent work by Wannenwetsch et al. [WKR17] derived a probabilistic approximation of the posterior of the flow field from the energy functional and computed flow mean and covariance via Bayesian optimization. In this work, we also provide a joint flow and uncertainty estimation approach.

3.7 Summary

The conclusions that can be drawn from the insights of the traditional methods are:

- Variational methods provide a solid framework and are good for objects that are larger than their displacement. Otherwise they revert to precomputed descriptor matches. Occlusions and motion boundaries are not modeled explicitly.
- Discrete optimization is too slow for optical flow directly. Instead, discrete methods usually operate on (potentially erroneous) superpixels or again require good precomputed motion candidates. Due to the lower complexity, good discrete implementations for disparity exist.

(3.7.1)

- How to obtain the best descriptor matches is an open problem, many different heuristics have been proposed.
- Features for descriptor matching should be learned from data.

The problem definition of Chapter 2 outlined the general challenges of optical flow. One can observe that existing methods focus on particular aspects. While some of them may be more relevant for particular application scenarios, none of the existing methods are able to provide a general algorithm that addresses all challenges simultaneously. To the present day, the optical flow problem seems to be too complex to allow for a general algorithm.

Chapter 4

Convolutional Neural Network Basics

This work presents an approach to disparity and optical flow estimation with Convolutional Neural Networks (CNNs). This chapter lays out the basics of CNNs and draws some relationships to the traditional methods from the last chapters. For a detailed reference of CNNs, please refer to [GBC16].

4.1 Basic Concepts

Connections

The concept of a neural network is based on the early idea of feed-forward networks, where neurons are ordered in layers and connections transport information from one layer to the next. The network architecture is handcrafted and describes how the neurons are organized, which connections exist and how signals are computed. In the training process, the importance of these connections (commonly referred to as their *weight*) is learned from a training dataset. The setup of a feed-forward network is illustrated in Figure 4.1.

Neuron

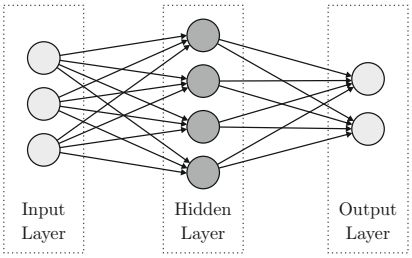
The basic building block of a network is a neuron which takes multiple input connections and combines the information to a single output signal (Figure 4.1b). Let $x_{l,i}$ denote the output of neuron i in layer l. The output of the neuron j in the next layer $l+1$ is then computed as:

$$x_{j,l+1} = f\Big(\big(\sum_{k \in C_j} w_k x_{k,l}\big) + b_j\Big), \tag{4.1.1}$$

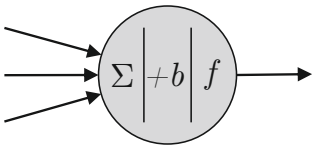
where C_j denotes the input connections for neuron j. The activation function f is most important and distinguishes a neural network from a linear mapping.

Activation Functions

In the beginning, inspired by neuroscience, tanh and the sigmoid functions were used, as depicted in Figures 4.2a and 4.2b. The problem with these nonlinearities is that they suffer from vanishing gradients. Krizevskij et al. [KSH12] introduced a much simpler variant that led to the breakthrough: the Rectified Linear Unit

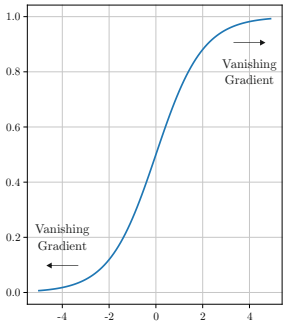


(a) A basic feed-forward network: data is flowing from left to right (sometimes also referred to as bottom to top), and neighboring layers are fully connected.

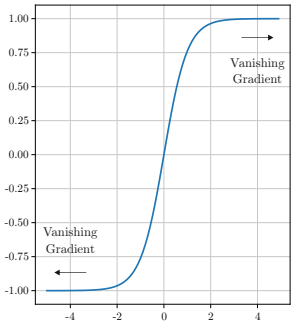


(b) Detailed view of a neuron. Incoming information is summed up, a bias is added and finally, the activation function is applied.

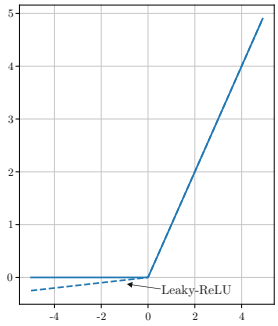
Figure 4.1: **Neural network.**



(a) The sigmoid activation function. For large negative and positive values, the gradient vanishes.



(b) The tanh activation function. For large negative and positive values, the gradient vanishes. For initialization, the advantage over the sigmoid is that at 0, the function takes the value 0.



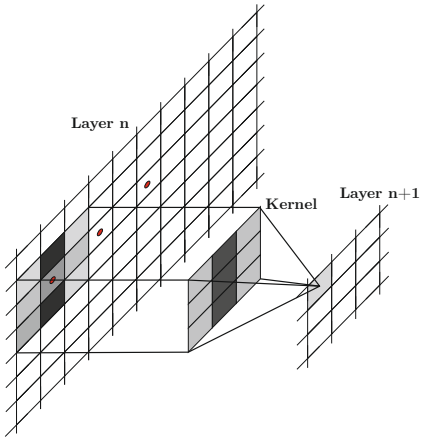
(c) The ReLU activation function. In the leaky variant, a small slope is present in the negative domain.

Figure 4.2: **Activation functions.**

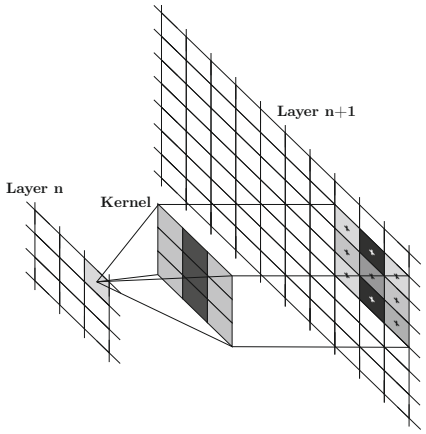
(*ReLU*) shown in Figure 4.2c. In the positive domain, it is linear and does not suffer from the vanishing gradients. Because of the partial linearity, ReLUs preserve many of the properties that make linear models easy to optimize with gradient-based methods [GBC16]. In the negative domain, the ReLU is also linear, but gradients become 0, which can lead to "dead" neurons. However, it is likely that at least for some training sample the value will lie in the positive domain and cause normal gradients, which will eventually balance the use of the neuron. In order to support this further, the *leaky* ReLU was introduced [MHN13] (Figure 4.2c), which is also used in this work. It creates small gradients even in the negative domain.

Convolutional Networks

Applying a fully connected network as in Figure 4.1a to an image would lead to learning different filter masks for each image location and would result in an excessive amount of redundant weights. For an image, spatially invariant features are actually of importance. This leads to the concept of making C_j from Equation 4.1.1 a function



(a) The convolution presents a kernel with a set of weights that is applied to a grid of image locations. The grid can be dense or with a stride, as illustrated by the red points.



(b) The upconvolution performs the opposite operation: for a location of the low-resolution feature map, several different weights are applied and the values are added to the high-resolution feature map (indicated by "+").

Figure 4.3: **Convolution variants.**

of x, y, effectively "convolving" a learnable filter mask with the input. This is referred to as a Convolutional Neural Network (CNN) [LB98] and is depicted in Figure 4.3a. The convolution may optionally be equipped with a stride s that leads to a reduction in spatial dimension (which is equivalent to downsampling the *output* feature map). The same principle may also be applied in the other direction by taking a single input and adding it to several output locations, which leads to a higher resolution feature map and is referred to as *upconvolution* [LSD15, DSB15] (see Figure 4.3b). FlowNet was one of the first works using upconvolutions and applying them on feature maps with additional features from the encoder (see Section 6.1.3).

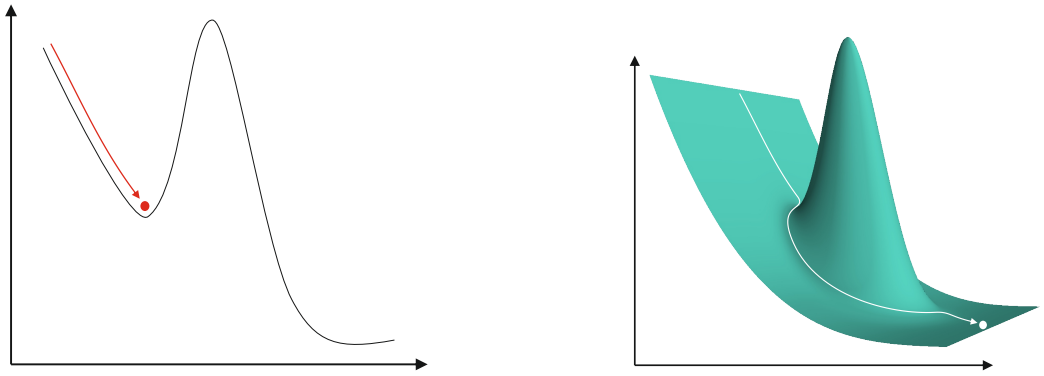
4.2 Training and Convergence

Loss Function

In order to train a neural network, a *loss function* L is applied at the end. This allows for computing gradients to each weight $\frac{\delta L}{\delta w_i}$ respectively by using the chain rule for each weight w_i in the network. Traditionally, machine-learning has avoided the difficulty of general optimization by carefully designing the objective function and constraints to ensure that the optimization problem is convex [GBC16].

Non-convexity

Neural networks have a very large number of parameters and are nonconvex. On the other hand, neural networks can approximate almost any function [Cyb89, Hor91]. It took a long time for the research community to accept that such nonconvex models can be very useful in practice. Due to the memory requirements and computation complexity, simple first-order methods based on gradient descent are used for optimization, such as Stochastic Gradient Descent (SGD) [GBC16]. To achieve a good



(a) In a one-dimensional space, a local minimum is unavoidable.

(b) In a higher-dimensional space, a local minimum in one dimension can be circumvented through the additional dimension.

Figure 4.4: **Local minima in high-dimensional spaces.** The figure shows the same function in both 1D and 2D space. In this example, the 2D space allows for circumventing the local minimum.

generalization, some weight regularization is added to the loss function L. For a local minimum, the necessary condition is that the gradient becomes 0 in all dimensions. An important insight is that this becomes increasingly unlikely the more dimensions exist [CHM+15] (e.g., the presented FlowNet from this work has ∼40M weights). The intuition behind this is illustrated in Figure 4.4. In this particular example, the local minimum can be circumvented by the additional dimension. In general, the more dimensions there are, the less likely it becomes to reach a point where the gradient is 0 in all dimensions.

Local Minima

Choromanska et al. [CHM+15] conjecture that simulated annealing and SGD converge to a band of low critical points, and that all critical points found there are local minima of a high quality, when measured by the test error. They state that in small networks, the probability of reaching a poor local minimum is actually higher and that it is beneficial to increase the network size. They also conjecture that recovering the global minimum becomes harder as the network size increases, but that this is irrelevant in practice because the global minimum is often not desired because of overfitting.

ADAM

Normal gradient descent can lead to strong oscillations. For this reason, the solver is usually extended with a momentum. A further improvement in dealing with plateaus (vanishing gradients) and cliffs (exploding gradients) is to normalize the learning rate. Momentum and learning-rate normalization have led to the popular ADAM solver [KB15], which is also used successfully in this work.

Despite their nonconvexity and the absence of convergence guarantees, CNNs have proven to be among the most powerful methods in almost every computer vision discipline [KSH12, SYLK18, UZU+17, ISB18, EPF14, RDGF16, DLHT16, RFB15].

4.3 Feature Hierarchies

In all machine-learning tasks, the most significant factor is the input representation. The breakthrough of deep learning was the ability to learn a hierarchy of such features, which is expressed by the word "deep". It solves this central problem in representation learning by introducing representations that are expressed in terms of other, simpler representations, enabling the computer to build complex concepts out of simpler ones [GBC16]. An example of such a feature hierarchy is shown in Figure 4.5. It shows a person being represented by simpler concepts, such as corners and contours, which are ultimately defined in terms of edges.

Descriptor Modeling

Statement 2.1.8 explained that because of the aperture problem, pixel compounds need to be considered and the required size of such a compound is unknown. Section 3.1 mentioned that descriptors learned with CNNs outperform all engineered methods. CNNs with a stack of several layers from Equation 4.1.1 can actually implement HOG descriptors (derivatives, direction projection, binning and accumulation can all be expressed by appropriately configured convolutions) and can also learn biologically motivated filters, such as Gabor filters [LNSC14]. Thus, traditional descriptors are a special case of what CNNs can implement, and CNNs can possibly converge to a combination of those or even to an unknown superior solution.

(4.3.1) Beyond features, the remaining question is if CNNs can also learn the best heuristics for correspondence and regularization algorithms. This is the central part of this work and will be presented in the following chapters.

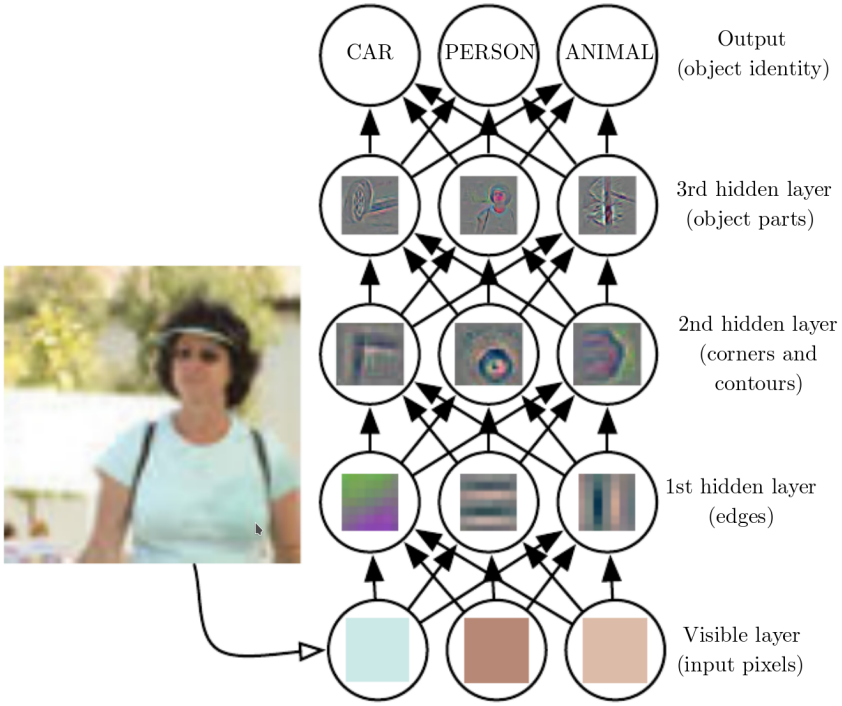


Figure 4.5: **Feature hierarchy learned by a CNN.** The figure shows the different levels of abstraction learned by a deep network. The bottom layers take pixels as input. First-level features then extract lines and edges. Further layers aggregate features to contours and parts. The final layers allow for deciding upon the whole object's class. Source: [GBC16].

Chapter 5

Training Data

Training data was published in the works [DFI+15], [MIH+16] and [IMS+17]. Generating the data was the contribution of other authors. The author of this thesis contributed to selecting the correct priors and proposed the priors for the ChairsSD-Hom dataset.

Modalities

Section 2.4.3 has already explained that optical flow is a secondary feature and cannot be directly captured by a sensor. Obtaining large amounts of accurate training data for optical flow is therefore far from easy. For this reason, we revert to generating such data synthetically. In general, we generate the following modalities for each dataset:

- images,
- forward and backward optical flow,
- forward and backward occlusions and
- forward and backward motion boundaries.

For the scene flow data, we generate the above for the left and the right camera and additionally provide:

- left-right and right-left disparities,
- left-right and right-left disparity occlusions,
- left-right and right-left disparity changes and
- left and right depth boundaries.

Generalization

We also provide the scene flow data across more than two frames. The datasets presented in the following will be rendered from 2D and 3D models but are in general far away from realistic scenarios. For a detailed ablation study of dataset aspects please refer to [MIF+18]. Interestingly, the rest of this work will show that the task

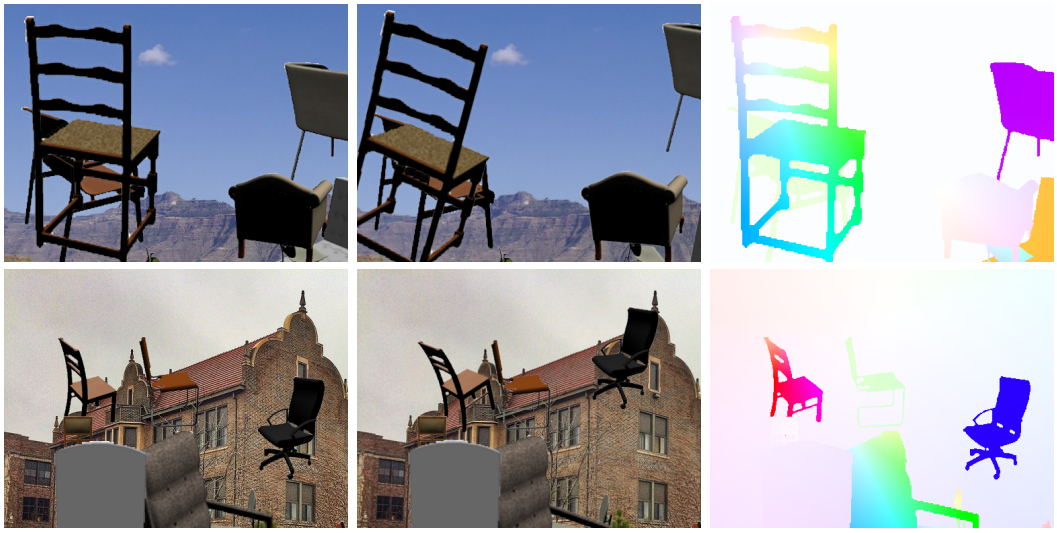


Figure 5.1: **Two examples from the FlyingChairs dataset.** We generate simplistic training data by placing chairs on front of a background and by applying randomly sampled affine transformations. Each row shows the two images and the visualized flow field.

of correspondence estimation can be learned and generalizes very well from such unrealistic data. The nature of correspondence estimation is in general very different from classification: correspondence estimation requires finding similar objects in both images. Notably, this does not require knowing about all possible object types, but only about the general appearance of a possible object.

5.1 FlyingChairs

Existing datasets, such as Sintel [BWSB12] and KITTI [GLU12, MG15], cannot supply the required large amount of data necessary to train a CNN for optical flow. In order to create this amount of data, we take a very simple approach and generate a dataset that we name *FlyingChairs*. This dataset is generated from 2D images exclusively and only valid for two-frame optical flow.

Objects As backgrounds, we use 964 images with a resolution of $1,024 \times 768$ pixels that were downloaded from Flickr. As foreground objects, we use 809 chair models from the dataset of Aubry et al. [AME+14], each rendered from 62 views: 31 azimuth angles and 2 elevation angles. In order to generate the first image in an image pair, we take a background image and randomly position a random set of chairs on top. Examples are shown in Figure 5.1. The number of the chairs is sampled uniformly from 16 to 24, the types and viewpoints of the chairs are sampled uniformly and the locations of the chairs are sampled uniformly from the whole image. The sizes of the chairs (in pixels) are sampled from a Gaussian with mean 200 and standard deviation 200, and then clamped between 50 and 640.

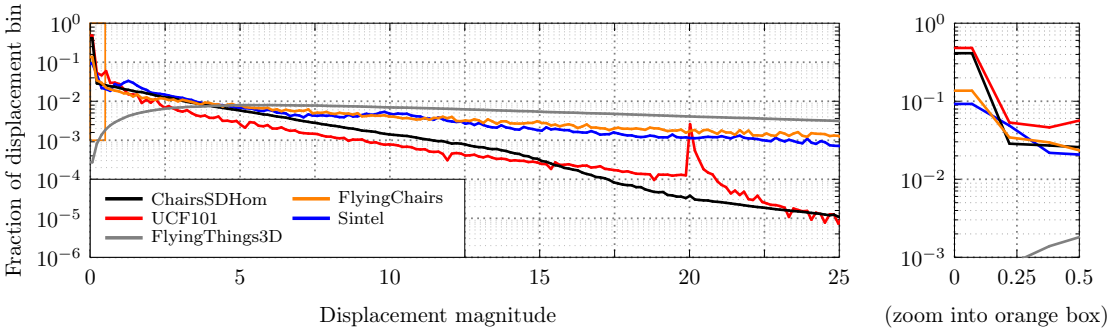


Figure 5.2: **Displacement magnitude histograms.** Left: histogram of displacement magnitudes of different datasets. The y-axis is logarithmic. Right: zoomed view for very small displacements. The FlyingChairs dataset very closely follows the Sintel dataset, while the ChairsSDHom dataset is close to UCF101. FlyingThings3D has few small displacements and for larger displacements also follows Sintel and FlyingChairs. The FlyingThings3D histogram appears smoother because it contains more raw pixel data as well as due to its randomization of 6-DOF camera motion. The spike in UCF101 comes from clipping x- and y-ranges to 20px respectively for storage (note that this still allows non-axis-aligned flows to have magnitudes larger than 20px).

TRANSFORMATIONS

To generate the second image in a pair and the flow field, we apply random transformations to the chairs and the background. Each of these transformations is a composition of zooming, rotation and translation. The parameters to be sampled are the zoom coefficient, the rotation angle and the translation vector. We aim at roughly matching the displacement distribution of Sintel, as shown in Fig. 5.2. For details of the parameter distributions, please refer to the supplemental material of [DFI+15].

Given the transformation parameters, it is straightforward to generate the second image in the pair as well as the flow field and the occlusion map; the same can also be done in the backward direction. We finally cut each image into four quarters, resulting in four image pairs of 512×384 pixels each. We use this procedure to generate $22,872$ image pairs in total. Note that this size is chosen arbitrarily and could be larger in principle. In summary:

(5.1.1) FlyingChairs is a simplistic dataset generated only from 2D data and only with 2D transformations.

5.2 ChairsSDHom

As one example of a real-world dataset, we examine the action recognition dataset UCF101 [SZS13]. We compute optical flow by using LDOF [BM11] and compare the flow magnitude distribution to other datasets, as shown in Figure 5.2. While FlyingChairs is similar to Sintel, UCF101 is fundamentally different and contains significantly more small displacements.

Figure 5.3: **Example images from the ChairsSDHom dataset.** Motivated by UCF101 [SZS13], we create a small displacement version of FlyingChairs and add homogeneous backgrounds.

Homogeneous Backgrounds

In addition, the recorded scenes of UCF101 often contain a homogeneous background. We thus add scenes with weakly textured backgrounds that are monochrome or contain very subtle color gradients (see Figure 5.3). Such monotonous backgrounds are not unusual in natural videos, but never appear in FlyingChairs. As described in Section 2.1.4, homogeneous areas do not allow for correspondence estimation, and in principle any motion is possible. The most plausible prior is that such regions do not move. Hence, in order to introduce a meaningful prior, we keep the homogeneous background images fixed.

With some minor differences in the technical implementation, the dataset itself is generated in the same manner as FlyingChairs but with much smaller displacements. We name the dataset *ChairsSDHom* (SD = Small Displacements, Hom = Homogeneous backgrounds).

5.3 FlyingThings3D

FlyingChairs and ChairsSDHom only apply 2D affine transformations to objects. As explained in Section 2.1.2 and Figure 2.4, general 3D motion of objects cannot be described by affine transformations. Furthermore, scene flow can also only be inferred if the underlying data is truly 3D.

3D Modeling

In order to overcome these limitations, the FlyingThings3D dataset was created which contains the complete ground-truth scene flow in forward and backward direction. We use the open-source 3D creation suite Blender to animate a large number of objects with complex motions. The resulting information is complete even in occluded regions since the render engine always has full knowledge about all scene points.

Objects

The base of each scene is a large textured ground plane. We generated 200 static background objects with shapes that were randomly chosen from cuboids and cylinders. Each object was randomly scaled, rotated, textured and then placed on the ground plane. In order to populate the scene, we downloaded $35,927$ detailed 3D models from Stanford's ShapeNet [SCH15][1] database. From these, we assembled a training set of $32,872$ and a test set of $3,055$ models. Furthermore, the model categories were split disjointly among the training and the test set.

[1] `http://shapenet.cs.stanford.edu/`

Transformations

We sampled between five and twenty random objects from this object collection and randomly textured every material of every object. Each ShapeNet object is translated and rotated along a smooth 3D trajectory among multiple frames. The camera itself is also animated. The texture collection is a combination of procedural images created with ImageMagick[2], landscape and cityscape photographs from Flickr[3], and texture-style photographs from Image*After[4]. Like the 3D models also the textures were split into disjoint training and test sets.

Rendering

Given the intrinsic camera parameters (focal length, principal point) and the render settings (image size, virtual sensor size and format), we project the 3D motion vector of each pixel into a 2D pixel motion vector in the image plane. Depth is directly retrieved from a pixel's 3D position and converted to disparity by using the known configuration of the virtual stereo rig. We compute the disparity change from the depth component of the 3D motion vector. Examples are shown in Fig. 5.4.

Like the Sintel dataset, this dataset also includes two distinct versions of every image: the *clean pass* shows colors, textures and scene lighting but no image degradations, while the *final pass* additionally includes postprocessing effects such as simulated depth-of-field blur, motion blur, sunlight glare and gamma-curve manipulation.

[2] `http://www.imagemagick.org/script/index.php`

[3] `https://www.flickr.com/` Noncommercial public license. We used the code framework by Hays and Efros [HE08]

[4] `http://www.imageafter.com/textures.php`

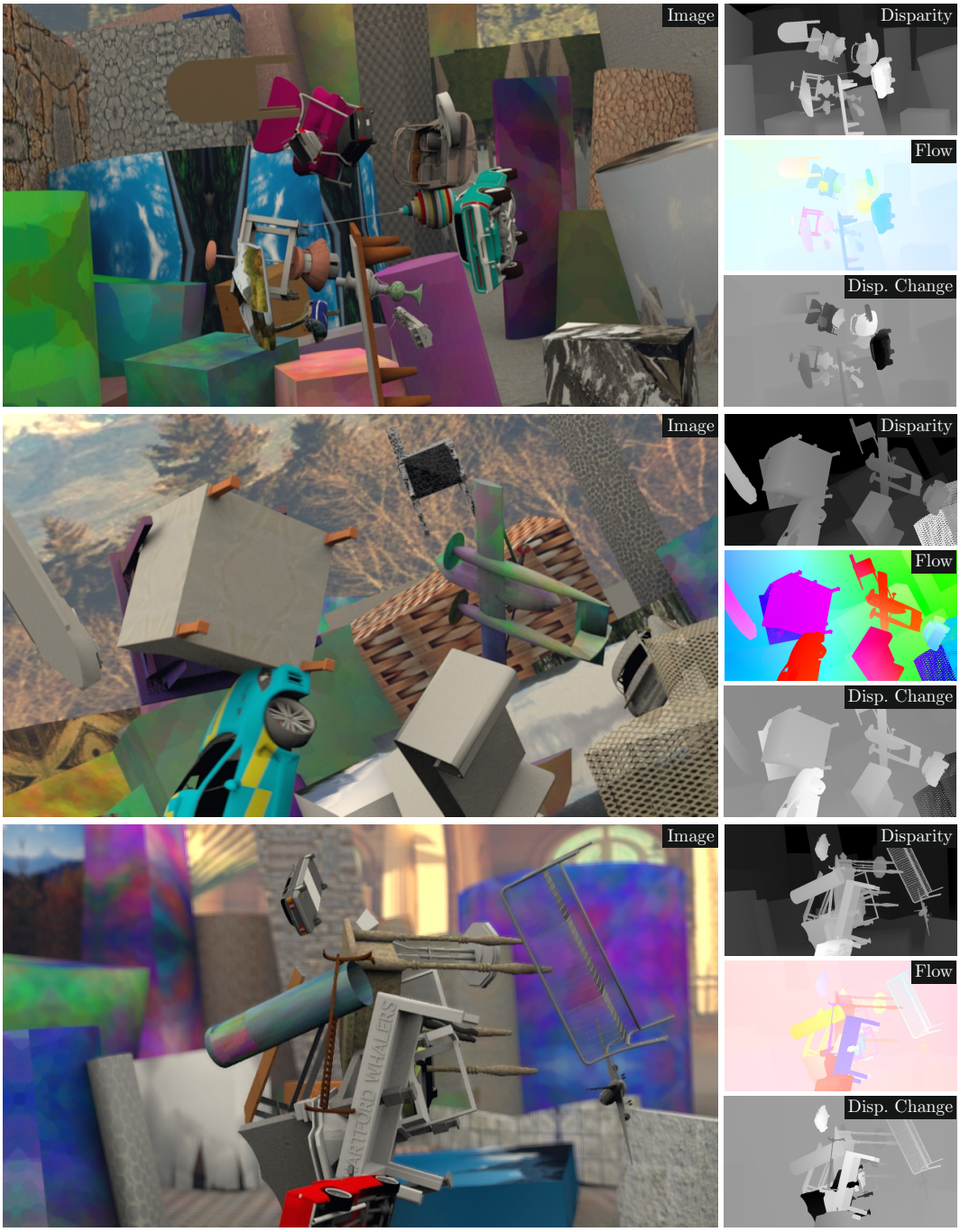


Figure 5.4: **Examples from the FlyingThings3D dataset.** The figure shows left images, left-right disparity, left flow and left-right disparity change (corresponding to motion in camera direction).

Chapter 6

FlowNet

FlowNet, published in [DFI+15], was a joint work together with Alexey Dosovitskiy and Philipp Fischer. The author of this thesis contributed with his expertise in optical flow, to the concept of FlowNetC and the design of the correlation layer. The first work did not train the networks equally. The author showed that FlowNetC outperforms FlowNetS and proposed the dataset schedules which boosted boosted the performance (published in [IMS+17]). Ablation studies from this chapter are also the author's contributions. In [DFI+15], the author furthermore contributed the variational refinement.

End-to-end CNNs

Past work has already shown that CNNs are good at computing discriminative features for matching. A key contribution of this work is showing that CNNs are capable of solving the whole matching and regularization task end-to-end, as illustrated in Figure 6.1:

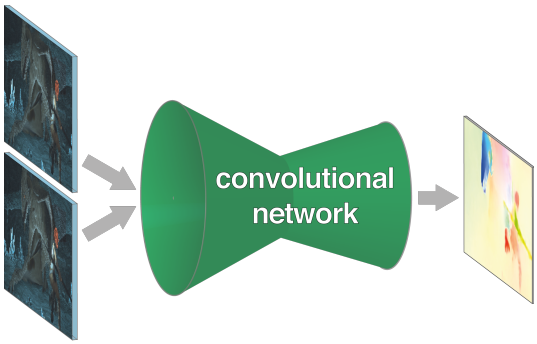


Figure 6.1: **FlowNet.** A key contribution of this thesis are neural networks that solve the complete optical flow task end-to-end, including feature extraction, matching and regularization. The information is first spatially compressed to a more abstract representation in a contractive part and then refined back to full resolution in an expanding part.

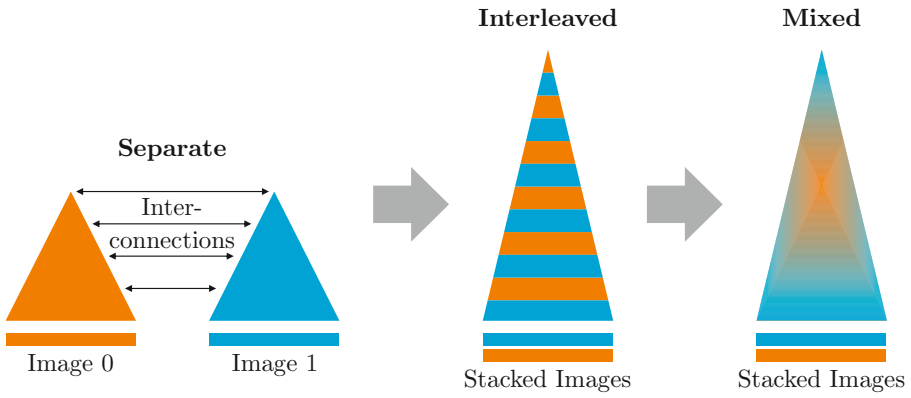


Figure 6.2: **Siamese networks and stacked images.** The left side shows a Siamese architecture with two separate networks on two images. Interconnections need to be added in order to be able to determine correspondences. It is possible to integrate both networks into a single interleaved identical one (shown in the middle). This furthermore allows for closely entangling the two networks and for making arbitrary connections between them (shown on the right), leading to a single encoder that takes the stacked images as input.

Feature Hierarchies

We follow the concept of learning feature hierarchies and going to a more abstract representation. As an example, consider a moving person. The lowest-level features are lines and edges which can then be aggregated in a hierarchy. E.g., composition of lines and edges can lead to a finger, this in turn can lead to a hand, an arm and the whole person. This is similar to the concept of feature hierarchies for classification described in Section 4.3.

Traditional methods use downsampling to obtain more abstract representations. In order to be able to consider the whole person as an entity, a coarse resolution is required. Unfortunately, the coarse resolution results in a loss of information. CNNs allow for obtaining more abstract representations by reducing the spatial dimension while increasing the feature dimension, which allows for retaining information about the object. Such a feature hierarchy enables to construct the required pixel compounds described in Section 2.1.2 with different granularities. To this end, we propose two encoder-decoder network architectures that will now be described in more detail. The first one is very generic while the second one is more tailored to the correspondence estimation task.

6.1 Network Architectures

6.1.1 FlowNetS Encoder

The required compound sizes and the level of abstraction depend on the data itself. This means that when building a hierarchy, the level at which to check for correspondences is unknown.

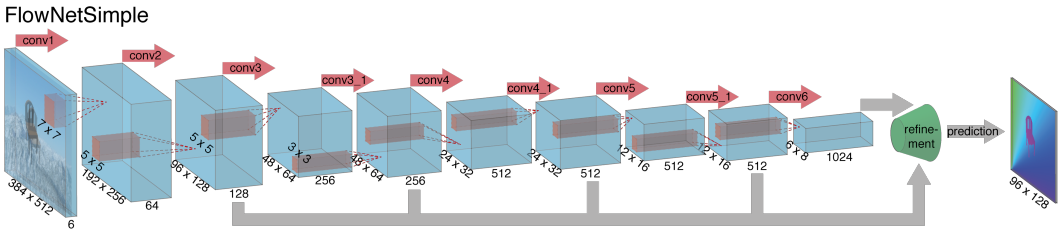


Figure 6.3: **FlowNetS architecture.** The figure shows the FlowNetS encoder architecture, which consists of nine convolutional layers. Six of them have stride of 2. This results in a reduction of the spatial dimension by $\frac{1}{64}$. At the same time, the feature dimension is increased to $1,024$. Filter sizes are 7×7, 5×5 and 3×3 for all subsequent convolutions. The decoder is indicated by a green funnel (shown in detail in Figure 6.5).

Interleaved Encoder

The straightforward approach is to construct a Siamese network and to introduce interconnections at all possible levels in order to account for the unknown level at which to match. However, such a Siamese network can also be represented as a single interleaved network. Introducing arbitrary connections then leads to a single encoder that takes the stacked images as input. This is illustrated in Figure 6.2 (see previous page). Such an encoder still has the ability to contain two disjoint Siamese parts while it can also learn to arbitrarily entangle them.

FlowNetS

We use such an approach for the first variant of our network, which we call FlowNetSimple (*FlowNetS*). Where and how exactly the matching and regularization happens in this case is left entirely to the learning process. The architecture of the resulting encoder is shown in Figure 6.3.

6.1.2 FlowNetC Encoder

In the second variant, we closely follow the traditional approaches by designing a network that performs the feature extraction, matching and regularization steps explicitly. This is shown in Figure 6.4.

FlowNetC

In the first step, three levels of convolutions are applied in order to extract features on both images in a Siamese fashion. For the matching, we introduce a special *correlation* layer which performs multiplicative comparisons between two feature maps. Due to the special layer, we name this network architecture FlowNetCorr (*FlowNetC*).

Correlation

Given two multi-channel feature maps $\mathbf{f}_1, \mathbf{f}_2 : \mathbb{R}^2 \to \mathbb{R}^k$ (with w, h, and k being their width, height and number of channels), our correlation layer lets the network compare each feature vector from $\mathbf{f}_1$ with each feature vector from $\mathbf{f}_2$. The correlation of two feature vectors at locations $\mathbf{x}_1$ and $\mathbf{x}_2$ is then defined as:

$$c(\mathbf{x}_1, \mathbf{x}_2) = \langle \mathbf{f}_1(\mathbf{x}_1), \mathbf{f}_2(\mathbf{x}_2) \rangle \quad . \tag{6.1.1}$$

Note that Equation 6.1.1 is identical to a 1×1 convolution in neural networks, but instead of convolving data with a filter, it does so with other data. For this reason, it allows for normal backpropagation but has no trainable weights. Comparing all patch

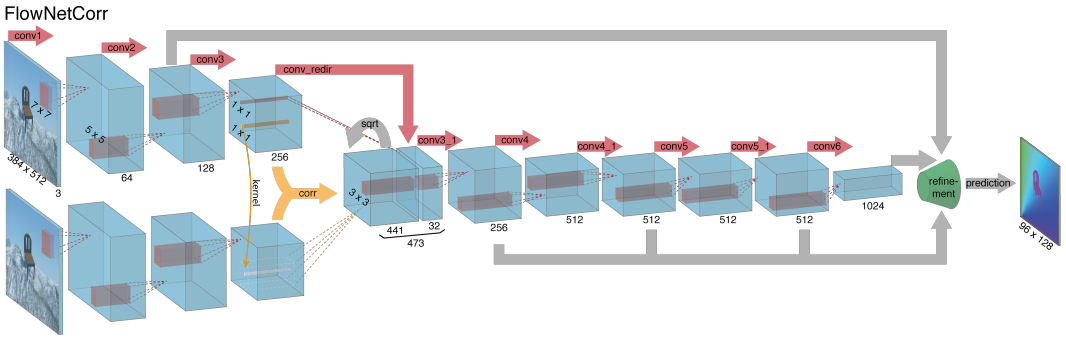


Figure 6.4: **FlowNetC architecture.** The FlowNetC architecture starts with a Siamese part that extracts features on both images. The features are then passed to a custom correlation layer that computes correlations within local neighborhoods. The correlation result is then passed through an encoder-decoder architecture for regularization. The decoder is indicated by a green funnel (shown is in detail in Figure 6.5).

combinations involves $w^2 \cdot h^2$ such computations, yields a large result and makes efficient forward and backward passes intractable. Thus, for computational reasons, we limit the maximum displacement for comparisons to a local neighborhood and also introduce striding in both feature maps.

Given a maximum displacement η, we construct displacement vectors $\mathbf{d} \in [-\eta, +\eta] \times [-\eta, +\eta]$ and compute $c(\mathbf{x}_1, \mathbf{x}_1 + \mathbf{d})$ for a given $\mathbf{x}_1$ for each possible $\mathbf{d}$. We use strides s_1 to quantize $\mathbf{x}_1$ globally and s_d to quantize $\mathbf{d}$ within the neighborhood centered around $\mathbf{x}_1$. For each feature location $\mathbf{x}_1$, this comes to a two-dimensional result according to the possible values of $\mathbf{d}$. In order to avoid a four-dimensional space, we organize the relative displacements in channels in practice, i.e. we reshape the result to a one-dimensional vector representing the output feature vector of the operation.

Regularization

The last step is then to perform regularization on the correlation result. In order to achieve this, we add an encoder-decoder network on top (see Figures 6.4 and 6.5). In the encoder, we use six more convolutions to obtain the same final feature map as for FlowNetS. Notably, in this architecture, we define where and how the matching happens and what the largest feature compound size is (as the receptive field before the convolution). This introduces engineered priors and does not preserve the ability of a general network to learn the optimal feature hierarchies. However, the task of matching larger compounds as a whole can still be accomplished by the regularization after the correlation.

6.1.3 Decoder

In order to bring the abstract representation back to full resolution, we use the decoder depicted in Figure 6.5 (see next page). For increasing the resolution we use upconvolutions as described in Section 4.1. Each step increases the resolution by a factor of 2. We repeat this four times, resulting in a predicted flow for which the resolution is still four times smaller than the input. Using this resolution and

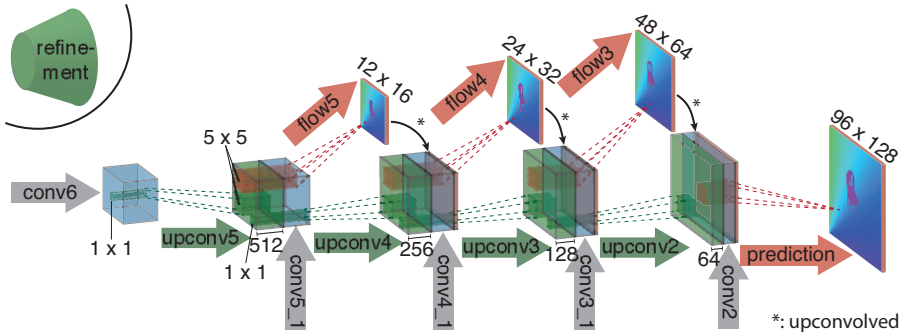


Figure 6.5: **Decoder architecture.** The decoder uses upconvolutions to upsample previous feature maps. Flow fields are predicted at each resolution (deep supervision), upsampled and concatenated to the features together with those from the encoder. These steps are repeated so as to obtain a high-resolution flow field.

performing a final bilinear upsampling step results in best speed/accuracy trade-off. Optionally, one may also add two more upconvolutions to obtain the full resolution in more detail. We also make use of the concept of *deep supervision* to predict a downsampled flow field at each resolution.

Skip Connections

Notably, we introduce *skip connections* from each part of the encoder by concatenating with its features. This way, we preserve both the high-level information passed from coarser feature maps and fine local information provided in lower-layer feature maps. The decoder can use this information to accurately spread estimated motion to object boundaries. Furthermore, information at a certain level can also pass directly from the encoder to the decoder by a skip connection (e.g. a small object).

6.2 Analysis

6.2.1 Training Details

In order to train the networks, we use the FlyingChairs and FlyingThings3D datasets (Chapter 5) and apply the endpoint error as a loss function (Equation 2.4.1). We fix the momentum parameters as recommended in [KB15] to $\beta_1 = 0.9$ and $\beta_2 = 0.999$. For regularization, we add an L2 loss on the weights of the network with a coefficient of 0.0004. Since in a certain way, every pixel is a training sample, we use fairly small mini-batches of eight image pairs for FlyingChairs and four image pairs for FlyingThings3D. Note that the resolution of FlyingThings3D is higher and the smaller batch size results in approximately the same amount of pixels.

We start with a learning rate of $\lambda = 1e\text{–}4$ and then divide it by 2 every 100k iterations after the first 300k. In total, this amounts to 600k iterations. We name this schedule S_{short}. Further learning rate schedules will be presented in Section 6.2.8. The results of training the networks with different solver configurations are shown in Table 6.1.

Solver	FlowNetS	FlowNetC
SGD [GBC16]	7.76	8.85
SGD+momentum [GBC16]	7.36	7.73
ADAM [KB15]	**6.80**	**7.00**

Table 6.1: **Solver results.** The table shows results on Sintel train clean after training FlowNetS and FlowNetC for 600k iterations with different solver configurations. In comparison, LDOF [BM11] achieves an EPE of 4.29 on the same dataset. The results show that our proposed CNNs are initially capable of estimating optical flow and the ADAM solver performs best.

(6.2.1) The first observation from Table 6.1 is that optical flow estimation is possible with the proposed CNNs. The best results are achieved with the ADAM solver.

Interestingly, both network variants can be trained with plain SGD. Adding momentum improves the results. Further normalizing the learning rate with the ADAM solver performs best in both cases and will be used in subsequent experiments. For complete details of the network architecture and solver configuration, the reader is referred to [DFI+15]. Further improvements to the training procedure will be presented in the following.

6.2.2 Augmentation

A widely used strategy to improve generalization of neural networks is data augmentation [KSH12, EPF14]. Even though the presented datasets are fairly large, we find that using augmentations significantly increases the performance and prevents overfitting. We include two types of augmentation:

Geometric Augmentation

As **geometric augmentation**, we apply translation, rotation and scaling with different coefficients in x- and y-direction. Because we want to increase the variety not only of images but also of flow fields, we apply the same strong geometric transformation to both images of a pair but additionally a smaller relative transformation between the two images. We adapt the flow field accordingly by applying the per-image augmentations to it as well.

Color Augmentation

As **color augmentation**, we apply changes in brightness, contrast, gamma and color, and we add Gaussian noise.

For complete details of the augmentation parameters, the reader is referred to [DFI+15]. We show some examples of the applied augmentations in Figure 6.6 (see next page) and provide an ablation study in Table 6.2 (see next page). We observe that:

(6.2.2) Color augmentation does not help for FlowNetS, but improves FlowNetC results by approx. 17%. Notably, geometric augmentation decreases the error by approx. 33% in both cases in comparison to using no augmentation. Combining both augmentations gives the best results, thus we use this configuration for subsequent experiments. In total, the effect of augmentation is strongest for FlowNetC, reducing the error by approx. 46%. Note that the result for FlowNetC with augmentation (3.77) now outperforms the traditional method LDOF [BM11] (4.29) by 0.52.

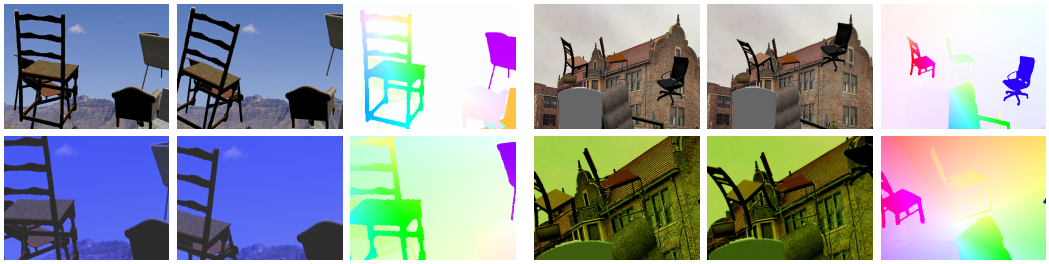


Figure 6.6: **Data augmentation examples.** The top row shows image pairs and flow visualizations of the original data. The bottom row shows the data after the randomized online augmentation.

Augmentation Type	FlowNetS	FlowNetC
None	6.80	7.00
Color	6.91	5.81
Geometric	4.57	3.88
Geometric+Color	**4.45**	**3.77**

Table 6.2: **Augmentation results.** The table shows results on Sintel train clean after training FlowNetS and FlowNetC for 600k iterations with different augmentation configurations. Adding both types of augmentation yields the best results.

Losses	FlowNetS	FlowNetC
Level 2	4.67	3.95
Levels 6,5,4,3,2	**4.45**	**3.77**

Table 6.3: **Deep supervision results.** The table shows results on Sintel train clean for networks trained with and without deep supervision. The deep supervision is not required but increases performance slightly.

6.2.3 Deep supervision

Next, we perform an ablation study of the deep supervision losses:

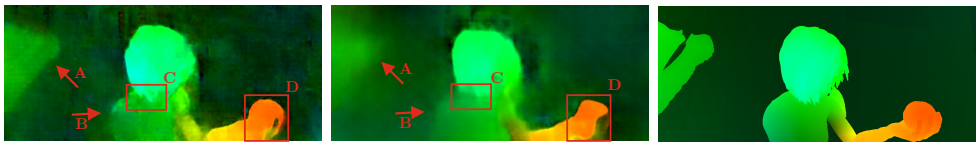
(6.2.3) The results from Table 6.3 show that the networks can also well be trained with only a final loss in the end, with deep supervision not being mandatory. Adding deep supervision improves results slightly.

6.2.4 Skip Connections and Bottleneck

The skip connections are an essential part of the network architecture. In principle, any middle block between the same encoder and decoder levels can be bypassed through a skip connection. This can also be seen as creating an ensemble within a

Skip connections	FlowNetS	FlowNetC
No	4.70	3.87
Yes, no backprop.	4.89	3.86
Yes	**4.45**	**3.77**

Table 6.4: **Results for using skip connections.** The table shows results on Sintel train clean for different configurations of skip connections. Adding skip connections with backpropagation gives the best results.



(a) Skip connections (b) No skip connections (c) Ground truth

Figure 6.7: **Effect of skip connections.** The figure shows results of a FlowNetS trained with and without skip connections by using the Middlebury visualization. Training networks without skip connections is still possible but leads to blurring (A and B) and missing or incorrectly estimated details (C and D).

single network. In FlowNetS and FlowNetC, the skip connections can be used to transport image features and finer-grained correspondence information to the decoder. The image features are needed in order to spatially propagate coarse flows from the bottleneck correctly to object boundaries. Note that in the coarse representation of the bottleneck, features allow for encoding multiple flows per location. This information needs to be spread out when going to finer resolutions until eventually selecting a single flow per pixel.

(6.2.4) The experiments from Table 6.4 show that it is also possible to train networks without skip connections and that results are only slightly worse.

Note that in this case, all image information has to pass through the bottleneck. As the bottleneck's capacity is limited, one would expect blurred edges and missing details. These effects are visible in Figure 6.7.

Noise However, when adding the skip connections, one can generally observe that predictions become more noisy (see Figure 6.7a). Assumingly, this comes from the network trying to estimate optical flow from image features instead of from correspondence information. Disabling the gradients through the skip connections does not mitigate the effect (see Table 6.4). Reducing the noise will be further discussed in Section 6.2.5.

Details In order to further investigate the information flowing through the network, we perform experiments trying to reconstruct the image from the bottleneck. Figure 6.8 (see next page) shows that quite detailed image information is generally present. As expected, the information about the first image is generally sharper than for the second image (this arises from the fact that edges of the flow field correspond to edges of the first image), and using an initial network without skip connections leads to more details

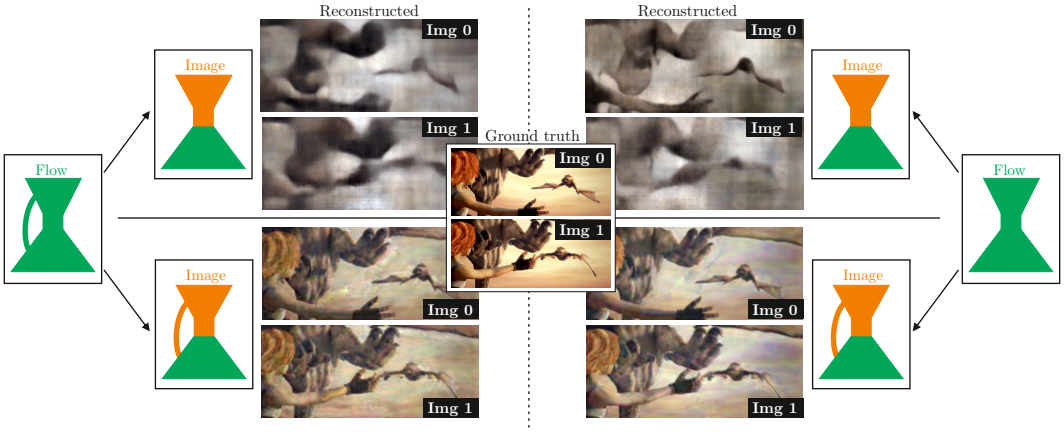


Figure 6.8: **Image reconstruction from bottleneck.** The experiments are based on taking a trained FlowNetS with skip connections (left) or without skip connections (right) and training a new decoder (orange) while leaving the initial part of the network fixed (green). In general, one can observe that some image features are present in the bottleneck (top) but do not contain color information. As expected, using an initial network without skip connections (right) propagates finer details through the bottleneck.

being propagated through the bottleneck. Note that although the bottleneck has a very coarse resolution ($\frac{1}{64}$th of the original), it in general still contains somewhat detailed information (top left of Figure 6.8).

6.2.5 Value Normalization

For training CNNs, it is common to normalize input and output data to a common range of $[-1, 1]$ [KSH12, SZ14b]. This improves convergence and numerical stability. In the FlowNet case, normalizing the optical flow ground truth means to divide the values by approximately 20.

(6.2.5) We find that using value normalization does not influence the EPE significantly but produces much more noise than without normalization, as illustrated in Figure 6.9.

Since there are no different types of output that require to be normalized to a common range and the numerical modification by a factor of 20 is not too large, the optimization should in principle not be affected by scaling the output values.

However, FlowNet actually reinserts the predicted values into the network during the coarse-to-fine refinement in the decoder. The previously predicted optical flow field is upsampled and concatenated with the upsampled features as well as the encoder skip connections features (see Figure 6.5). The next convolution operates on this mixture of data. Scaling the flow values up leads to respectively smaller weights in the next convolution. We conjecture that due to weight decay, the smaller weight values allow the next convolution to put the focus on the upsampled flow values more easily and leads to a clean coarse-to-fine refinement instead of trying to infer

(a) Values scaled by $\frac{1}{20}$ (b) Unscaled values (c) Ground truth

Figure 6.9: **Effect of ground truth normalization.** The figure shows two different networks by using the Middlebury visualization. The one on the left is trained with ground-truth values scaled by $\frac{1}{20}$ while the other one is trained without scaling. Notably, the downscaling introduces noise, and removing it leads to much better predictions.

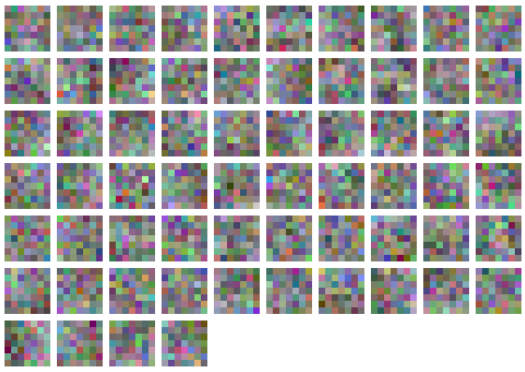


Figure 6.10: **Visualization of first-layer filters of FlowNetC.** The filters have only little structure while the network can use them very well to estimate optical flow.

flow values from the skip connections. The latter can hardly give flow estimations (only for displacements inside the receptive field), and trying to infer flow from image features naturally leads to noise produced by the optimization.

6.2.6 Learned Features

The first layers of CNNs operate directly on images, hence it is possible to visualize their filter masks as color patches. The visualization of the first-layer filtermasks for FlowNetC are shown in Figure 6.10. While other works have reported Gabor-like filters [ZF14], it is interesting to see that FlowNetC uses filters with very little structure. For FlowNetS, the case is similar, and training the networks longer leads to a similar solution, too. This indicates that the network converges to a solution different from what humans would engineer.

In contrast, the filters that are applied to the output of the correlation layer have a very visible structure, as shown in Fig. 6.11 (see next page). Different filters are selective for different flow directions and magnitudes, resembling different motion patterns.

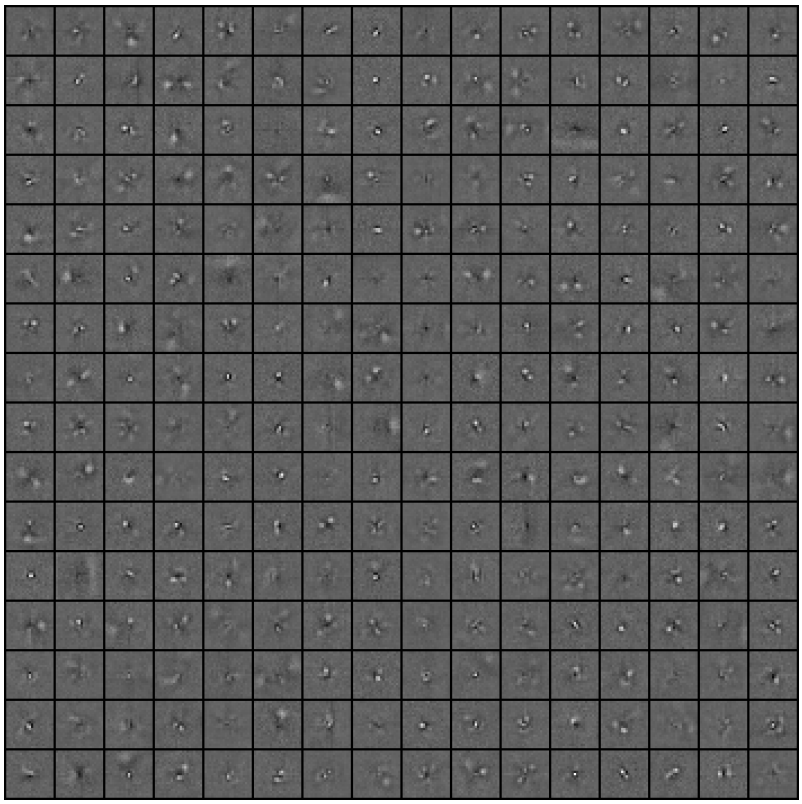


Figure 6.11: **Visualization of features after correlation.** Visualization of filters applied on top of the correlation layer in FlowNetC. There are 256 filters, and for each of them, we show the weights shaped as a 21×21 pixels patch, where each pixel corresponds to a displacement vector. The center pixel of each patch corresponds to zero displacement. The filters favor interesting displacement patterns.

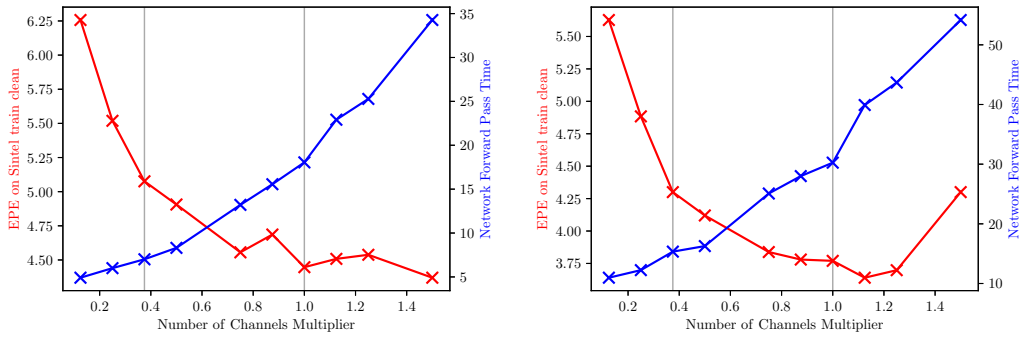
6.2.7 Network size

We experiment with the network size by taking only a fraction of the number of channels for every layer in the network. Figure 6.12 shows the network accuracy and runtime for different network sizes. One can observe that even very small networks are still able to estimate good optical flow while being much faster.

(6.2.6) In order to obtain small and fast variants, we use a fraction of 3/8 of the original number of channels as a good trade-off between speed and accuracy.

6.2.8 Datasets and schedules

Well-selected training data is crucial for the success of supervised training. The past sections have shown that it is possible to train a network on FlyingChairs and to obtain a performance on Sintel surpassing LDOF [BM11].



(a) FlowNetS (b) FlowNetC

Figure 6.12: **Performance over network size.** The figures show accuracy and runtime depending on the network width. The multiplier 1 corresponds to the width of the original FlowNet architecture. Wider networks do not improve the accuracy significantly. For fast execution times, a factor of $\frac{3}{8}$ is a good choice. Timings are taken from an Nvidia GTX 1080.

It is interesting to see by itself that a CNN can estimate optical flow and learns to do so by using the very simplistic chairs dataset. This shows that the concept of correspondence can be learned independently of the actual objects as long as the general dataset statistics are similar to the test data (these include real-world backgrounds and object sizes). The question seems reasonable if a more sophisticated dataset can further improve the performance. For this reason, we introduced FlyingThings3D in Section 5.3. This dataset contains a larger variety of objects, is rendered from true 3D motion and contains realistic lighting effects.

We perform experiments on the different datasets with different training schedules and fine-tuning. To this extent, we introduce a schedule S_{long} that is a lengthened version of the original schedule S_{short} and a schedule S_{fine} for fine-tuning. The schedules are illustrated in Figure 6.13 (see next page). In later chapters, we also use schedules shortened by a constant factor. We denote this by a division, e.g., S_{short} shortened by a factor of 2 is denoted $S_{short}/2$.

Results of training networks on the different datasets and with the different schedules are shown in Table 6.5 (see next page). The first observation is that using longer schedules improves the results. Extending the schedule even further on the same dataset does not lead to significant improvements any more, indicating that networks trained with S_{long} are converged quite well. Surprisingly, results are worse when the networks are trained on FlyingThings3D rather than on FlyingChairs, despite FlyingThings3D being more sophisticated and having similar statistics.

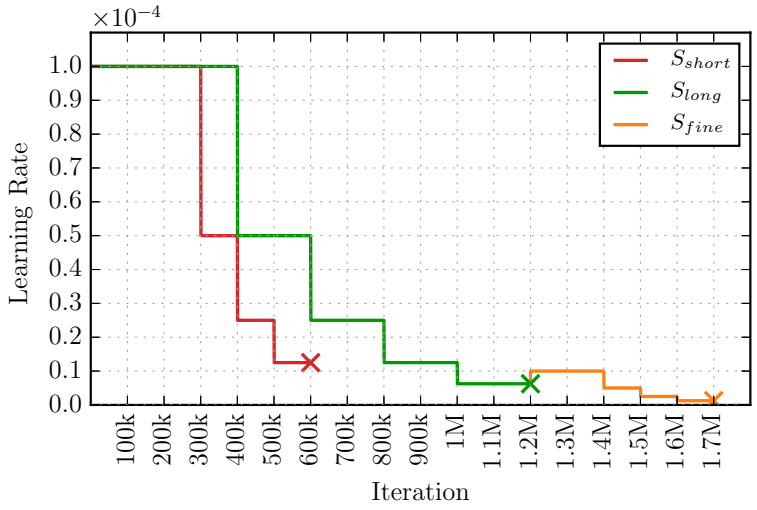


Figure 6.13: **Learning-rate schedules.** S_{short} is the original schedule with 600k iterations. S_{long} is a similar schedule but extended to twice the length (1.2M iterations). S_{fine} starts with a low learning rate and extends the schedule for a fine-tuning (additional 500k iterations).

Architecture	Datasets	S_{short}	S_{long}	S_{fine}
FlowNetS	Chairs	4.45	-	-
	Chairs	-	4.24	4.21
	Things3D	-	5.07	4.50
	mixed	-	4.52	4.10
	Things3D →Chairs	-	5.07	4.40
	Chairs →Things3D	-	4.24	**3.79**
Architecture	Datasets	S_{short}	S_{long}	S_{fine}
FlowNetC	Chairs	3.77	-	-
	Chairs	-	3.58	3.48
	Things3D	-	3.66	3.53
	mixed	-	3.41	3.29
	Things3D →Chairs	-	3.66	3.54
	Chairs →Things3D	-	3.58	**3.04**

Table 6.5: **Dataset schedules.** Results of training FlowNets with different schedules on different datasets (one network per row). Numbers indicate endpoint errors on Sintel train clean. FlyingChairs and FlyingThings3D are abbreviated to Chairs and Things3D, *mixed* denotes an equal mixture of both. Training on Chairs first and fine-tuning on Things3D yields the best results (the same holds true when testing on the KITTI dataset; see [IMS+17]).

We make the following observations:

(6.2.7) **The order of presenting training data with different properties matters.** Although FlyingThings3D is more realistic, training on FlyingThings3D alone leads to worse results than training on FlyingChairs. The best results are consistently achieved when first training on FlyingThings and only then fine-tuning on FlyingThings3D. This schedule also outperforms training on a mixture of both datasets.

We conjecture that the simpler FlyingChairs dataset helps the network learn the general concept of color matching without developing possibly confusing priors for 3D motion and realistic lighting too early. The result indicates the importance of training data schedules for avoiding shortcuts when learning generic concepts with deep networks.

(6.2.8) **FlowNetC consistently outperforms FlowNetS.** In all cases of Table 6.5, FlowNetC performs better than FlowNetS. We conclude that although it makes the network less generic, including a correlation operation is more efficient.

(6.2.9) **Improved results.** Just by modifying datasets and training schedules, we improved the FlowNetS result by ∼15% and the FlowNetC result by ∼20%.

6.3 Summary

This section has shown that:

- The two different CNNs are capable of learning matching and regularization heuristics for optical flow,
- FlowNetC contains more engineering but outperforms FlowNetS (which is most generic),
- augmentation is very important and strongly affects performance,
- skip connections and deep supervision add some performance but are not significant, and
- training details matter, especially the order of dataset presentation and best results are obtained when training first on FlyingChairs and then on FlyingThings3D.

(6.3.1)

A truly revolutional insight is that learning optical flow with a plain encoder-decoder network is possible.

Chapter 7

SceneFlowNet

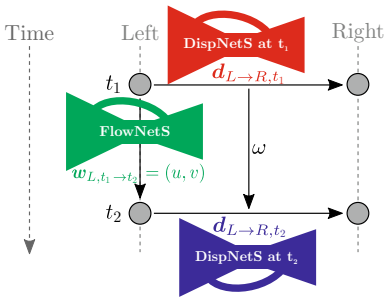
The SceneFlowNet published in [MIH+16] was the sole contribution of the author.

As described in Section 2.2, disparity estimation resembles a special case of the optical flow problem. The FlowNetS and FlowNetC architectures as well as the network stacks can therefore be applied to the disparity task. The benchmark results from Chapter 9 in fact show that this can yield state-of-the-art results.

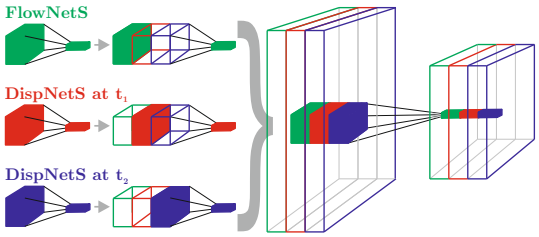
While disparity estimation is in some aspects simpler than optical flow estimation, the "Ivy league" that combines both tasks is scene flow. It is required for reconstruction as well as motion estimation and provides an important basis for numerous higher-level challenges, such as driver assistance, autonomous systems and robot navigation. Over the past decades, research has focused on the sub-tasks of disparity and optical flow estimation with considerable success, while the full scene flow problem has not been explored to the same extent. Although partial scene flow outside of occluded areas can simply be assembled from the sub-task results, it is expected that the joint estimation of all components would be advantageous.

Since the input to scene flow estimation comprises four different images (see Section 2.3), it is not straightforward to design an architecture using correlations. The most general solution for the joint task is to use a FlowNetS with the four frames as input and the different modalities as output. We present the first scene-flow-estimation CNN by constructing such a network from already existing Flow- and DispNets.

As shown in Figure 7.1a, we use a FlowNetS to estimate flow $\boldsymbol{w}_{L,t_1 \to t_2}$ between the left frames from time t_1 to t_2 and two DispNetSs to estimate the disparities $\boldsymbol{d}_{L \to R,t_1}$, $\boldsymbol{d}_{L \to R,t_2}$ from left to right at times t_1 and t_2. In order to construct a joint network, we interleave these three networks as illustrated in Figure 7.1b. Basically, this just combines the three data streams in one network by constructing larger convolutions with more input and output channels. The output channels of each FlowNet and DispNet then work only on input channels from the same network, while weights to input channels of other networks are set to 0. As such, the networks are just combined into one large network but still function completely independently.



(a) The figure shows our networks inserted into Figure 2.11. The disparities $\boldsymbol{d}_{L\to R,t_1}$ and $\boldsymbol{d}_{L\to R,t_2}$ are estimated by DispNets, and the optical flow $\boldsymbol{w}_{L,t_1\to t_2}$ is estimated by a FlowNet.



(b) Interleaving of the weights of the FlowNet (green) and the two DispNets (red and blue) into a SceneFlowNet. The figure shows how three convolutions from the same level are merged into one. The new filter masks are created by taking the weights of one network (left) and setting the weights of the other networks to zero, respectively (middle). The outputs from each network are then concatenated to yield one big network with three times the number of inputs and outputs (right).

Figure 7.1: **Joint SceneFlowNet from a FlowNet and two DispNets.** The SceneFlowNet outputs the flow $\boldsymbol{w}_{L,t_0\to t_1}$, the disparities $\boldsymbol{d}_{L\to R,t_0}$, $\boldsymbol{d}_{L\to R,t_1}$ and additionally the disparity change ω.

	Flow	Disparity	Disp. Ch
FlowNet	13.78		
DispNet		2.41	
FlowNet +500k	12.18		
DispNet +500k		2.37	
SceneFlowNet +500k	**10.99**	**2.21**	0.79

Table 7.1: **SceneFlowNet compared to single FlowNet and DispNet.** The table shows the performance of solving the single tasks compared to solving the joint scene flow task, trained and tested on FlyingThings3D. FlowNet was initially trained for 1.2M and DispNet for 1.4M iterations. +500k denotes 500k more iterations. The SceneFlowNet is initialized with the FlowNet and DispNet as described in Figure 7.1b. The table shows that solving the joint task yields better results in each individual task.

It gets interesting when one replaces the zeros by a small amount of noise and fine-tunes the resulting network for the joint scene flow task. In this case, mutual connections can evolve between the base networks. The results are given in Table 7.1. They show that:

(7.0.1) Solving the joint scene flow task inside one network performs better than solving the individual tasks.

This concludes that scene flow can be solved as a joint estimation task by CNNs.

Chapter 8

Network Stacks

The network stacks and concept of FlowNet 2.0, published in [IMS+17], are the sole contribution of the author.

Chapter 6 has shown that optical flow estimation with CNNs is generally possible but in terms of quality not close to state of the art yet. Traditional approaches rely on iterative methods [BM11, WRHS13, RWHS15, BTS15] and it is reasonable to ask whether CNNs can also benefit from estimating optical flow in multiple steps.

8.1 Stacking two Networks for Flow Refinement

To this end, we propose to stack networks to refine the result sequentially. In order to produce an initial solution, a FlowNet-S or FlowNet-C is used. The solution is then fed into another encoder-decoder network with the task to obtain an improved solution. This is illustrated in Figure 8.1. The first network in the stack gets the

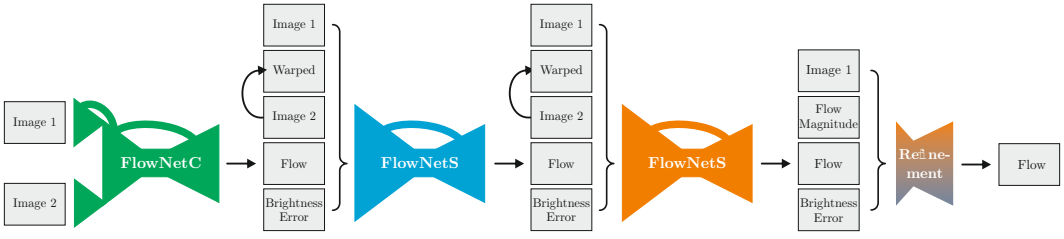


Figure 8.1: **FlowNet2 network stack.** The base network operates on the two images and can be a FlowNetS or FlowNetC. Subsequent networks get the images, the previously estimated flow, the warped second image and a brightness error as input. The brightness error is computed by subtracting the warped second image from the first one. The stack is optionally followed by a final refinement network that only gets the first image in addition to flow, flow magnitude and brightness error as input.

Stack architecture	Training enabled		Warping included	Warping gradient enabled	Loss after		EPE on Chairs test	EPE on Sintel clean
	Net1	Net2			Net1	Net2		
Net1	✓	–	–	–	✓	–	3.01	3.79
Net1 + Net2	✗	✓	✗	–	–	✓	2.60	4.29
Net1 + Net2	✓	✓	✗	–	✗	✓	2.55	4.29
Net1 + Net2	✓	✓	✗	–	✓	✓	2.38	3.94
Net1 + W + Net2	✗	✓	✓	–	–	✓	1.94	**2.93**
Net1 + W + Net2	✓	✓	✓	✓	✗	✓	1.96	3.49
Net1 + W + Net2	✓	✓	✓	✓	✓	✓	**1.78**	3.33

Table 8.1: **Ablation study for stacking two networks.** Evaluation of options when stacking two FlowNetS networks (Net1 and Net2). Net1 was trained with the Chairs→Things3D schedule from Section 6.2.8. Net2 is initialized randomly and subsequently. Net1 and Net2 are then trained together, or only Net2 is trained on Chairs with S_{long}. The column "Warping gradient enabled" indicates whether the warping operation produces a gradient during backpropagation. When training without warping, the stacked network overfits to the Chairs dataset. The best results on Sintel are obtained when fixing Net1 and training Net2 with warping.

images $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ as input. Subsequent networks get $\boldsymbol{I}_1$, $\boldsymbol{I}_2$, and the previous flow estimate $\boldsymbol{w}_i = (\boldsymbol{u}_i, \boldsymbol{v}_i)$ where i denotes the index of the network in the stack.

To make an assessment of the previous error and the computation of an incremental update easier for the network, we also optionally warp the second image $\boldsymbol{I}_2(x, y)$ via the flow $\boldsymbol{w}_i$ using bilinear interpolation to $\tilde{\boldsymbol{I}}_{2,i}(x, y) = \boldsymbol{I}_2(x + u_i, y + v_i)$. This way, the next network can check how well the estimated flow reconstructs the first image by comparing $\boldsymbol{I}_A$ and $\tilde{\boldsymbol{I}}_{2,i}(x, y)$ and can focus on finding the remaining increment. This is also motivated by the Gauss-Newton approach from Section 3.2.1 that uses different warpings of the second image to compute updates. When using warping, we additionally provide $\tilde{\boldsymbol{I}}_{2,i}$ and the error $e_i = ||\tilde{\boldsymbol{I}}_{2,i} - \boldsymbol{I}_1||$ as input to the next network (see Figure 8.1). Thanks to bilinear interpolation, the derivatives of the warping operation can be computed, which enables training of stacked networks end-to-end. For details of the derivation, the reader is referred to the supplemental material of [IMS+17].

Table 8.1 shows the effects of stacking two networks, of warping and of end-to-end training. We take the FlowNet2-S from Chapter 6 (trained on Chairs with S_{long} as well as Things3D with S_{short}) and add another FlowNetS on top. The second network is initialized randomly, and then the stack is trained on Chairs with the schedule S_{long}. We experimented with two scenarios: keeping the weights of the first network fixed or updating them together with the weights of the second network. In the latter case, the weights of the first network are fixed for the first 400k iterations to first provide a good initialization of the second network.

We report the error on Sintel train *clean* and on the test set of Chairs. Since the Chairs test set is much more similar to the training data than Sintel, comparing results on both datasets allows us to detect tendencies of overfitting.

	Number of Networks			
	1	2	3	4
Architecture	s	ss	sss	
Runtime	**7ms**	14ms	20ms	–
EPE	4.55	3.22	3.12	
Architecture	S	SS		
Runtime	18ms	37ms	–	–
EPE	3.79	2.56		
Architecture	c	cs	css	csss
Runtime	17ms	24ms	31ms	36ms
EPE	3.62	2.65	2.51	2.49
Architecture	C	CS	CSS	
Runtime	33ms	51ms	69ms	–
EPE	3.04	2.20	**2.10**	

Table 8.2: **FlowNet 2 stack variants.** Results on Sintel train clean for some variants of stacked FlowNet architectures following the best practices of Section 6.2.8 and 8.1. Each new network was first trained on Chairs with S_{long} and then on Things3D with S_{fine} (Chairs →Things3D schedule), while existing networks were kept fixed. Forward pass times are taken from an Nvidia GTX 1080.

We make the following observations:

(8.1.1) Just stacking networks without warping yields better results on Chairs but worse ones on Sintel; the stacked network is overfitting.

(8.1.2) Stacking *with* warping always improves results.

(8.1.3) Adding an intermediate loss after Net1 is advantageous when training the stacked network end-to-end.

(8.1.4) The best results are obtained by keeping the first network fixed and only training the second network.

Clearly, since the stacked network is twice as big as the single one, overfitting is an issue. The positive effect of flow refinement after warping can counteract this problem. Yet, the best of both is obtained when the stacked networks are trained one after the other. This avoids overfitting while having the benefit of flow refinement.

8.2 Stacking Multiple Diverse Networks

Rather than stacking identical networks, it is possible to stack networks of different types (FlowNetC and FlowNetS) as illustrated in Figure 8.1. We call the first one the *bootstrap* network as it differs from the second network by its inputs. However, the second network could be repeated for an arbitrary number of times in a recurrent fashion. We conducted this experiment and found that applying a network with the same weights multiple times as well as fine-tuning this recurrent part does not improve

results (for details, the reader is referred to the supplemental material of [DFI+15]). As also done in [NYD16, CP17], we therefore add networks with different weights to the stack. We use FlowNetC only in case of the bootstrap network, as the input to the next network is too diverse to be properly handled by the Siamese structure of FlowNetC.

Compared to identical weights, stacking networks with different weights increases the memory footprint but does not increase the runtime. In this case, the top networks are not constrained to a general improvement of their input but can perform different refinement tasks at different stages, and the stack can be trained in smaller pieces by fixing existing networks and adding new networks one at a time. We do so by using the Chairs →Things3D schedule with S_{long} and S_{fine} from Section 6.2.8 for every new network as well as the best configuration with warping from Section 8.1 (row 4 of Table 8.1).

(8.2.1) **Notation:** For networks trained with the schedule Chairs →Things3D, we use the name *FlowNet2*. Networks in a stack are trained with the same schedule one after the other. For the stack configuration, we append upper- or lower-case letters to indicate the types of networks and the original size or thin version with 3/8 of the channels. E.g.: *FlowNet2-CSS* stands for a network stack consisting of one FlowNetC and two FlowNetSs. *FlowNet2-css* is the same network stack but with fewer channels. "FlowNet2" by itself refers to the FlowNet2-CSS network stack from [IMS+17] that includes an extra network for small displacements trained on ChairsSDHom. As results from Chapter 9 will show, small displacements can be treated in the main stack with according training. For this reason, the small displacement network is omitted here.

Table 8.2 shows the performance of different network stacks. Most notably, the final FlowNet2-CSS result improves by ∼30% over the single network FlowNet2-C. Furthermore, two small networks always outperform one large network, despite being faster and having fewer weights: FlowNet2-ss (11M weights) over FlowNet2-S (38M weights) and FlowNet2-cs (11M weights) over FlowNet2-C (38M weights). Training smaller units step by step proves to be advantageous and enables to train very deep networks for optical flow.

8.2.1 Performance and Runtime

The last section illustrated that network stacks can boost the performance of optical flow estimation while still maintaining low runtimes. Figure 8.2 (see next page) gives an endpoint error vs. runtime comparison of the FlowNet2 family and traditional methods (a full benchmark evaluation will be given in Chapter 9). Depending on the type of application, the different stack configurations provide variants from 8 to 140 frames per second, which is orders of magnitude faster than traditional approaches with comparable performances. Up to today, many extensions that increased the performance even further have been proposed by the community [SYLK18, HTL18, NŠM18].

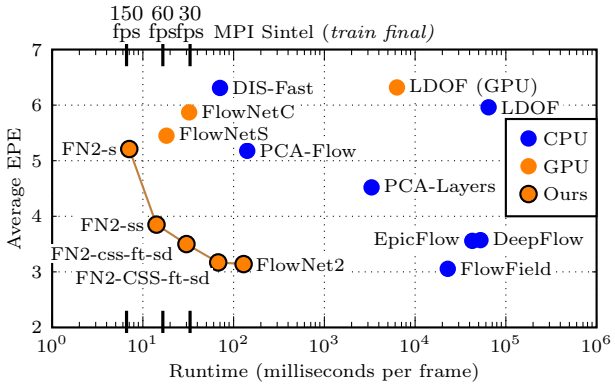


Figure 8.2: **FlowNet2 family endpoint error vs. runtime.** Endpoint error vs. runtime comparison to the fastest existing methods (at the time of publication [IMS+17]) with available code. The FlowNet2 family outperforms other previous methods by a large margin. "-ft-sd" indicates network stacks that were fine-tuned on the ChairsSDHom dataset.

Quantitative evaluations are shown in Figures 8.3 and 8.4:

(8.2.2) The observable improvement over FlowNetS is very large. Notably, FlowNet2 provides smooth flow fields and at the same time yields very crisp motion boundaries and fine details, while none of the traditional methods actually provide a comparable level of detail. This proves that CNNs are capable of solving the regularization problem much better.

Real-World Data Interestingly, this is also the case for the real-world data from Figure 8.4. Although the networks were trained on the unrealistic FlyingChairs, FlyingThings3D and ChairsSDHom dataset, they also seem to perform better on real data than traditional methods, yielding fine details and being very robust to compression artifacts. This proves that for the optical flow task, the important aspect of the training data is the concept of correspondence and not the actual objects themselves.

8.3 Evaluation on Applications

In order to assess performance of FlowNet2 in real-world applications, we compare the performance of action recognition and motion segmentation. For both applications, good optical flow is key. Thus, a good performance on these tasks also serves as an indicator for good optical flow.

For motion segmentation, we rely on the well-established approach by Ochs et al. [OMB14] to compute long-term point trajectories. A motion segmentation is obtained from these by using the state-of-the-art method from Keuper et al. [KAB15]. The results are shown in Table 8.3 (turn page). The original model in Ochs et al. [KAB15] is built on LDOF [BM11]. We also included other popular optical flow methods in the comparison. The initial FlowNetS [DFI+15] did not prove to be useful for motion segmentation. In contrast, the FlowNet2 is as reliable as other state-of-the-art methods while being orders of magnitude faster.

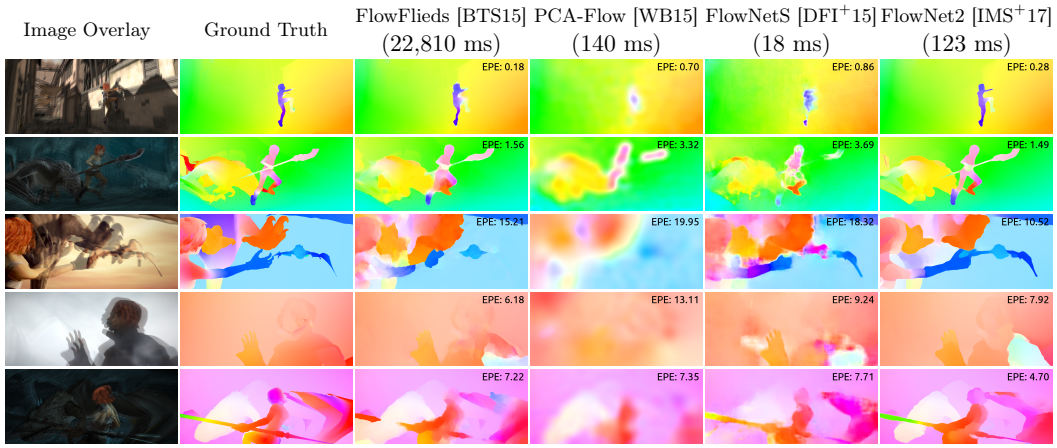


Figure 8.3: **FlowNet2 results on Sintel.** The table shows quantitative results on Sintel from traditional methods, the original FlowNetS [DFI+15] trained on FlyingChairs with S_{short} and FlowNet2 [IMS+17] (which includes FlowNet2-CSS). FlowNet2 performs similar to FlowFields and is able to extract fine details, while methods running at comparable speeds perform much worse (PCA-Flow and FlowNetS).

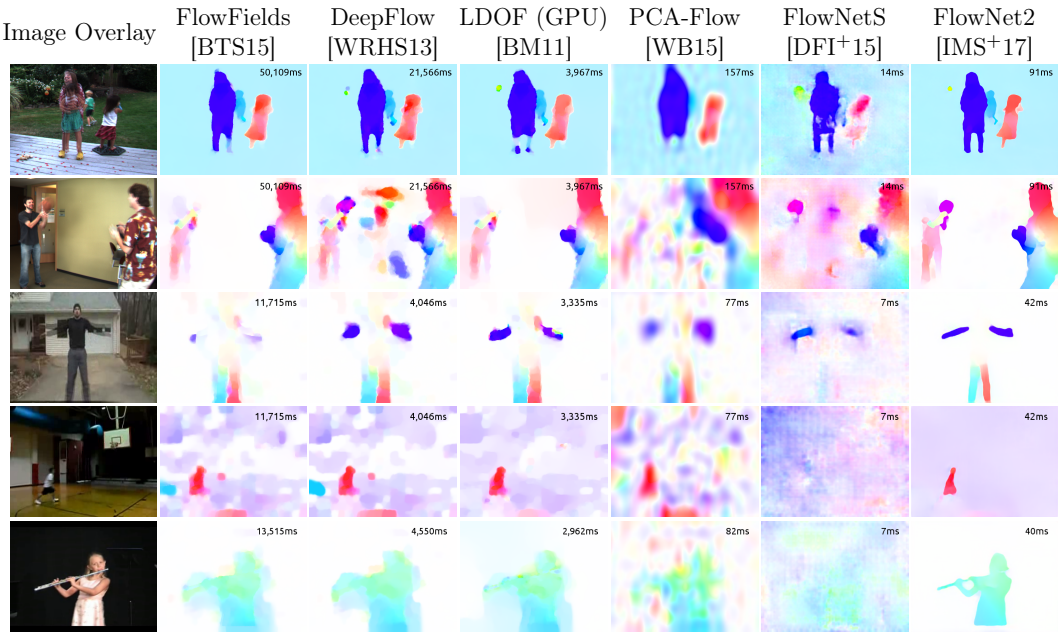


Figure 8.4: **FlowNet2 results on real images.** The table shows quantitative results on real images from traditional methods, the original FlowNetS [DFI+15] trained on FlyingChairs with S_{short} and FlowNet2 [IMS+17] (which includes FlowNet2-CSS). The top two rows are from the Middlebury dataset and the bottom three from UCF101. Note how well FlowNet2 generalizes to real-world data, i.e., it produces smooth flow fields as well as crisp boundaries and is robust to motion blur and compression artifacts. Given timings of methods differ due to different image resolutions.

Optical flow is also a crucial feature for action recognition. In order to assess the performance, we trained the temporal stream of the two-stream approach from Simonyan et al. [SZ14a] with different optical flow inputs. Table 8.3 shows that FlowNetS [DFI+15] did not provide useful results, while the flows from the FlowNet2 family yields comparable results to state-of-the-art methods.

8.4 Summary

This section has shown that:

- Optical flow estimation performance can be significantly increased with network stacks,
- network stacks deliver smooth flow fields, very crisp motion boundaries and fine details,

(8.4.1)

- CNNs solve the regularization problem much better than traditional methods,
- CNNs also have an extremely better speed/accuracy trade-off (by orders of magnitude),
- networks trained on the synthetic dataset generalize to real data very well, and
- real-world applications like motion segmentation and action recognition perform very well with FlowNet2 flows.

	Motion Seg.		Action Recog.
	F-Measure	Extracted Objects	Accuracy
LDOF-CPU [BM11]	79.51%	28/65	79.91%†
DeepFlow [WRHS13]	**80.18%**	29/65	**81.89%**
EpicFlow [RWHS15]	78.36%	27/65	78.90%
FlowFields [BTS15]	79.70%	**30/65**	79.38%
FlowNetS [DFI+15]	56.87%‡	3/62‡	55.27%
FlowNet2-css-ft-sd	77.88%	26/65	75.36%
FlowNet2-CSS-ft-sd	79.52%	30/65	**79.64%**
FlowNet2	**79.92%**	**32/65**	79.51%

Table 8.3: **Motion segmentation and action recognition results by using different methods. Motion Segmentation:** We report results by using [OMB14, KAB15] on the training set of FBMS-59 [BM10, OMB14] with a density of 4 pixels. Different densities and error measures are given in the supplemental material of [IMS+17]. "*Extracted objects*" refers to objects with F $\geq$ 75%. ‡FlowNetS is evaluated on 28 out of 29 sequences; on the *lion02* sequence, the optimization did not converge even after one week. **Action Recognition:** We report classification accuracies after training the temporal stream of [SZ14a]. We use a stack of five optical flow fields as input. †In order to reproduce results from [SZ14a], for action recognition, we use the OpenCV LDOF implementation. Note the generally large difference for FlowNetS and FlowNet2 as well as the performance compared to traditional methods.

Chapter 9

Joint Flow, Occlusion and Motion Boundary Estimation

The work presented in this chapter was started by the author and was continued together with Tonmoy Saikia. The author contributed the loss functions for motion boundaries and occlusions as well as the network setups.

Occlusions play an important role in disparity and optical flow estimation, since matching costs are not available in occluded areas and occlusions indicate depth or motion boundaries. As explained in Section 3.5, occlusions and motion boundaries are hard to integrate into traditional methods and pose a significant problem. In the past, occlusion estimation was stated as "notoriously difficult" [LZS13] and a "chicken-and-egg" problem in the literature [HR17, PRCBP16]. It is therefore most interesting to ask how the learned CNN approaches from this work can solve these problems.

9.1 Estimating Occlusions with CNNs

We extend the end-to-end CNN approach of Chapter 6 by occlusion estimation. To this end, we perform a set of experiments with FlowNetS with different inputs and outputs. For the occlusions, we add further output channels with per-pixel classes 0 and 1 to indicate occlusion. We train this output with a cross-entropy loss against the ground truth. The setup is illustrated in Figure 9.1:

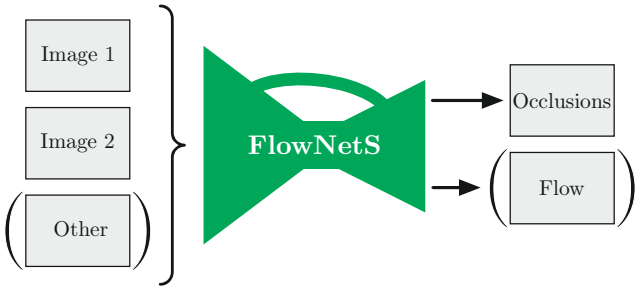


Figure 9.1: **Occlusion estimation.** We extend the basic FlowNet from Figure 6.1 by occlusion estimation and perform experiments with different in- and outputs. Optionally, we input flows, warped images and forward/backward consistency maps.

#	Input	F-measure	
		FlyingThings3D	Sintel train clean [BWSB12]
-	S2DFlow [LZS13]	-	0.470
1	Images A+B	0.790	0.545
2	Images A+B, GT fwd Flow	0.932	**0.653**
3a	Images A+B, GT fwd Flow, GT bwd flow	0.930	-
3b	Images A+B, GT fwd Flow, GT bwd flow warped+flipped	**0.943**	-
3c	Images A+B, GT fwd Flow, GT fwd/bwd consistency	**0.943**	-

Table 9.1: **Estimation of only occlusions from different inputs with a FlowNetS.** The first column indicates the experiment number. Networks were trained on FlyingChairs with schedule $S_{long}/2$ and fine-tuned on FlyingThings3D with $S_{fine}/2$. S2DFlow was the state-of-the-art method at the time of publication (ECCV 2018). The results show that contrary to literature [PRCBP16, LZS13, HAB11], occlusion estimation is even possible from just the two images and the result is significantly better than state of the art. Since Sintel train clean [BWSB12] does not provide the ground-truth backward flow, we additionally report numbers on FlyingThings3D [MIH+16]. Providing the optical flow, too, clearly improves the results. Contrary to traditional methods, only the forward flow is sufficient.

Before we come to a joint estimation of occlusions and optical flow, we investigate some experiments to estimate only occlusions from different input data. The results are shown in Table 9.1.

In order to determine occlusions, one basically has to check if a patch from the first image is still present in the second image. This is very similar to the correspondence estimation task in that it requires patch comparison. Instead of outputting the correspondence to the matching patch, the output in this case is whether a matching patch exists. In order to make this decision, a local neighborhood usually has to be considered, and a locally determined threshold on the similarity between the patches is required. The results from the first experiment of Table 9.1 show that CNNs can actually perform this task extremely well without the explicit correspondence estimation.

As a next step, we additionally provide the ground-truth forward optical flow to the network (experiment #2 in Table 9.1). In this setting, the network has to compare source and target locations of flow vectors to find occluded areas. Since the ground-truth flow input comes in addition to the images from the first experiment, the performance increases as expected.

The traditional way to determine occlusions is to determine if forward and backward flows are consistent [ADPS07] (see also Figure 3.10). In the final set of experiments, we therefore also provide different versions of the backward flow. Experiment #3a directly includes the backward flow. Note that a consistency check requires to look up the backward flow at the target location of the forward flow. For this reason, we also warp the backward flow with the forward flow in experiment #3b. In experiment #3c, we do not input the backward flow directly but instead precompute the consistency (the Euclidean difference between forward and backward flow) and input it. We observe that none of these really outperform inputting only the forward flow.

	Configuration	EPE	F-measure
1	FlowNetC estimating flow	3.21	-
2	FlowNetC estimating occlusions	-	**0.546**
3	FlowNetC estimating flow + occlusions	**3.20**	0.539
4	FlowNetC-Bi estimating fwd/bwd flow + fwd occlusions	3.26	0.542

Table 9.2: **Joint estimation of flow and occlusions with a FlowNetC.** The first column indicates the experiment number. Networks were trained on FlyingChairs with schedule $S_{long}/2$ and fine-tuned on FlyingThings3D with $S_{fine}/2$. Results are from Sintel train clean. As can be seen from the results, joint estimation of oclusions and flow neither meaningfully improves nor degrades the flow performance.

From the experiments of Table 9.1, we conclude:

(9.1.1) **Occlusion estimation without optical flow is possible and by itself surpasses state of the art.** In contrast to existing literature, where classifiers are always trained with flow input [PRCBP16, LZS13, HAB11, LSS12b] or occlusions are estimated jointly with optical flow [SLP14, HR17], we show that a deep network can learn to estimate the occlusions extremely well directly from two images.

(9.1.2) **Using the flow as input helps.** The flow provides the solution for correspondences, and the network can use these correspondences. Clearly, this helps, particularly since we provided the correct optical flow.

(9.1.3) **Adding the backward flow yields almost the same results.** Providing the backward flow directly does not help. This can be expected, because the information for a pixel of the backward flow is stored at the target location of the forward flow, and a look-up is difficult for a network to perform. Warping the backward flow or providing the forward/backward consistency yields almost the same result. This finding is contrary to traditional methods, where occlusion checks are usually performed on forward and backward flow.

9.2 Joint Estimation of Occlusions and Flow

9.2.1 Within a Single Network

In this section, we investigate estimating occlusions jointly with optical flow, as many traditional methods try to do [SLP14, HR17]. Here, we provide only the image pair and therefore can use a FlowNetC instead of a FlowNetS. The first experiment from Table 9.2 shows that just occlusion estimation with a FlowNetC performs similar to the FlowNetS of the last section. Surprisingly, from experiments #1 to #3 of Table 9.2, we find that joint flow estimation neither meaningfully improves nor deproves the flow or the occlusion quality. In row #4 of the table, we additionally estimate the backward flow to enable the network to reason about forward/backward consistency. The suffix "-Bi" indicates that we add a second correlation from the second to the first image to the network. However, we find that this does not affect performance in a meaningful way either.

As mentioned in the last section, both occlusion and flow estimation require finding similar patches among both images. When estimating only optical flow, occlusions need to be regarded by deciding that no correspondence exists for an occluded pixel and by filling the occluded area with information inferred from the surroundings. Therefore, knowledge about occlusions is mandatory for correct correspondence and flow estimation.

(9.2.1) Since the results of Section 9.1 show that occlusion estimation is an easy task for CNNs and making the occlusion estimation in our network explicit does not change the result, we conclude that an end-to-end trained network only for flow already implicitly performs all necessary occlusion reasoning. By making it explicit, we obtain the occlusions as an additional output at no cost, but the flow itself remains unaffected.

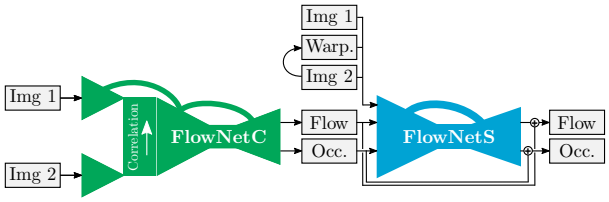
9.2.2 With a Refinement Network

In the last section, we investigated the joint estimation of flow and occlusions in a single network, with the result that the network implicitly solves the "chicken-and-egg" problem very well. We now extend the occlusion estimation to a network stack and investigate if the estimated occlusions can help refine the flow. To this end, we propose the three stack variants from Figure 9.2 that will be explained in the following.

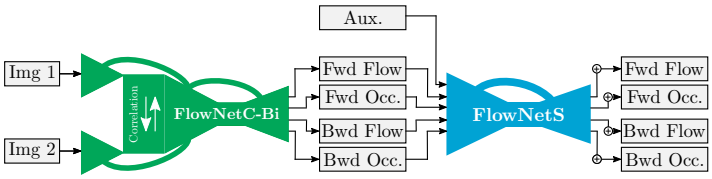
We make two general modifications: 1.) We leave away the brightness error input for the refinement networks as it can easily be computed by the network by taking the difference between the warped second and the first image. 2.) We further modify the stack by integrating the suggestion of Pang et al. [PSR+17] and add residual connections to the refinement networks. As described in [HZRS16], it is not trivial for CNNs to learn an identity mapping. We find that posing the second network as residual leads to a similar overall performance, but with the difference of much faster convergence. 3.) Due to the faster convergence, we shorten all schedules by a factor of 2, i.e., each network in the stack is trained with schedule $S_{long}/2$ on FlyingChairs and $S_{fine}/2$ on FlyingThings3D. While this slightly changes results of individual networks in the stack, we find that the overall performance with a stack of three networks converges to the same result as FlowNet2.

The first variant of our joint estimation stack shown in Figure 9.2a (see next page) is a simple extension of the stack from Chapter 8.1 by estimating and inputting the occlusions in addition to the flow.

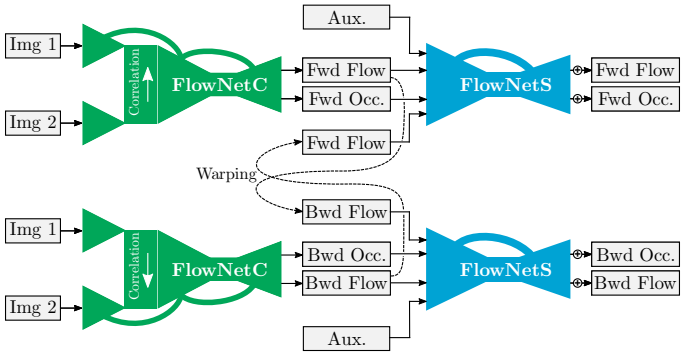
The second variant from Figure 9.2b (see next page) extends the stack to estimating forward and backward flows jointly, as is common in traditional methods that perform occlusion reasoning [SLP14, HR17]. In principle, the second network can make use of the consistency constraint outside of occluded regions in this setup by using the occlusion output from the first network. Note, though, that such a consistency check requires warping, as the backward flow has to be looked up at the target positions of the forward flow.



(a) Extension of network stack with occlusions and residual connections.



(b) Architecture for joint estimation of forward/backward flows and occlusions. The suffix "-Bi" indicates that we add a second correlation from the second to the first image to the network. See figure caption for more details.



(c) Dual forward and backward estimation architecture with mutual warping. See figure caption for more details.

Figure 9.2: **Joint flow and occlusion refinement stack variants.** Overview of possible refinement stacks for flow and occlusions. The residual connections are only shown in the first figure and indicated by + elsewhere. Aux. refers to the images plus a warped image for each input flow respectively.

In order to model the warping explicitly, we propose the third variant shown in Figure 9.2c, in which we model forward and backward flow estimation as separate streams and perform the mutual warping to the other direction after each network. E.g., we warp the estimated backward flow after the first network to the coordinates of the first image by using the forward flow. The network then has the forward and the corresponding backward flow at the same pixel position and can compute the difference to check the consistency directly.

Configuration	EPE	F-measure
Only flow as in FlowNet2-CS [IMS+17]	2.28	-
+ occlusions (Figure 9.2a)	**2.25**	**0.590**
+ bwd direction (Figure 9.2b)	2.77	0.572
+ mutual warping (Figure 9.2c)	2.25	0.589

Table 9.3: **Results of joint flow and occlusion refinement stacks.** Networks were trained on FlyingChairs with schedule $S_{long}/2$ and fine-tuned on FlyingThings3D with $S_{fine}/2$ respectively. Results are from Sintel train clean. Simply adding occlusions in a straightforward manner performs better than or similar to more complicated approaches. In general, adding occlusions does not meaningfully improve the flow estimation.

The results of the three architecture variants are given in Table 9.3. While the first variant (Figure 9.2a) and the last variant (Figure 9.2c) are indifferent about the additional occlusion input, the architecture with joint forward and backward estimation (Figure 9.2b) performs worse. This shows that the network does not exploit the redundancy between forward and backward flows, but estimating them jointly rather increases the complexity.

Overall, we find that designing architectures motivated by traditional methods to model consistency implicitly or explicitly does not improve the results. Furthermore, adding the occlusion estimation does not help the flow estimation. This means that either the occluded areas are already filled correctly by the base network or in a stack without explicit occlusion estimation, the second network can easily recover occlusions from the flow (which was found in Statement 9.1.2) and does not require the explicit input.

(9.2.2) We finally conclude that occlusions can be obtained at no extra cost but do not actually influence the flow estimation, and that it is best to leave the inner workings to the optimization by using only the baseline variant (Figure 9.2a). This is contrary to the findings from traditional methods. However, the quality of the estimated occlusions by itself surpasses traditional methods by far and thus is important for applications that actually require occlusions.

9.3 Flow, Occlusion and Motion Boundary Estimation Stack

In order to estimate high-quality occlusions and motion boundaries along with the flow, we extend the stack of Figure 9.2a to the full FlowNet2 stack, as illustrated in Figure 9.3 (see next page). While occlusions are important for refinement from the beginning, boundaries are only required in later refinement stages. Therefore, we add the boundaries in the third network. Experimentally, we also found that when adding depth or motion boundary prediction in earlier networks, these networks predicted details better but failed more rigorously in case of errors. Predicting exact boundaries early would be contrary to the concept of a refinement pipeline.

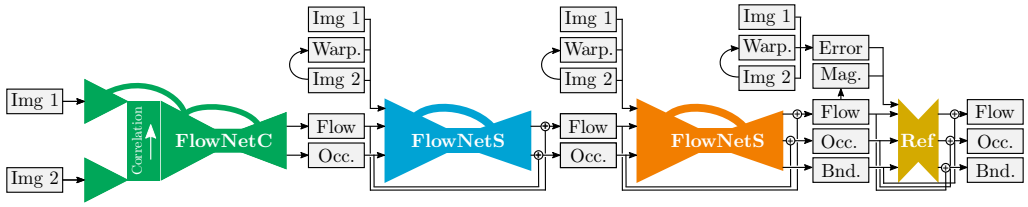


Figure 9.3: **Flow, occlusion and motion boundary estimation stack.** In order to estimate occlusions and motion boundaries, we follow the FlowNet2 architecture and build the full stack. In the final network, we add motion boundaries to refine all modalities together.

9.4 Benchmark Results

We train the full network stack in the last section for optical flow and disparity estimation with additional outputs for occlusions as well as depth or motion boundaries and evaluate all modalities on public benchmarks.

(9.4.1) We name the networks from the stack of Figure 9.3 that are trained with schedules $S_{long}/2$ and $S_{fine}/2$ on FlyingChairs and FlyingThings3D, DispNet3 and FlowNet3. We denote the optional refinement network with an "R" in the name, e.g. FlowNet3-CSSR.

9.4.1 Occlusion Estimation

In Tables 9.4 and 9.5, we compare our occlusion estimations to other methods. In terms of disparity our method outperforms Kolmogorov et al. [KMT14] for all scenes except one. For the more difficult case of optical flow, we outperform all existing methods by a very large margin:

(9.4.2) At the time of publication at ECCV 2018, FlowNet3-CSSR sets a new state of the art for occlusion estimation.

As already indicated by Table 9.1, this confirms that CNNs can solve the occlusion estimation problem much better than any traditional algorithm. The qualitative results of Figure 9.4 (see next page) also support this finding. While consistency checking is able to capture mainly large occlusion areas, S2DFlow [LZS13] also manages to find some details. In many cases, MirrorFlow [HR17] misses details. On the other hand side, our CNN is able to estimate the fine details very well.

9.4.2 Motion Boundary Estimation

For motion boundary estimation, we compare our CNNs to Weinzaepfel et al. [WRHS15], which is the state-of-the-art method to the best of our knowledge. It uses a random forest classifier and is trained on the Sintel dataset. Although we do not train on Sintel, judging by the results of Table 9.6, our CNN outperforms their method by a

Method	F-Measure					
	Cones	Teddy	Tsukuba	Venus	Sintel clean	Sintel final
Kolmogorov et al. [KMT14]	0.45	**0.63**	0.60	0.41	-	-
Tan et al. [TCM17]	0.44	0.40	0.50	0.33	-	-
Ours (DispNet3-CSS)	**0.91**	0.57	**0.68**	**0.44**	**0.76**	**0.72**

Table 9.4: **Evaluation of estimated disparity occlusions from our DispNet3.** We compare our DispNet3 to other methods on examples from the Middlebury 2001 and 2003 datasets (results of Kolmogorov et al. [KMT14] and Tan et al. [TCM17] taken from [TCM17]) as well as the Sintel train dataset. Only in the Teddy scene of Middlebury, our occlusions are outperformed by Kolmogorov et al. [KMT14]. Unfortunately, no previous results are available on the Sintel benchmark.

Method	Type	F-Measure	
		clean	final
FlowNet2 [IMS+17]	consistency	0.377	0.348
MirrorFlow [HR17]	estimated	0.390	0.348
S2DFlow [LZS13]	estimated	0.470	0.403
Ours (FlowNet3-CSSR)	estimated	**0.703**	**0.654**

Table 9.5: **Evaluation of estimated flow occlusions from our FlowNet3.** We compare our FlowNet3 to other occlusion estimation methods on the Sintel train dataset. For the first entry, occlusions were computed by using forward/backward consistency post-hoc. Our approach yields much better occlusions by a large margin on both the clean and final version of the dataset.

Method	Sintel clean	Sintel final
Weinzaepfel et al. [WRHS15]	76.3	68.5
Ours (FlowNet3-CSSR)	**86.3**	**79.5**

Table 9.6: **Evaluation of estimated motion boundaries from our FlowNet3.** We compare our FlowNet3-CSSR motion boundary estimation to Weinzaepfel et al. [WRHS15] on the Sintel train dataset. The table shows the mean average precision as computed with their evaluation code. Although Weinzaepfel et al. [WRHS15] trained on the Sintel train clean dataset, our method outperforms theirs by a large margin on both the clean and final version of the dataset.

large margin. The improvement in quality is also visible very well in the qualitative results from Figure 9.5.

(9.4.3) At the time of publication at ECCV 2018, FlowNet3-CSSR sets a new state of the art for motion boundary estimation.

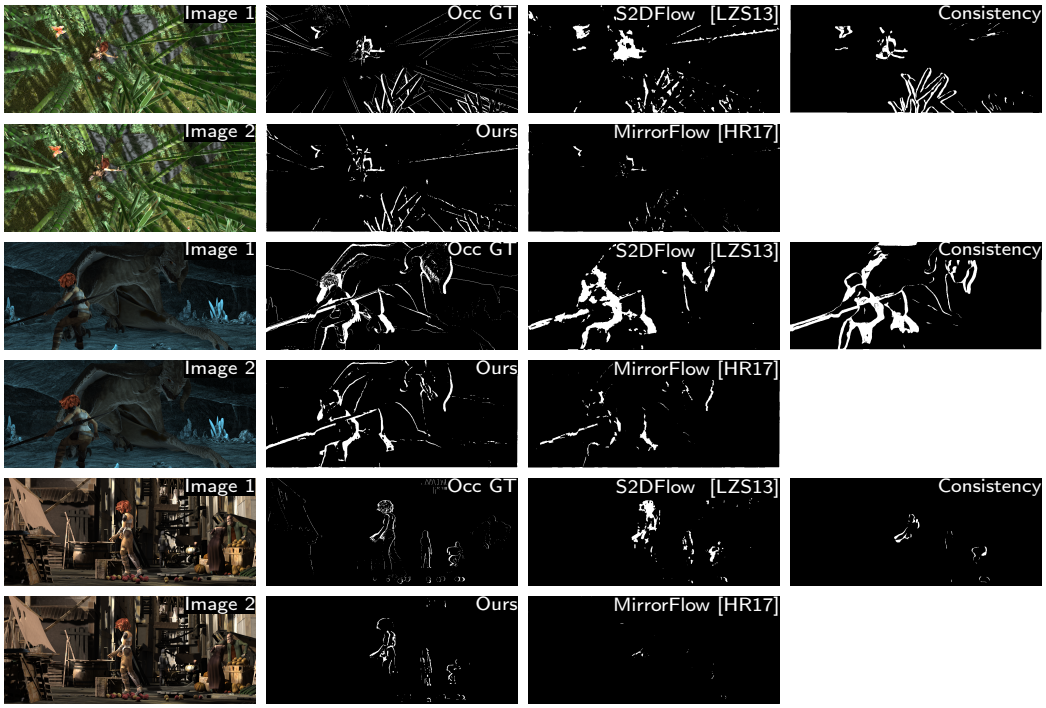
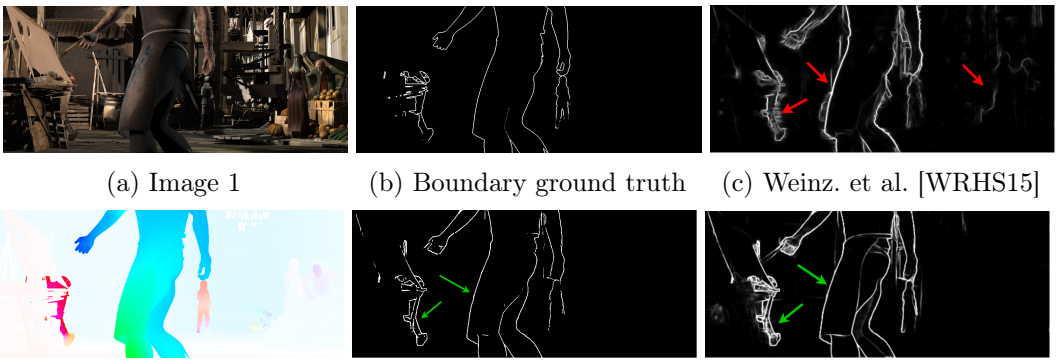


Figure 9.4: **Qualitative results for occlusion estimation.** In comparison to other methods and the forward-backward consistency check, our method is able to capture very fine details and is very close to the ground truth.



(a) Image 1 (b) Boundary ground truth (c) Weinz. et al. [WRHS15]

(d) Flow ground truth (e) Ours (hard) (f) Ours (soft)

Figure 9.5: **Qualitative results for motion boundary estimation.** Our approach succeeds in detecting the object in the background and has less noise around motion edges than existing approaches (see green arrows). Weinzaepfel et al. detect some correct motion details in the background. However, these details are not captured in the ground truth.

Method	Sintel (clean)	KITTI (2012)		KITTI (2015)		Runtime (msecs)
	AEE *train*	AEE *train*	Out-noc *test*	AEE *train*	D1-all *test*	
Standard:						
SGM [Hir05]	19.62	10.06	-	7.21	10.86%	1,100
CNN-based:						
Our DispNetC [MIH+16]	5.66	1.75	-	1.59	-	60
Our DispNetC-ft [MIH+16]	21.88	1.48	4.11%	(0.68)	4.34%	60
CRL [PSR+17]	16.13	**1.11**	-	(0.52)	2.67%	47
GC-Net [KMD+17]	-	-	**1.77**%	-	2.87%	90
MC-CNN-acrt [ŽL14]	-	-	2.43%	-	3.89%	67,000
DRR [GK17]	-	-	-	-	3.16%	40
L-ResMatch [SW17]	-	-	2.27%	-	3.42%	42,000
With joint occl. est.:						
SPS stereo [YMU14]	-	-	3.39%	-	5.31%	2,000
Our DispNet3-CSS	**2.33**	1.40	-	**1.37**	-	70
Our DispNet3-CSS-ft	5.53	(0.72)	1.82%	(0.71)	**2.19**%	70
Our DispNet3-css	2.95	1.53	-	1.49	-	**30**

Table 9.7: **Benchmark results for disparity estimation.** We report the average endpoint error (AAE) for Sintel. On KITTI, Out-noc and D1-all are used for the benchmark ranking on KITTI 2012 and 2015 respectively. Out-noc shows the percentage of outliers with errors of more than 3px in non-occluded regions, whereas D1-all shows the percentage in all regions. Entries in parentheses denote methods that were fine-tuned on the evaluated dataset. Our network denoted with "-ft" is fine-tuned on the respective training datasets. Timings were taken from an Nvidia GTX 1080Ti. We obtain state-of-the-art results on Sintel and KITTI 2015. Also, our networks generalize well across domains, as shown by the good results of the non-fine-tuned networks.

9.4.3 Disparity Estimation

The benchmark results for disparity estimation are given in Table 9.7. Note that all other CNN approaches appeared after DispNet which was itself contained in one of the publications from this thesis [MIH+16].

(9.4.4) At the time of publication at ECCV 2018, our DispNet3-CSS sets a new state of the art on Sintel, and our DispNet3-CSS-ft does so on KITTI 2015.

9.4.4 Optical Flow Estimation

The benchmark results for optical estimation are given in Table 9.8 (see next page). Note that all other CNN approaches appeared after FlowNet which was itself contained in one of the publications from this thesis [DFI+15].

(9.4.5) At the time of publication at ECCV 2018, our FlowNet3-CSS-ft sets new states of the art on KITTI 2012 and KITTI 2015.

Method	Sintel (clean) AEE		Sintel (final) AEE		KITTI (2012)		KITTI (2015)		Runtime (msecs)
	train	*test*	*train*	*test*	*AEE train*	*Out-noc test*	*AEE train*	*F1-all test*	
Standard:									
EpicFlow [RWHS15]	2.27	4.12	3.56	6.29	**3.09**	7.88%	9.27	26.29%	42,000
FlowfieldsCNN [BVS17]	-	3.78	-	5.36	-	4.89%	-	18.68%	23,000
DCFlow [XRK17]	-	**3.54**	-	5.12	-	-	-	14.86%	9,000
CNN-based:									
Our FlowNet2 [IMS+17]	**2.02**	3.96	**3.14**	6.02	4.09	-	10.06	-	123
Our FlowNet2-ft [IMS+17]	(1.45)	4.16	(2.01)	5.74	(1.28)	-	(2.30)	11.48%	123
SpyNet [RB17]	4.12	6.69	5.57	8.43	9.12	-	-	-	**16**
SpyNet-ft [RB17]	(3.17)	6.64	(4.32)	8.36	(4.13)	12.31%	-	35.07%	**16**
PWC-Net [SYLK18]	2.55	-	3.93	-	4.14	-	10.35	33.67%	30
PWC-Net-ft [SYLK18]	(2.02)	4.39	(2.08)	**5.04**	-	4.22%	(2.16)	9.80%	30
With joint occl. est.:									
MirrorFlow [HR17]	-	3.32	-	6.07	-	4.38%	-	10.29%	660,000
S2D flow [LZS13]	-	18.48	-	6.82	-	-	-	-	2,280,000
Our FlowNet3-CSS	2.08	3.94	3.61	6.03	3.69	-	**9.33**	-	68
Our FlowNet3-CSS-ft	(1.47)	4.35	(2.12)	5.67	(1.19)	**3.45**%	(1.79)	**8.60**%	68
Our FlowNet3-css	2.65	-	4.05	-	5.05	-	11.74		33

Table 9.8: **Benchmark results for optical flow estimation.** We report the average endpoint error (AAE) for all benchmarks, except KITTI where Out-noc and F1-all are used for the benchmark ranking on KITTI 2012 and 2015 respectively. Out-noc shows the percentage of outliers with errors of more than 3px in non-occluded regions, whereas F1-all shows the percentage in all regions. Entries in parentheses denote methods that were fine-tuned on the evaluated dataset. Timings were taken from an Nvidia GTX 1080Ti. On the Sintel dataset, the performance of our networks is comparable to FlowNet2. When comparing to other methods with joint occlusion estimation, we are faster by multiple orders of magnitude. On KITTI 2012 and 2015, we obtain state-of-the-art results among all optical flow methods (two-frame, non-stereo).

9.5 Application to Motion Segmentation

We apply the estimated occlusions to the motion segmentation framework by Keuper et al. [KAB15]. Like [BM10], this approach computes long-term point trajectories based on optical flow. For deciding when a trajectory ends, the method depends on reliable occlusion estimates. These are commonly computed by using the post-hoc consistency of forward and backward flow, which has been shown to perform badly Table 9.5.

We replace the post-hoc estimation with the occlusions from our FlowNet3-CSS and from other methods. Table 9.9 (see next page) shows the clear improvements obtained by the more reliable occlusion estimates on the common FBMS-59 motion segmentation benchmark. In row #4, we show how adding our occlusions to flow estimations of FlowNet2 can improve results. This shows that by only adding occlusions, we recover 30 objects instead of 26. The last result from our joint estimation of flow and occlusions further improves the findings. Besides the direct quantitative and qualitative evaluation from the last sections, this shows the usefulness

Method	FBMS test set (30 sequences)			
	Precision	Recall	F-Measure	#Objects
Third-Order Multicut [Keu17]	87.77%	71.96%	79.08%	29/69
DeepFlow [WRHS13]	88.20%	69.39%	77.67%	26/69
Ours FlowNet2	86.73%	68.77%	76.71%	26/69
Ours FlowNet2 + our occ	85.67%	70.15%	77.14%	30/69
Ours FlowNet3-CSS	**88.71%**	**73.60%**	**80.45%**	**31/69**

Table 9.9: **Motion segmentation results by using our occlusions.** Results of motion segmentation from Keuper et al. [KAB15] on the FBMS-59 test set [BM10, OMB14] (with a sampling density of 8px). The fourth row uses flows from FlowNet2 [IMS+17] combined with our occlusions (from FlowNet3-CSS). The improved results show that occlusions help the motion segmentation in general. The last row shows the segmentation when using our joint estimation of flow and occlusions, which performs best and also improves over the recent state of the art on sparse motion segmentation by using higher order motion models [Keu17]. *5mm

of our occlusion estimates in a relevant application. Our final results even outperform the recently proposed third-order motion segmentation with multicuts [Keu17]:

(9.5.1) At the time of publication at ECCV 2018, using flow and occlusions from our FlowNet3-CSS sets a new state of the art in motion segmentation.

9.6 Summary

This section has shown that:

- For CNNs, estimation of occlusions is very easy and joint estimation is not required,
- estimated occlusions for flow are state of the art and motion segmentation can benefit significantly from these occlusions,

(9.6.1)

- estimated motion boundaries for flow are state of the art,
- the stack for disparity estimation achieves state of the art in disparity estimation on KITTI 2015 and
- the stack for flow estimation achieves state of the art in flow estimation on KITTI 2012 and 2015.

(States of the art were reported for the time of publication at ECCV 2018.)

Chapter 10

Extending Training with Unlabeled Images

The concept of this chapter was developed together with Osama Makansi. The author also contributed the hinge loss for an unconstrained assessment metric.

The networks that have been introduced so far have only been trained on synthetic images, while they show surprisingly good results on other datasets, too. In contrast to semantic tasks, such as object detection or semantic segmentation, the optical flow estimation task seems to generalize very well. This is because learning the concept of correspondence is different from recognition and does not depend so much on the content of the images.

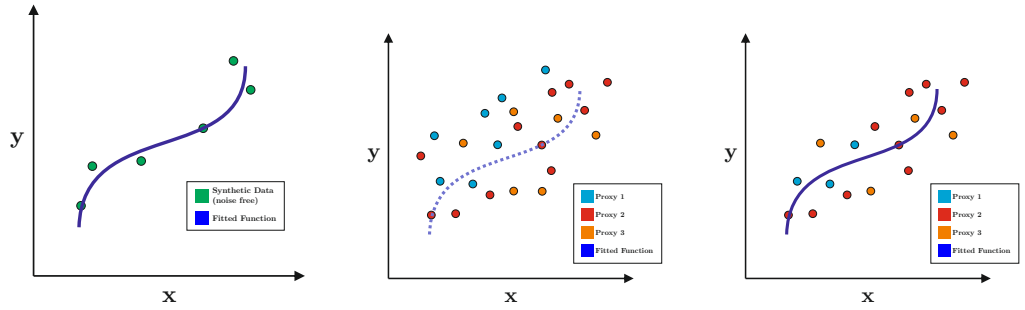
Still, knowledge about motion priors and correct apertures is very important. To this end, it would be most interesting to extend the training to real-world data. However, as explained in Chapter 5, optical flow is a secondary feature and cannot be directly captured by a sensor. As a dense, sub-pixel-accurate annotation by hand is not possible either, this raises the question if one can make use of existing methods that compute optical flow.

Approaches

Multiple strategies have been proposed on how to integrate real images into the training procedure. These span from using the same unsupervised training loss for the network, as is used in variational methods [AP16, MHR18], over multi-task learning with an auxiliary task that allows for learning from unlabeled images [SZB17] to training on *pseudo* ground truth obtained from running an (unsupervised) variational method [ZLNH17].

Proxies / Pseudo Ground-Truth

Here we will follow the last approach [ZLNH17] and fine-tune our networks on pseudo ground truth obtained from *proxy* methods. Note that our usual synthetic training datasets do not contain any noise (as shown in Figure 10.1a), while using a proxy introduces noise in form of errors in the optical flow (as shown in Figure 10.1b). As long as this noise does not introduce a general bias, the network can act as a regularizer and still converge to a good solution.



(a) The normal case is that the network fits a function to noise-free synthetic data. The function is not fitted perfectly, as the model is limited and the correct regularization parameters are unknown.

(b) If ground truth is not available, one can use several existing methods as proxies. These introduce noise from errors. Note that proxies 1 and 3 have biases.

(c) We filter the proxies with our proposed FusionNet so as to remove bias and to reduce noise. We use the FusionNet output to fine-tune FlowNet. These networks outperform the original FlowNets despite the noisy data. The network training acts as a regularization.

Figure 10.1: **Network acting as a regularizer on noisy data.**

FusionNet and Augmented-FlowNet

In order to minimize the noise and to make sure our pseudo ground truth does not contain a bias, we construct the latter by using a diverse set of proxies and fusing them with a network we call *FusionNet*. We then use the pseudo ground truth to fine-tune existing networks that were originally trained on synthetic data (as shown in Figure 10.1c). We call this fine-tuned network *AugmentedFlowNet*. Our results show that these AugmentedFlowNets outperform original FlowNet and that using the FusionNet with many proxies performs better than training on a single proxy alone – and in some cases even better than the proxies used to generate the data. In summary, we show that the network training acts as a regularization and that the noise introduced due to a single incorrect proxy is not significant. Therefore, one can in fact use the real data to improve the networks.

10.1 FusionNet

We assume that the various optical flow estimation methods have different strengths and weaknesses. This does not outrule that these methods may also have many difficulties in common which introduce noise to the training. However, as long as there are differences, we aim at removing potential biases of particular methods and want to choose the method that works best on a particular problem.

Assessment Network

To this end, we propose an *assessment* network that predicts the errors of the optical flow estimated by a set of existing methods (as shown in Figure 10.2; see next page) and is trained on synthetic data with available ground-truth optical flow. At the first glance, this training on synthetic images looks like we are back at square one. However, the task of the assessment is different from the task of flow estimation itself. First, we benefit from the information contained in the various input flow fields.

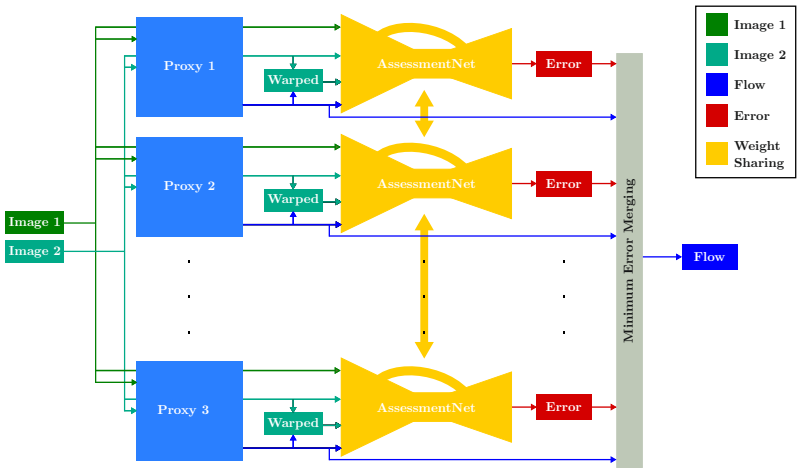


Figure 10.2: **Overview of the FusionNet principle.** Given the input images, the optical flow is estimated with various existing methods (proxies). Each proxy's optical flow estimate is used to warp the second image. The two input images, the warped image and the flow are fed into the proposed assessment network which is trained on predicting the error of each flow field. Finally, the flow fields are merged by locally choosing the flow vector with the minimum predicted error.

Second, the assessment task may generalize more easily to other domains than the one of optical flow estimation, since it must only find ways to predict errors rather than predicting the flow field itself.

The assessment network uses a typical encoder-decoder architecture with skip connections; the architecture details are the same as in FlowNetS. It takes the two input images into account in conjunction with the flow estimate and the second image warped by that flow. The error maps predicted by the assessment network are used to optimally combine the estimated flow fields. We refer to the complete setup shown in Figure 10.2 as FusionNet. We investigate two different loss functions: an L_1 loss and a hinge loss.

10.1.1 L_1 Loss

For training the assessment network with an L_1 loss, we let the network directly estimate the pixel-wise endpoint error (Equation 2.4.1) e by applying an L_1 loss to the ground truth e_{gt}:

$$\mathcal{L}_1(e) = |e_{\text{gt}} - e|. \tag{10.1.1}$$

In principle, one could train a separate, specialized network for each input method to be assessed. However, since we want to improve the generalization of the assessment network, we use the same network for assessing all input flows and rather sample the mini-batches from the different methods during training.

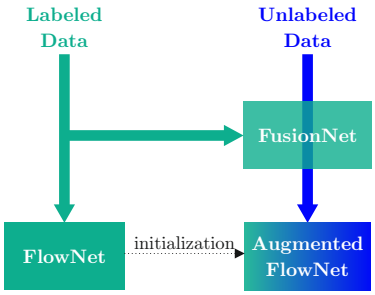


Figure 10.3: **Data domain transfer by using FusionNet.** Using our FusionNet to augment a FlowNet: FlowNet and FusionNet are trained on labeled data. Subsequently, FusionNet is used to augment FlowNet with large amounts of unlabeled data.

10.1.2 Hinge Loss

Directly applying an L_1 loss on the error makes the network estimate the error for each method. However, for the fusion, we only need to know the input methods with the lowest error. That means that the L_1 loss potentially solves a harder problem than necessary to reach the actual goal. A related problem to picking the input with the smallest error is the one of designing a distance metric in order to match patches. This metric only needs to reflect the ranking, e.g. "A is closer to B than A is to C" [SJ04, WS09b]. Many feature-learning algorithms use this as a triplet loss [XRK17, WSL+14, HA15b, SKP15, WL15]. With the same motivation, we use the well-known multi-class hinge loss [MD11, DGI16]

$$\mathcal{L}_{Margin}(e_1, ..., e_N) = \sum_{i \neq j} \max(0, m + e_j - e_i), \tag{10.1.2}$$

where j is the index of the method with the lowest error according to the ground truth, and m is the minimum margin between the best estimate and the other estimates. If the predicted best error corresponds to the true index j, and all other errors are larger than e_j by at least m, this loss will be zero. Otherwise, each error that is above the allowed margin will contribute to the loss. Since the network is allowed to rescale the errors, we can set $m = 1$ without loss of generality. Note that the errors predicted with this loss by the network do no longer correspond to the L_1 error but may be rescaled. The rescaling factor may even be different for each pixel. Obviously, the hinge loss implies joint training of the assessment network while giving all N methods as input.

10.2 Augmented FlowNet

Given the FusionNet from the last section, we can apply it to any unlabeled data to estimate high-quality optical flow. However, running FusionNet is very costly, since it requires running the various, partially very slow optical-flow-estimation proxy methods. In order to have a fast optical flow estimation at test time, we use the

optical flow fields estimated with FusionNet as proxy ground truth so as to fine-tune a FlowNet, for instance to optimize it for a specific domain or to make it run better on general real-world videos. The principle is illustrated in Figure 10.3 (see previous page).

10.3 Experiments

10.3.1 Datasets

In order to train the initial FlowNet and AssessmentNetwork, we use the two synthetic datasets FlyingChairs and FlyingThings3D from Chapter 5 and train with the schedules S_{long} and S_{fine} respectively. For the unsupervised fine-tuning, we use various unlabeled datasets that we grouped into two domains: animation movies and driving.

Animation movies

We collected several animation movies from the Blender project [Pro17] and used them for unsupervised training. Such animation movies bear the option to derive ground-truth optical flow, as shown in Butler et al. [BWSB12] and Mayer et al. [MIH+16], but we did not use this option here and rather used just the unlabeled videos for training. For the evaluation in this domain, we used the official Sintel benchmark dataset [BWSB12].

Driving

Driving scenes are a popular application domain for optical flow. For unsupervised training, we took approximately 100k frames from the Frankfurt part of the publicly available Cityscapes dataset [COR+16]. For the evaluation in this domain, we used the two publicly available KITTI2012 [GLU12] and KITTI2015 [MG15] benchmark datasets.

Motion Segmentation

For an indirect evaluation of the optical flow on a motion segmentation task, we used approximately 32k frames from the UdG-MS19 and UdG-MS20 datasets [MDSL17] for unsupervised training. We evaluated the motion segmentation on the FBMS benchmark dataset [OMB14].

10.3.2 FusionNet

We evaluated FusionNet with the following optical-flow-estimation techniques as input: LDOF [BM11], DeepFlow [WRHS13], EpicFlow [RWHS15], FlowFields [BTS15] and FlowNet2. There are some very recent methods with even better performances, such as DCFlow [XRK17], PWC-Net [SYLK18] and MR-Flow [WSLB17], but their code was not operational in time to include them for the experiments. A nice property of FusionNet is that new methods can be integrated trivially at any time to further improve results.

Performance compared to Proxies

Table 10.1 compares FusionNet to the state of the art on the common benchmark datasets. FusionNet consistently outperforms each of the techniques that have been provided as input, which demonstrates that the assessment network is able to locally select the best optical flow vectors. As a consequence, this brings it close to the most recent state of the art and would most likely outperform it if these methods were also included for selection. Table 10.1 also reports the results when selecting

	Method	Animation Domain				Driving Domain				
		Sintel *clean* AEE		Sintel *final* AEE		KITTI 2012 AEE		KITTI 2012 F1-noc	KITTI 2015 AEE	KITTI 2015 Fl-all
		train	*test*	*train*	*test*	*train*	*test*	*test*	*train*	*test*
Inputs	LDOF [BM11]	4.65	7.56	6.16	9.12	10.26	12.4	21.93%	17.71	–
Inputs	DeepFlow[WRHS13]	2.66	5.38	4.02	7.21	5.78	5.8	7.22%	13.14	28.48%
Inputs	EpicFlow[RWHS15]	2.27	4.11	3.57	6.28	3.52	3.8	7.88%	**9.23**	**26.29%**
Inputs	FlowNet2[IMS+17]	2.02	3.96	3.62	6.02	4.09	–	–	10.06	–
Inputs	FlowFields[BTS15]	**1.86**	**3.75**	**3.40**	**5.81**	**3.08**	**3.5**	**5.77%**	9.26	–
Best	DCFlow[XRK17]	–	3.54	–	5.12	–	–	–	–	14.86%
Best	PWC-Net[SYLK18]	2.55	4.39	3.93	**5.04**	**4.14**	**1.7**	**4.22%**	10.35	**9.60%**
Best	MR-Flow[WSLB17]	**1.83**	**2.53**	3.59	5.38	–	–	–	–	12.19%
Ours	FusionNet-L_1	1.59	–	**3.10**	–	3.11	–	–	**8.13**	–
Ours	FusionNet-Hinge	**1.58**	**3.20**	3.18	**5.50**	**2.97**	**3.6**	**5.50%**	8.18	**21.44%**
	Oracle	0.96	–	1.83	–	2.04	–	–	5.77	–

Table 10.1: **Comparison of FusionNet to the state of the art.** The upper section of the table corresponds to the input methods used for FusionNet. FusionNet performs better than any of the input methods. The "Oracle" fusion refers to a fusion based on the ground-truth error.

the flow vectors based on the ground-truth error ("Oracle"), indicating the limit that FusionNet could achieve with the respective optical flow fields given as input.

10.3.3 Augmented FlowNet

While FusionNet yields excellent optical flow that combines the best from all available methods, it requires 84 seconds per frame. In contrast, FlowNet2 runs at 8 frames per second[1].

Use of FusionNet Table 10.2 (see next page) first shows the influence of the choice of the proxy ground truth when fine-tuning a basic FlowNetC. Augmenting the FlowNet with an optical flow field that is superior to the baseline improves results, whereas inferior flow fields can decrease the performance. When using just a single proxy method, there arises the dilemma of which method to choose. As one would expect, feeding a random mixture of samples from various methods during fine-tuning (Rand. Mix) does not yield the best of all involved methods but approximately their average. In contrast, the use of FusionNet resolves the dilemma.

Domains We also distinguish between augmentation for a specific domain and generic augmentation. In the first case, we augment the FlowNet only on data from the respective domain, i.e. animation movies in case of Sintel and driving videos in case of KITTI; in the second case, data from both domains is used for fine-tuning.

Table 10.2 shows that domain-specific augmentation improves results considerably on KITTI, which is a very special scenario. The error is almost cut by half. However, the generic augmentation is not much worse either, as it also benefits from the training data from the special domain, even though it is now mixed with data from another

[1] FlowNet2 runtime is reported on an Nvidia GTX1080 GPU, while the traditional methods run on the CPU.

	Method	Animation Domain Sintel *clean*	Animation Domain Sintel *final*	Driving Domain KITTI 2012	Driving Domain KITTI 2015
	Baseline	3.07	4.46	6.66	12.47
Dom.	AugmentedFlowNetD-FlowNet2	**2.79**	**4.05**	4.26	9.60
Dom.	AugmentedFlowNetD-FlowFields	2.97	4.18	**3.62**	**8.01**
Dom.	AugmentedFlowNetD-EpicFlow	3.09	4.21	3.70	8.11
Dom.	AugmentedFlowNetD-DeepFlow	3.34	4.45	4.90	12.11
Dom.	AugmentedFlowNetD-LDOF	3.66	4.69	7.93	14.93
Dom.	AugmentedFlowNetD-Rand. Mix	3.07	4.20	4.14	9.61
Dom.	AugmentedFlowNetD-FusionNet (L1)	2.80	3.97	3.69	8.17
Dom.	AugmentedFlowNetD-FusionNet (Hinge)	**2.75**	**3.97**	**3.52**	**7.65**
Gen.	AugmentedFlowNetG-FusionNet (L1)	2.84	4.06	3.91	8.34
Gen.	AugmentedFlowNetG-FusionNet (Hinge)	**2.78**	**4.02**	**3.77**	**8.01**

Table 10.2: **Influence of the proxy ground truth on the augmented FlowNetC.** Average endpoint errors are reported on the training sets of Sintel and KITTI. Augmentation with a single proxy can improve results, but it is not obvious which method to choose. Using multiple methods to generate the proxy ground truth (FusionNet) yields consistent improvements across all benchmarks. The upper part of the table shows experiments which are trained on domain-specific data (denoted "AugmentedFlowNetD"), while the bottom part shows experiments where the training data came from multiple domains to yield a generic network (denoted "AugmentedFlowNetG").

domain. Obviously, the network can automatically figure out at test time from which domain the input is from and can apply the appropriate priors from that domain.

UnFlow Comparision

Table 10.3 extends the augmentation to a stacked FlowNet and compares it to UnFlow [MHR18]. UnFlow uses an unsupervised loss and can thus be specialized conveniently to any domain. The table shows results for UnFlow trained on CityScapes or the unlabeled data from KITTI, outperforming the supervised baseline which was trained on synthetic data outside this domain. For a better comparison to our strategy, we also report results for a semi-supervised version of UnFlow, i.e., it is initialized with a FlowNet trained on synthetic data before the unsupervised training starts.

Results show that the domain adaptation with the augmented FlowNet is clearly superior to the one of UnFlow[2]. As we already observed in Table 10.2, there is no significant difference between domain-specific training and training on a joint set of domains. This is also true for the stacked network.

[2] UnFlow does not require any supervision, which makes it biologically more plausible. From the engineering perspective, however, this is irrelevant.

	Method	Animation Domain Sintel *clean*	Animation Domain Sintel *final*	Driving Domain KITTI 2012	Driving Domain KITTI 2015
	Baseline	3.07	4.46	6.66	12.47
Dom.	UnFlow-C-CityScapes [MHR18]	–	–	5.08	10.78
Dom.	UnFlow-C-ours	4.22	5.12	5.01	11.07
Dom.	UnFlow-C-KITTIraw [MHR18]	–	–	3.78	8.80
Dom.	UnFlow-CS [MHR18]	–	–	3.30	8.14
Dom.	UnFlow-CSS [MHR18]	–	–	**3.29**	**8.10**
Dom.	AugmentedFlowNetD-C	2.75	3.97	3.52	7.65
Dom.	AugmentedFlowNetD-CS	2.20	3.45	2.35	5.84
Dom.	AugmentedFlowNetD-CSS	**2.09**	**3.34**	**2.05**	**5.35**
Gen.	AugmentedFlowNetG-C	2.78	4.02	3.77	8.01
Gen.	AugmentedFlowNetG-CS	2.21	3.49	2.44	5.90
Gen.	AugmentedFlowNetG-CSS	**2.10**	**3.38**	**2.17**	**5.18**

Table 10.3: **Comparison of augmented FlowNet stacks to UnFlow.** Augmented-FlowNetD-C and UnFlow-C-ours are trained on the same domain-specific data and are initialized with the same model (Baseline). The results show that the purpose of domain adaptation is better achieved with the augmentation based on FusionNet than with the unsupervised loss of UnFlow. The results for the FlowNet augmented with data from both domains even show that it is not necessary to train separate networks for each domain, but that a generic network augmented on both domains is equally good.

10.4 Benchmark Results

Table 10.4 (see next page) compares the stacked augmented FlowNet to the state of the art at time of publication on arXiv (Mon, 20 Aug 2018). On the KITTI benchmarks, the augmented FlowNet sets a new state of the art after being fine-tuned with the ground truth from the KITTI training set, too. But also the generic version, which has not been fine-tuned with ground truth data, yields very good results. The direct comparison to FlowNet2 quantifies the improvement on stacked networks due to the augmentation.

Interestingly, the stacked augmented FlowNet often even outperforms the FusionNet proxy. This is due to the fine-tuning with ground truth in case of the domain-specific network. Sometimes, the generic network is also better than FusionNet. This proves the ability of the network to act as a regularizer.

10.5 Results on Motion Segmentation

Finally, we also evaluated the augmented FlowNet in an application scenario. We augmented the FlowNet on data from UdG-MS19 as well as UdG-MS20 [MDSL17] and fed its optical flow into the motion segmentation approach by Keuper et al. [KAB15]. The motion segmentation performance was evaluated on the FBMS benchmark [OMB14].

Table 10.5 (see next page) shows that the adaptation to real images clearly helps a FlowNetC improve motion segmentation results. For the stacked network, our CS pipeline performs slightly better than the complete FlowNet2 stack.

10.6 Summary

This chapter has shown that:

(10.6.1)

- Network training acts as regularization and can lead to better results than the supplied pseudo ground truth,
- large amounts of unlabeled real-world data can be used to significantly improve FlowNet, and
- the presented approach of using FusionNet outperforms the state-of-the-art unsupervised approach UnFlow and achieves a new state of the art on both KITTI datasets.

	Method	Animation Domain				Driving Domain				
		Sintel *clean* AEE		Sintel *final* AEE		KITTI 2012 AEE		KITTI 2012 F1-noc	KITTI 2015 AEE	KITTI 2015 Fl-all
		train	*test*	*train*	*test*	*train*	*test*	*test*	*train*	*test*
Un/Semi	DSTFlow [RYN+17]	6.93	5.20	7.82	5.92	10.43	12.4	–	16.79	39.00%
	GAN-OpticalFlow[LHY17b]	3.30	6.27†	4.68	7.31†	7.16	6.8	–	16.02	31.01%
	Hybrid-OpticalFlow-NextFrame[SZB17]	–	–	–	–	5.31	9.2	39.12%	10.19	–
	UnFlow-CSS[MHR18]	–	–	7.91	10.22	3.29	1.7†	4.28%†	8.10	11.11%†
Sup	Our FlowNet2 [IMS+17]	2.02	3.96	3.62	5.74†	3.55	1.8†	–	8.94	11.48%†
	PWC-Net[SYLK18]	2.55	3.86†	3.93	**5.04**†	4.14	1.7†	4.22%†	10.35	9.60%†
	DCFlow[XRK17]	–	3.54	–	5.12	–	–	–	–	14.86%
	MR-Flow[WSLB17]	1.83	**2.53**	3.59	5.38	–	–	–	–	12.19%
Ours	Our AugmentedFlowNetD-CSS	2.09	4.22†	3.34	5.63†	**2.05**	**1.5**†	**3.97%**†	5.35	**8.57%**†
	Our AugmentedFlowNetG-CSS	2.10	3.69	3.38	5.20	2.17	2.7	7.04%	**5.18**	20.09%
	Our FusionNet-Hinge	**1.58**	3.20	**3.18**	5.50	2.97	3.6	5.50%	8.18	21.44%

Table 10.4: **Benchmark results for FusionNet and AugmentedFlowNet.** We compare to state-of-the-art methods at time of publication on arXiv (Mon, 20 Aug 2018). Numbers marked with † have been obtained after fine-tuning on the training set of the respective benchmark. On the KITTI benchmarks, we clearly extend the state of the art. Thanks to additional fine-tuning with ground-truth data, the augmented network even performs better than the FusionNet proxy; but also the generic version, which has only been fine-tuned on the FusionNet proxy, gets close to FusionNet and sometimes even outperforms it.

Method	F1 Measure	Extracted Objects
FlowNetC (baseline)	60.52%	10/69
AugmentedFlowNetD-C	65.29%	13/69
FlowNet2 [IMS+17] (stacked baseline)	76.72%	26/69
AugmentedFlowNetD-CS	**77.09%**	**28/69**

Table 10.5: **AugmentedFlowNet motion segmentation results.** Results are taken from the FBMS test set [OMB14]. We fed the optical flow from the listed methods into the motion segmentation approach by Keuper et al. [KAB15]. The augmentation on real data clearly improved over the FlowNetC baseline. For the stacked network, our CS pipeline performs slightly better than the complete FlowNet2 stack.

Chapter 11

Uncertainty Estimation

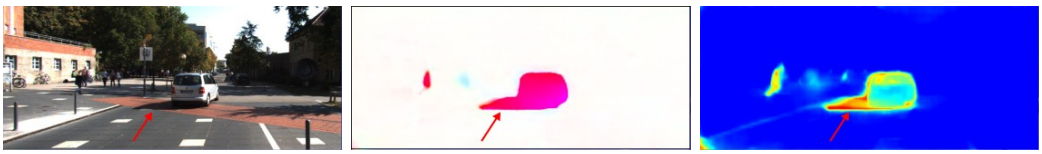
The contents of this chapter come from the joint work with Özgün Çiçek and Silvio Galesso, published in [IÇG+18]. The multi-hypotheses network named FlowNetH is a sole contribution by the author.

Black-Box Nature

A valid critique of learning-based approaches is their black-box nature: since all parts of the problem are learned from data, there is no strict understanding of how the problem is solved by the network. Although the results from Chapters 8, 9 and 10 show that the networks generalize well across various datasets and also yield good application performances, there is no guarantee that they will also work in different scenarios that contain unknown challenges or yield spurious results for badly conditioned input data.

Reliability for Applications

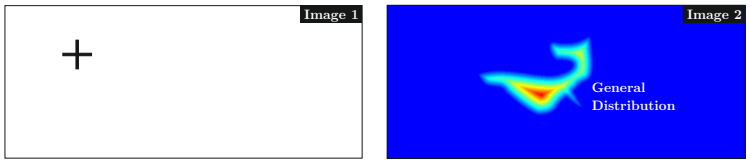
In real-world scenarios, such as control of an autonomously driving car, an erroneous decision can be fatal; thus, it is not possible to deploy such a system without information about how reliable the underlying estimates are. We should expect an additional estimate of the network's own uncertainty, such that the network can highlight hard cases where it cannot reliably estimate the optical flow or where it must decide between multiple probable hypotheses. An example is shown in Figure 11.1:



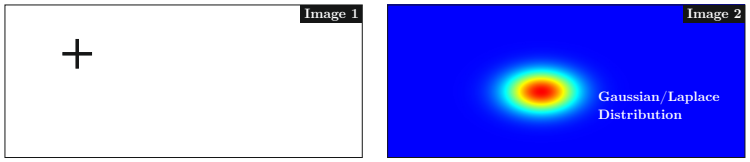
(a) Image (b) Flow prediction (c) Uncertainty prediction

Figure 11.1: **Example of joint estimation of optical flow and its uncertainty.** The estimated uncertainty (visualized as a heatmap) marks the optical flow in the shadow of the car as unreliable (pointed out by the red arrow), contrary to the car itself, which is estimated with higher certainty. Marked as most reliable is the optical flow for the static background.

(a) Given are two images. The correspondence for the location marked in the first image is to be determined in the second image. Common approaches provide only a single point estimate.



(b) The most general form would be to estimate the complete distribution over the second image.



(c) A simplification is to estimate parametric distributions, such as Gaussian or Laplace.

Figure 11.2: **Probabilistic formulation of correspondence.**

In the case of the transparent shadow, two apparent motions actually exist: the motion of the shadow and the motion of the scene. Distinguishing them requires semantic knowledge and special training data. Since the network was only trained on FlyingChairs and FlyingThings3D, it does not have the capability of distinguishing them and chooses to output the shadows motion. However, it can also recognize that the case is ambiguous and inform about its own uncertainty, as shown in Figure 11.1c.

Deep networks in computer vision typically yield only their single preferred prediction rather than the parameters of a distribution (as shown in Figure 11.2a). Even among traditional methods, only one method that performs among the state-of-the-art approaches for optical flow provides uncertainty estimates [WKR17]. In the following, we investigate how CNNs can estimate their local uncertainty about the correctness of their prediction for the regression setting, which is vital information when building decisions on top of the estimations and crucial for use in any real application. For the first time, we compare several strategies and techniques to estimate uncertainty in a large-scale computer vision task like optical flow estimation and introduce a multi-hypothesis approach.

11.1 Formulation of Uncertainty

Although the illustration from Figure 11.1c (see last page) is intuitive, a formal definition of uncertainty is not straightforward. The commonly accepted requirement is that an uncertainty measure must rank image pixels similar to ground-truth errors (see Section 3.6 and Figure 3.11).

While many approaches try to infer confidence measures from model parameters (Section 3.6) and others try to predict the ground-truth error or its negative exponential directly [UZU+17], here, we take the most general approach and predict probability distributions. Our work extends the work by Kendall and Gal [KG17].

Point Estimates

Assume we have a dataset $\mathcal{D} = \{(\mathbf{x}_0, \mathbf{y}_0^{\text{gt}}), \dots, (\mathbf{x}_N, \mathbf{y}_N^{\text{gt}})\}$ which is generated by sampling from a joint distribution $p(\mathbf{x}, \mathbf{y})$. In CNNs, it is normally assumed that there is a unique mapping from $\mathbf{x}$ to $\mathbf{y}$ by a function $f_{\mathbf{w}}(\mathbf{x})$ which is parametrized by weights $\mathbf{w}$ that are optimized according to a given loss function on a training dataset $\mathcal{D}$. In the case of optical flow, we denote the trained network as a mapping from the input images $\mathbf{x} = (\mathbf{I}_1, \mathbf{I}_2)$ to the output optical flow $\mathbf{y} = (\mathbf{u}, \mathbf{v})$ as $\mathbf{y} = f_{\mathbf{w}}(\mathbf{I}_1, \mathbf{I}_2)$, where $\mathbf{u}, \mathbf{v}$ are the x- and y-components of the flow field. This means that for each location in the first image, the predicted correspondence is a point estimate, as illustrated in Figure 11.2a (see previous page).

Probabilistic Formulation

If one now allows uncertainty, this implies that more or less likely alternative solutions must also exit. The result is a general probability distribution over all possible correspondence locations in the second image $p(\mathbf{y}|\mathbf{x}, \mathcal{D})$. An example is illustrated in Figure 11.2b (see previous page).

(11.1.1) The uncertainty is represented by how wide the distribution is spread or, in other words, by the distribution's entropy.

Uncertainty Estimate

In order to estimate the general uncertainty, the actual shape of the distribution is irrelevant. For this reason, we approximate the distribution by a parametric distribution, namely the Laplace distribution (as shown in Figure 11.2c; see previous page). We choose Laplace because its negative logarithm corresponds most closely to an L_1 distance, while a Gaussian would most closely correspond the the squared distance. This makes the Laplace distribution more robust to outliers and actually puts it closer to the endpoint error. The univariate Laplace distribution has two parameters a and b and is defined as:

$$\mathcal{L}(x|a, b) = \frac{1}{2b} e^{-\frac{|x-a|}{b}}. \tag{11.1.2}$$

The variance of this distribution is $\sigma^2 = 2b^2$ and serves as a measure for uncertainty. For optical flow, it is common to treat x- and y-components as independent [WKR17], which makes the distribution axis aligned. We also follow this approach:

$$\mathcal{L}(u, v|a_u, a_v, b_u, b_v) \approx \mathcal{L}(u|a_u, b_u) \cdot \mathcal{L}(v|a_v, b_v). \tag{11.1.3}$$

In general, one could also revert to an isotropic distribution. Because we use an axis-aligned distribution, we obtain two variances for x- and y-direction and use the entropy to obtain a single scalar for the uncertainty.

11.2 Sources of Uncertainty

Kendall and Gal [KG17] distinguish between two different sources of uncertainties:

Epistemic Uncertainty

Epistemic Uncertainty represents uncertainty about the correct model parameters $\mathbf{w}$, given the model f. This uncertainty can be explained away with an infinite amount of training data. Since this is not possible in practice, epistemic uncertainty naturally leads to modeling distributions over likely model parameters $p(\mathbf{w}|\mathcal{D}, f)$ which resemble the concept of Bayesian Neural Networks (BNNs). The uncertainty of the output values then has to be determined by marginalization (Equation 11.3.2) over the weights.

Aleatoric Uncertainty

The *aleatoric uncertainty* $p(\mathbf{y}|\mathbf{x})$ is a function of the input data $\mathbf{x}$ independent of any model as well as training data and accounts for the data's intrinsic characteristics, such as noise (e.g. sensor noise) or disturbed image areas (e.g. ambiguities, textureless regions, shadows, transparent objects and motion blur). For aleatoric uncertainty, multiple solutions are possible and the given input information is not sufficient to narrow them down further. However, it is still possible to estimate the uncertainty itself as a function of the input data. This uncertainty estimates the variance of the output $\mathbf{y}$, given very similar input cases $\mathbf{x}$. Note that according to [KG17] it does not depend on the training data.

Ambiguities

One should note that a separation of the two uncertainties can be very unclear. For example, consider two poles at the side of a road. If they look identical, there is no "guarantee" that they are not spuriously interchanged in the second image. This leads to aleatoric uncertainty. However, the uncertainty can significantly be reduced by the prior knowledge that the poles are attached to the road and do not move. This prior knowledge comes from the model and actually compensates the aleatoric uncertainty (but the definition of aleatoric uncertainty is just that it cannot be compensated by more data). In this case, one cannot draw a line between the two uncertainties.

If we supply only synthetic ground truth, the ground truth is noise-free (up to color interpolation due to sub-pixel positions). The only source of aleatoric uncertainty would arise due to ambiguities or occlusions. However, the FlyingChairs and FlyingThings3D datasets do not contain completely homogeneous areas and do not present ambiguous cases for a human. Up to the occlusions, this can be seen as an argument that only epistemic uncertainty is relevant.

Instead, from our results, we observe that a single network can also learn to explain that it cannot solve certain cases. Reasons might be that for this case, too few training samples were available and it learned to treat them as outliers, or because the model f is not powerful enough (both result in a bias/variance trade-off). As an example, let's say we define a model f that can estimate a maximum displacement of 200 pixels. If the data contains displacements up to $1,000$ pixels, a single network trained with the distribution output and loss attenuation that will be introduced in Section 11.3.2 will learn that these cases are outliers. Note that in reality, another model could solve them and that they do not induce aleatoric uncertainty. The consequence is that the single model can predict epistemic uncertainty (which again contradicts the definition of requiring a distribution). For these reasons, the separation of the two uncertainties is to be taken with much caution.

Uncertainty due to f

In fact, we will later see that modeling only uncertainty depending exclusively on the input $\mathbf{x}$ is sufficient and that these networks also predict epistemic uncertainty well (although they do contain any distributions over weights). What is missing in the definition of the two uncertainties are inaccuracies that neither come from the weights $\mathbf{w}$ nor the input data $\mathbf{x}$, but actually from the model definition f (where f includes the training procedure). This uncertainty arises due to stochastic gradient descent as well as the limited and hand-crafted model selection of f and its capacity, which lead to models that are not fully converged or simply have limited accuracy in general. The loss functions introduced by Kendall and Gal [KG17] account this uncertainty as aleatoric, although from a conceptual point of view one would consider it epistemic since it relates to the model f.

Empirical and Predictive

To avoid the confusion, in this work, we will therefore not use the terms epistemic and aleatoric but rather introduce the terms *empirical* and *predictive*:

(11.2.1) **Empirical uncertainty** comes from uncertainty about the model parameters $p(\mathbf{w}|\mathcal{D}, f)$, given the model f of limited capacity and a limited training dataset $\mathcal{D}$. Furthermore, we compute the empirical uncertainty only from a small set of samples $\mathbf{w}_i$ from $p(\mathbf{w}|\mathcal{D}, f)$. These are selected empirically or by coarse approximations of BNNs, as described in Section 11.3. The empirical uncertainty is an approximation of the epistemic uncertainty.

(11.2.2) **Predictive uncertainty** is uncertainty that arises from the input data $\mathbf{x}$ and the hand-crafted model f as well as its limitations. This uncertainty can be estimated by a single network, but in contrast to aleatoric uncertainty, it can also inform about the own model's limitations and those that arise from the convergence process.

11.3 Bayesian Neural Networks and Frequentist Approximations

(11.3.1) Bayesian Neural Networks (BNNs) replace the the deterministic network's weight parameters $\mathbf{w}$ with a distribution $p(\mathbf{w}|\mathcal{D}, f)$, where f represents the network architecture and $\mathcal{D}$ a finite and fixed training set.

Here, we will denote a network architecture as a *model* and the weights as the *model parameters* or the *paramter set*. Note that a BNN does *not* provide a distribution over all possible models but only over all possible parameter sets. In most cases, it is assumed that the hand-crafted model f is general enough, such that the resulting BNN can be seen as a substitute for all possible models, but this is not true per se. Instead, it is important to note that a BNN only provides a distribution over implementations of the model engineered by a human, which introduces a bias. In the following, if a distribution depends on $\mathbf{w}$, it implicitly also depends on f and we will omit the dependency on f for brevity.

Bayesian neural networks have been shown to obtain well-calibrated uncertainty estimates while maintaining the properties of standard neural networks [Nea96, Mac92]. While BNNs are easy to formulate, they are difficult to perform inference

in [KG17]. For the inference, one needs to marginalize over all possible sets of weights:

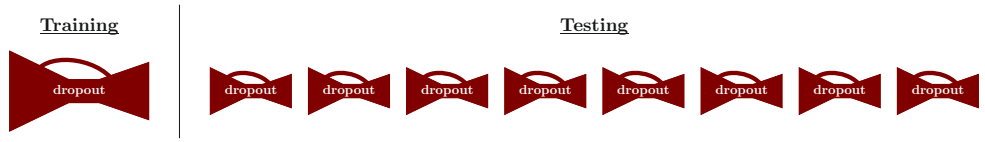
$$p(\mathbf{y} \mid \mathbf{x}, \mathcal{D}, f) = \int p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) p(\mathbf{w} \mid \mathcal{D}) d\mathbf{w} \,. \tag{11.3.2}$$

MCMC This is not possible in practice so approximations need to be made. Early work [Nea96] mostly used Markov Chain Monte Carlo (MCMC) methods to sample networks from the distribution of the weights, where some, for instance Hamiltonian Monte Carlo, can make use of the gradient information provided by the backpropagation algorithm. More recent methods generalize traditional gradient-based MCMC methods to the stochastic mini-batch setting, where only noisy estimates of the true gradient are available [CFG14, WT11]. However, even these recent MCMC methods do not scale well to high-dimensional spaces, and since contemporary encoder-decoder networks like FlowNet have millions of weights, they do not apply in this setting.

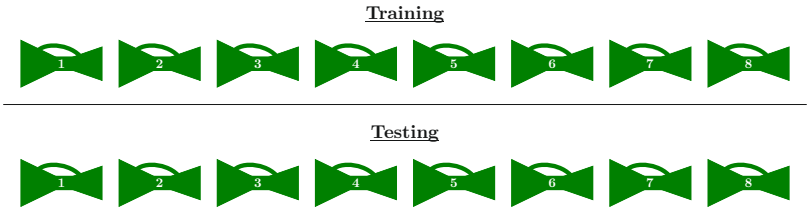
Variational Inference Instead of sampling, variational inference methods try to approximate the distribution of the weights by a more tractable distribution $q^*_\theta(\mathbf{w})$, parameterized by θ [KG17, Gra11, BCKW15, GG15]. This replaces the intractable problem of averaging over all weights in the BNN with an optimization task, where one seeks to optimize over the parameters of the simple distribution instead. A drawback is that approximating the true distribution by the simple one usually requires and independence assumption of the model parameters, which significantly affects the prediction quality. Even though they usually scale much better with the number of datapoints and weights than their MCMC counterparts, they have been applied only to much smaller networks [HA15a, BCKW15] than in the present work.

Dropout Dropout variational inference is a practical approach for approximate inference in large and complex models [GG15]. This inference is done by training a model with *dropout* before every weight layer and also by performing dropout at test time in order to sample from the approximate posterior (stochastic forward passes referred to as Monte Carlo dropout) [GG16a, KG17]. Dropout can be interpreted as a variational Bayesian approximation, where the approximating distribution $q^*_\theta(\mathbf{w})$ is a mixture of two Gaussians with small variances and the mean of one of the Gaussians is fixed at 0 [KG17]. We extend upon this work, but show that the dropout strategy is not the best solution and other strategies yield better results.

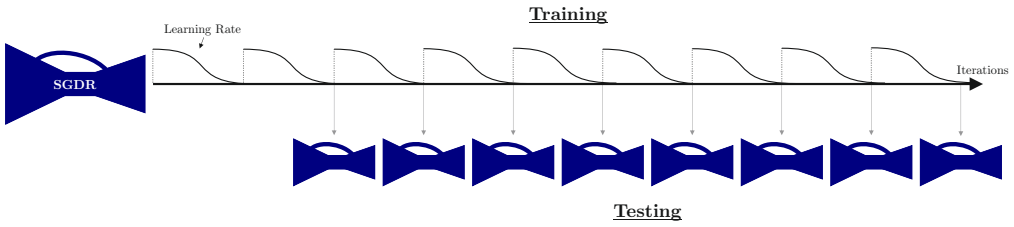
Boot-strapping Obtaining distributions over network weights or outputs is also possible by frequentist methods, which tend to be much simpler to implement than Bayesian approaches but do not follow a formal framework. One such method is *bootstrapping* [Bis06] where M different models are trained with different initialization and on M different bootstrapped subsets of the training data. The expectation is that this will provide independent samples from the weight distribution. The approach is easy to implement and scales nicely to high-dimensional spaces, since it only requires point estimates of the weights. While bootstrapping does not ensure diversity of the models and in the worst case can lead to M identical models, Lakshminarayanan et al. [LPB16] argued that ensemble model averaging can be seen as dropout averaging. They trained individual networks with random initialization and random data shuffling where each network predicts a mean and a variance. During test time, they combined the individual model predictions to account for the empirical uncertainty of the network.



(a) **Dropout:** Dropout randomly zeros out activations and is applied during training and testing. This can also be seen as generating an ensemble during test time on-the-fly.



(b) **Bootstrapping:** An ensemble is created by training different instances completely separately. The instances are equipped with different initializations and use different data subsets.



(c) **SGDR:** Ensemble members are obtained as snapshots. The training is repeatedly restarted, aiming at visiting different local minima during the convergence process.

Figure 11.3: **Overview of ensemble generation approaches.**

SGDR

We also consider so-called *snapshot ensembles* [HLP17] in our experiments. This approach follows the seemingly contradictory goal of learning an ensemble of multiple neural networks without incurring any additional training costs. The idea is to use the nonconvex nature of neural networks and the ability of SGD to converge to and escape from local minima. To this extent, the learning rate is repeatedly annealed and raised again (referred to as restarts). This procedure is called *Stochastic Gradient Descent with warm Restarts* (*SGDR*) [LH17].

In summary, existing work provides three different methods to obtain different sets of weights for CNNs: 1.) sampling via dropout, 2.) bootstrapping and 3.) SGDR. While the first one follows the Bayesian framework, the other two are frequentist methods. An overview is provided in Figure 11.3.

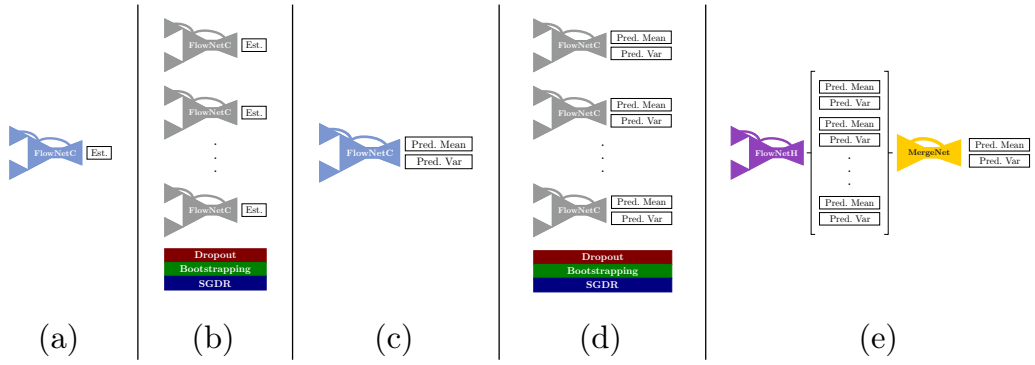


Figure 11.4: **Networks for uncertainty prediction. (a)** FlowNetC trained with EPE (no uncertainty). **(b)** Same network as (a), where an ensemble is built by using dropout, bootstrapping or SGDR. **(c)** FlowNetC trained with negative-log-likelihood to predict mean and variance. **(d)** Same network as (c), where an ensemble is built by using dropout, bootstrapping or SGDR. **(e)** FlowNetH trained to predict multiple hypotheses with variances, which are merged to a single distributional output.

11.3.1 Empirical Uncertainty Estimation

Modelling empirical uncertainty implies a transition from a single network (Figure 11.4a) to an ensemble (Figure 11.4b) by using M different model parameter sets, such that the mean and the variance of the distribution $p(\mathbf{y}|\mathbf{x}, \mathcal{D}, f)$ can be approximated with the empirical mean and the variance of the individual networks's predictions. Let $f_{\mathbf{w}_i}(\mathbf{x})$ denote the model f with parameter set $\mathbf{w}_i$ of an ensemble of M networks with outputs $\mathbf{u}_{\mathbf{w}_i}$ and $\mathbf{v}_{\mathbf{w}_i}$ respectively. The different model parameterizations $\mathbf{w}_i$ can be generated with the approaches from the last section (dropout, bootstrapping and SGDR) and reflect samples from the distribution of all possible model parameterizations $p(\mathbf{w}|\mathcal{D})$. We can then compute the empirical mean and the variance for the $\mathbf{u}$-component of the optical flow by:

$$\boldsymbol{\mu}_{\mathbf{u}} = \frac{1}{M} \sum_{i=1}^{M} \mathbf{u}_{\mathbf{w}_i}(\mathbf{x}) \,, \tag{11.3.3}$$

$$\boldsymbol{\sigma}_{\mathbf{u}}^2 = \frac{1}{M} \sum_{i=1}^{M} (\mathbf{u}_{\mathbf{w}_i}(\mathbf{x}) - \boldsymbol{\mu}_{\mathbf{u}})^2 \,, \tag{11.3.4}$$

and accordingly for the $\mathbf{v}$-component.

11.3.2 Predictive Uncertainty Estimation

For predictive uncertainty estimation, we need a single model generating a prediction of $\mathbf{y}$ and its associated uncertainty from the input data $\mathbf{x}$. Formally, we extend the model f to a model ϕ to output the parameters $\boldsymbol{\theta}$ of a parametric distribution (in our case, mean and variance of a Laplace distribution as motivated by Section 11.1) that should approximate $p(\mathbf{y}|\mathbf{x}, \mathcal{D})$ [NW94]. Note that this models a distribution over network *outputs* instead of *weights*. Such networks can be optimized by maximizing

their log-likelihood:

$$\mathbf{w}^* = \operatorname{argmax}\left[\log p(\mathcal{D} \mid \mathbf{w}, \phi)\right] = \operatorname{argmax}\left[\frac{1}{N}\sum_{i=1}^{N} \log p(\mathbf{y}_i \mid \phi_{\mathbf{w}}(\mathbf{x}_i))\right] \tag{11.3.5}$$

and the predictive distribution for an input $\mathbf{x}$ is then defined as:

$$p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}, \phi) \equiv p(\mathbf{y} \mid \phi_{\mathbf{w}}(\mathbf{x}))\,. \tag{11.3.6}$$

By using the Laplace distribution from Equation 11.1.2, we obtain a version of FlowNet with outputs a_u, a_v, b_u, b_v by minimizing the negative log-likelihood of Eq. 11.1.3:

$$-\log(\mathcal{L}(u|a_u, b_u) \cdot \mathcal{L}(v|a_v, b_v)) = -\log(\mathcal{L}(u|a_u, b_u)) - \log(\mathcal{L}(v|a_v, b_v))) \tag{11.3.7}$$

$$= \frac{|u - a_u|}{b_u} + \log b_u + \frac{|v - a_v|}{b_v} + \log b_v. \tag{11.3.8}$$

This case corresponds to a single FlowNetC predicting flow and predictive uncertainty as illustrated in Figure 11.4c (see page 104). Let us consider the u-component:

$$-\log(\mathcal{L}(u|a_u, b_u)) = \frac{\overbrace{|u - a_u|}^{L_1 \text{ distance}}}{\underbrace{b_u}_{\text{attenuation}}} + \underbrace{\log b_u}_{\text{trade-off}}\,. \tag{11.3.9}$$

Loss Attenuation

The residual error term $|u - a_u|$ corresponds to an L_1 loss, i.e., the loss function will enforce means a_u that are close to the ground-truth u. Furthermore, the residual term is divided by the scale parameter b_u (corresponding to the variance $\sigma^2 = 2b^2$). If the network is unable to solve for low error terms (either due to noise in $\mathbf{x}$ or due to limitations of f), the network has the option to output large values for b_u. This is called *loss attenuation*. According to [KG17], this acts similar to a robust regression function.

Trade-Off

Finally, the logarithm term acts as a trade-off and discourages the model to explain data with noise. This term makes sure that the L_1 distance is actually minimized and large b_u are only predicted when really necessary. It is important to note that the trade-off between a_u and b_u is not arbitrarily chosen but comes from the formulation of the loss with a Laplace distribution and thus also enforces the errors of the model to follow this distribution.

11.3.3 Bayesian Uncertainty Estimation

We can now use the network with the probabilistic output from the last section and insert it into Equation 11.3.2 to obtain a Bayesian formulation:

$$p(\mathbf{y} \mid \mathbf{x}, \mathcal{D}, \phi) = \int p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) p(\mathbf{w} \mid \mathcal{D}) d\mathbf{w} \tag{11.3.10}$$

$$= \int p(\mathbf{y} \mid \phi_{\mathbf{w}}(\mathbf{x})) p(\mathbf{w} \mid \mathcal{D}) d\mathbf{w}. \tag{11.3.11}$$

This integral cannot be computed in closed form, but by sampling M networks $\mathbf{w}_i \sim p(\mathbf{w}|\mathcal{D})$ from the posterior distribution and using a Monte-Carlo approximation [Nea96], we can approximate it as:

$$p(\mathbf{y} \mid \mathbf{x}, \mathcal{D}, \phi) \approx \sum_{i=1}^{M} p(\mathbf{y} \mid \phi_{\mathbf{w}_i}(\mathbf{x})). \tag{11.3.12}$$

Note that this represents M different networks and therefore leads to an ensemble that effectively combines the empirical and predictive uncertainty estimation. If we assume that the single network's output $\boldsymbol{\theta}_i = \phi_{\mathbf{w}_i}(\mathbf{x})$ consists of a mean and a variance $\boldsymbol{\theta}_i = (\boldsymbol{\sigma}_i^2, \boldsymbol{\mu}_i)$, we can use the law of total variance to compute mean and variance of the mixture distribution from Equation 11.3.12 [KG17]:

$$\boldsymbol{\mu}_{\mathbf{u}} = \frac{1}{M} \sum_{i=1}^{M} \boldsymbol{\mu}_{\mathbf{u},i}\,, \tag{11.3.13}$$

$$\boldsymbol{\sigma}_{\mathbf{u}}^2 = \frac{1}{M} \sum_{i=1}^{M} \left((\boldsymbol{\mu}_{\mathbf{u},i} - \boldsymbol{\mu}_{\mathbf{u}})^2 + {\boldsymbol{\sigma}^2}_{\mathbf{u},i} \right) \quad . \tag{11.3.14}$$

Combined Predictive / Empirical

Equations for the $\mathbf{v}$-component are analogous. We now have combined the approaches from Section 11.3.1 and Section 11.3.2 as illustrated in Figure 11.4d (see page 104), i.e., we now have combined the predictive and the empirical approach. This was initially proposed by Kendall and Gal [KG17] by using dropout variational inference. In this work, we apply the approach to optical flow and generate the ensemble members with the three approaches dropout, bootstrapping and SGDR. We compare all approaches in Section 11.5.

11.4 Predicting Multiple Hypotheses within a Single Network

The consequence of approximating BNNs is that multiple network forward passes with different sets of weights have to be performed (Figures 11.4b and 11.4d). This results in a drawback due to highly increased computational costs at runtime.

Hypotheses Estimation

In this section, we explain how to apply the Winner-Takes-All (WTA) loss in order to make multiple predictions within a single network [GRBK12, LPC+16, CK17, RLD+17], which can be seen as an ensemble within one network. We call the outputs of this network *hypotheses*. The WTA loss makes the hypotheses more diverse and leads to capturing more different solutions. However, because of the WTA loss, these solutions are not sampled from the distribution of the solution space anymore, and it is not possible to obtain the best solution by simply averaging, as has been done for the ensembles presented in the last section. We propose to use a second network to perform the merging task of the hypotheses to a single prediction and variance, as depicted in Figure 11.4e (see page 104). The second network is simply trained to regress the hypotheses to the optical flow ground truth.

Winner-Takes-All

Since a ground truth is available only for the single true solution, the question arises how to train a network in order to predict multiple hypotheses and how to ensure that each hypothesis encompasses meaningful information. To this end, we use a loss that

punishes only the best among the network output hypotheses $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M$ [GRBK12]. Let the loss between a predicted flow vector $\mathbf{y}(i,j)$ and its ground truth $\mathbf{y}^{\mathrm{gt}}(i,j)$ at pixels i,j be defined by a loss functon l. We minimize:

$$L_{hyp} = \sum_{i,j} l(\boldsymbol{y}_{\mathrm{best\_idx}(i,j)}, \boldsymbol{y}^{\mathrm{gt}}(i,j)) + \Delta(i,j)\,, \tag{11.4.1}$$

where best_idx(i,j) selects the best hypothesis per pixel according to the ground truth:

$$\mathrm{best\_idx}(i,j) = \underset{k}{\mathrm{argmin}} \left[EPE(\boldsymbol{y}_k(i,j), \boldsymbol{y}^{\mathbf{gt}}(i,j))\right]\,. \tag{11.4.2}$$

$\Delta = \Delta_u + \Delta_v$ encourages similar solutions to be in the same hypothesis k via one-sided differences, e.g. for the $\mathbf{u}$ component:

$$\begin{aligned}\Delta_u(i,j) = &\sum_{k;i>1;j} |y_{k,u}(i,j) - y_{k,u}(i-1,j)| + \\ &\sum_{k;i;j>1} |y_{k,u}(i,j) - y_{k,u}(i,j-1)|\,.\end{aligned} \tag{11.4.3}$$

For l, we either use the endpoint error from Eq. 2.4.1 or the negative log-likelihood from Eq. 11.3.7. In the latter case, each hypothesis is combined with an uncertainty estimation, and l also operates on a variance $\boldsymbol{\sigma}$. Equations 11.4.2 and 11.4.3 remain unaffected. For the best index selection, we stick to the EPE since this is the main optimization goal.

Encouraging Diversity In order to minimize L_{hyp}, the network must make a prediction close to the ground truth in at least one of the hypotheses. In locations where multiple solutions exist and the network cannot decide for one of them, it will predict several different likely solutions so as to increase the chance that the true solution is among these predictions. Consequently, the network will favor making diverse hypotheses in cases of uncertainty. In Tables 3 and 4 of the supplemental material of [IÇG+18], we provide visualizations of such hypotheses.

Collapsing In principle, L_{hyp} could collapse to using only one of the hypotheses' outputs. In this case, the other hypotheses would have very large error and would never be selected for backpropagation. However, due to the variability in the data and the stochasticity in training, such collapsing is very unlikely. We never observed that one of the hypotheses was not used by the network, and for the oracle merging, we observed that all hypotheses contribute more or less equally. We show this diversity in our experiments.

11.5 Experiments

In order to evaluate the different strategies for uncertainty estimation while keeping the computational costs tractable, we chose the FlowNetC architecture with improved training settings as a base model. Note that this section aims at comparing uncertainty

t

	Iter.	EPE
FlowNet2-C	600k	3.77
FlowNet2-C	1.2m	3.58
FlowNetC ours	600k	3.40

Table 11.1: **Improved FlowNetC settings.** Optical flow quality on Sintel train clean with the original FlowNet2-C (Section 8.1) and our implementation.

estimation techniques and not at improving the optical flow over the base model. The use of ensembles may lead to minor improvements of the optical flow estimates due to the averaging effect, but these improvements are not of major concern here. In the end, we will also show results for a large stacked network to demonstrate that the uncertainty estimation as such is not limited to small, simple networks.

11.5.1 Training Details

In contrast to FlowNet2-C, we use Batch Normalization [IS15] and a continuously dropping cosine learning-rate schedule [LH17]. This yields shorter training times and improves the results a little (see Table 11.1). We train on FlyingChairs and start with a learning rate of $2e-4$. For all networks, we fix a training budget of $600k$ iterations per network, with an exception for SGDR where we also evaluate performing some pre-cycles. For SGDR ensembles, we perform restarts every 75k iterations. We fix the T_{mult} to 1 so that each annealing takes the same number of iterations. We experiment with different variants of building ensembles by using snapshots at the end of each annealing. We always take the latest M snapshots when building an ensemble. For dropout experiments, we use a dropout ratio of 0.2 as suggested by Kendall and Gal [KG17]. For bootstrapped ensembles, we train M FlowNetC in parallel with bootstrapping, such that each network sees different 67% of the training data.

Ensemble Size

For the ensembles, we must choose the size M of the ensemble. The sampling errors for the mean and the variance decrease with increasing M. However, since networks for optical flow estimation are quite large, we are limited in the tractable sample size and restrict it to $M = 8$. We also use $M = 8$ for FlowNetH.

For SGDR, there is an additional pre-cycle parameter: in the beginning, snapshots have usually not yet converged, and the number of pre-cycles is the number of snapshots we discard before building the ensemble. In the supplemental material of [IÇG+18], we show that the later the snapshots are taken, the better the results are in terms of EPE and AUSE. We use eight pre-cycles in the following experiments.

11.5.2 Evaluation Metrics

Sparsification Plots

In order to assess the quality of the uncertainty measures, we use so-called *sparsification plots*, which were already introduced in Section 3.6 and are commonly used for this purpose [AHPB13, WKR17, KMG08, KN11]. Such plots reveal to which extent the estimated uncertainty coincides with the true errors. If the estimated variance

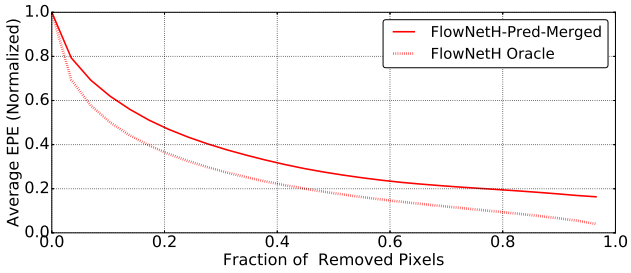


Figure 11.5: **Sparsification plot of FlowNetH-Pred-Merged.** Evaluated on the Sintel train clean dataset. The plot shows the average endpoint error (AEPE) for each fraction of pixels having the highest uncertainties removed. The oracle sparsification shows the lower bound by removing each fraction of pixels ranked by the ground-truth endpoint error. Removing 20% of the pixels results in halving the average endpoint error.

is a good representation of the model uncertainty and the pixels with the highest variance are removed gradually, the error should decrease in a monotonous manner. Such a plot of our method is shown in Figure 11.5. The best possible ranking of uncertainties is done by the true error to the ground truth. We refer to this curve as *oracle sparsification*. Figure 11.5 reveals that our uncertainty estimate is very close to this oracle.

Sparsification Error

For each approach, the oracle is different, hence a comparison among approaches by using a single sparsification plot is not possible. To this end, we introduce a measure which we call *sparsification error*. It is defined as the difference between the sparsification and its oracle. Since this measure normalizes the oracle out, a fair comparison of different methods is possible. In Figure 11.6a, we show sparsification errors for all methods we present in this paper. In order to quantify the sparsification error with a single number, we use the Area Under the Sparsification Error curve ($AUSE$).

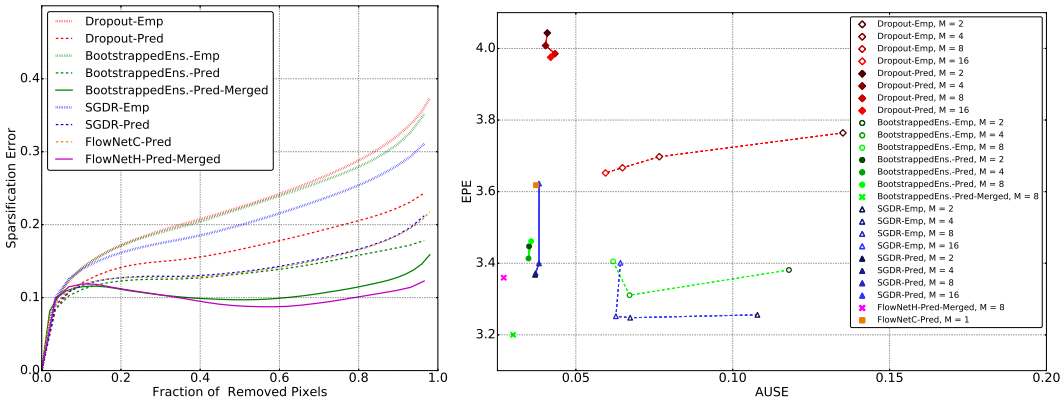
Oracle Epe

For each ensemble, we also compute the hypothetical endpoint error by considering the pixel-wise best selection from each member (decided by the ground truth). We call this error *Oracle EPE* and report it in combination with the empirical variances among the members in Table 11.2.

11.5.3 Comparison among Uncertainties from CNNs

(11.5.1)

Nomenclature: When a single network is trained against the endpoint error, we refer to it and the resulting ensemble as empirical (abbreviated as *Emp*; Figures 11.4a and 11.4b), while when the single network is trained against the negative log-likelihood, we refer to it and the ensemble as predictive (*Pred*; Figures 11.4c and 11.4d). When multiple samples or solutions are merged with a network, we add *Merged* to the name. E.g., FlowNetH-Pred-Merged refers to a FlowNetH that predicts multiple hypotheses and merges them with a network, using the loss for a predictive distribution for both hypotheses and merging respectively (Figure 11.4e).



(a) (b)

Figure 11.6: **Graphic evaluation of uncertainty estimation approaches. (a)** Sparsification error on the Sintel train clean dataset. The sparsification error (smaller is better) is the proposed measure for comparing the uncertainty estimates between different methods. FlowNetH-Pred-Merged and BootstrappedEnsemble-Pred-Merged perform best in almost all sections of the plot. **(b)** Scatter plot of EPE vs. AUSE for the tested approaches visualizing some content of Table 11.2.

	empirical (Emp)				predictive (Pred)				
	AUSE	EPE	Oracle EPE	Var.	AUSE	EPE	Oracle EPE	Var.	Runtime
FlowNetC	-	3.40	-	-	0.133	3.62	-	-	**38**ms
Dropout	0.212	3.67	2.56	5.05	0.158	3.99	2.96	3.80	320ms
SGDREnsemble	**0.191**	3.25	2.56	3.50	0.134	3.40	2.87	1.52	304ms
BootstrappedEnsemble	0.209	3.41	2.17	9.52	0.127	3.46	2.49	6.15	304ms
BootstrappedEnsemble-Merged					0.102	3.20	2.49	6.15	332ms
FlowNetH-Merged	-	3.50	1.73	**83.32**	**0.095**	3.36	1.89	**52.85**	60ms

Table 11.2: **Quantitative evaluation of uncertainty estimation approaches.** Comparison of flow and uncertainty predictions of all proposed methods with $M = 8$ on the Sintel train clean dataset. Oracle-EPE is the EPE of the pixel-wise best selection from the samples or hypotheses determined by the ground truth. Var. is the average empirical variance over the eight samples or hypotheses. Predictive versions (Pred) generally outperform empirical versions (Emp). Including a merging network increases the performance. FlowNetH-Pred-Merged performs best for predicting uncertainties and has a comparatively low runtime.

The results for all methods are summarized in Table 11.2 as well as Figures 11.6a and 11.6b and will be discussed in the following.

11.5.3.1 Empirical Uncertainty Estimation

The results show that uncertainty estimation with empirical ensembles is possible but worse than the other methods presented here. However, in comparison to predictive counterparts, empirical ensembles tend to yield slightly better EPEs, as will be discussed in the following.

11.5.3.2 Predictive Uncertainty Estimation

(11.5.2) The estimated uncertainty is better with predictive models than with the empirical ones. Even a single FlowNetC with predictive uncertainty yields much better uncertainty estimates than any empirical ensemble in terms of AUSE. The experiments confirm that it is advantageous to let a network estimate its own uncertainty and that the predictive uncertainty due to $\mathbf{x}$ and f is much more important than the approximated epistemic uncertainty due to $\mathbf{w}$.

This is because when training against a predictive loss function, the network has the possibility to explain outliers with the uncertainty (see Section 11.3.2). While the EPE loss tries to enforce correct solutions for outliers, too, the log-likelihood loss attenuates them. This also leads to slightly worse results for the EPE.

11.5.3.3 Predictive Ensembles

(11.5.3) Comparing ensembles of predictive networks to a single predictive network shows that the latter is already very close to the predictive ensembles and that the benefit of an ensemble is limited. This concludes that an epistemic uncertainty approximation over the weights $\mathbf{w}$ is not really relevant for FlowNet.

We also attribute this to loss attenuation: different ensemble members appear to attenuate outliers in a similar manner and induce less diversity, as can be seen by the variance between the members of the ensemble (column "Var." in Table 11.2). When comparing empirical to predictive ensembles, we can draw the following conclusions:

(11.5.4)
- Empirical estimation provides more diversity within the ensemble (variance column in Table 11.2).
- Empirical estimation provides lower EPEs and Oracle EPEs.
- All empirical setups provide worse uncertainty estimates than predictive setups.

Ensemble Types Analyzing the different types of ensembles, we see that the commonly used dropout technique [GG16a] performs worst in terms of EPE and AUSE, although the differences between the predictive ensemble types are not very large. SGDR ensembles provide better uncertainties, yet the variance among the samples is the smallest one. This is likely because later ensemble members are derived from previous snapshots of the same model. Furthermore, because of the eight pre-cycles, SGDR experiments ran for the largest number of training iterations, which could be an explanation why they provide a slightly better EPE than other ensembles. Bootstrapped ensembles provide the highest sample variance and the lowest AUSE among the predictive ensembles.

FlowNetH and Merging We finally analyze FlowNetH and uncertainty estimation with merging networks. In order to obtain a fair comparison, we also investigated putting a merging network on top of the predictive bootstrapped ensembles. Results show that the multi-hypotheses network (FlowNetH-Pred-Merged) is on par with BootstrappedEnsemble-Pred-Merged in terms of AUSE and EPE. However, including the runtime, FlowNetH-Pred-Merged yields the best trade-off (see Table 11.2 on previous page). Only FlowNetC and FlowNetH-Pred-Merged allow a deployment at interactive frame rates. Table 11.2 also

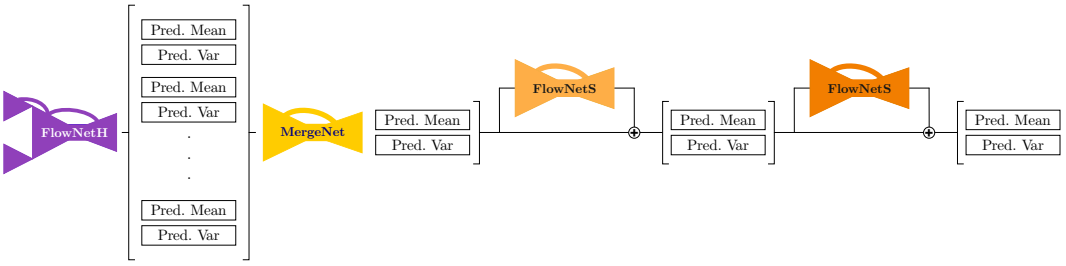


Figure 11.7: **Full flow and uncertainty estimation stack.** The first two networks are FlowNetH and the merging network as described in Section 11.4. The two stacked FlowNetS architecture networks estimate residual refinements as in Chapter 9.

shows that FlowNetH has a much higher sample variance and the lowest oracle EPE. This indicates that it internally has very diverse and potentially useful hypotheses that could be exploited even better in the future. For some visual examples, we refer to Tables 3 and 4 in the supplemental material of [IÇG+18].

11.5.4 Qualitative Evaluation

We compare the favored approach from the previous section (FlowNetH-Pred-Merged) to ProbFlow [WKR17], which is an energy-minimization approach and currently the state of the art for estimating the uncertainty of optical flow. Figure 11.8 (see next page) shows a qualitative comparison. A general observation is that ProbFlow predicts uncertainties mainly at image boundaries, which are the locations that most violate the smoothness assumption in the energy term. In contrast, FlowNetH-Pred-Merged predicts uncertain regions very well.

There is a lot of ambiguity in Figure 11.8a. Some background is visible between the slats at the top, and the flow is still inferred correctly. At the bottom, the ambiguity is too large and the flow is inferred incorrectly, which is also reflected by the predicted uncertainty. In Figure 11.8b, our approach correctly predicts uncertainty due to transparent windows, shadows and motion blur caused by rotating tires. Figure 11.8c shows a homogeneous board. Interestingly, the CNN approach is able to consider the correct aperture and to predict the flow correctly, while it also correctly predicts a high uncertainty due to the weakness of information. The remaining figures show some more interesting examples. Notably, ProbFlow often misses objects in the uncertainty prediction. In Figure 11.8h, it misses the ball entirely in both the flow and the entropy, while our approach is able to capture it and to predict a high uncertainty due to the motion blur.

For the full videos of the real-world dataset and further comments, please see the supplementary video which can be found on `https://youtu.be/HvyovWSo8uE`. A quantitative evaluation will be given in the next section.

11.5.5 Network Stacks and Benchmark Results

We now proceed to building the full network stack for FlowNetH-Pred-Merged, as illustrated in Figure 11.7. We denote this stack as FlowNetH-Pred-Merged-

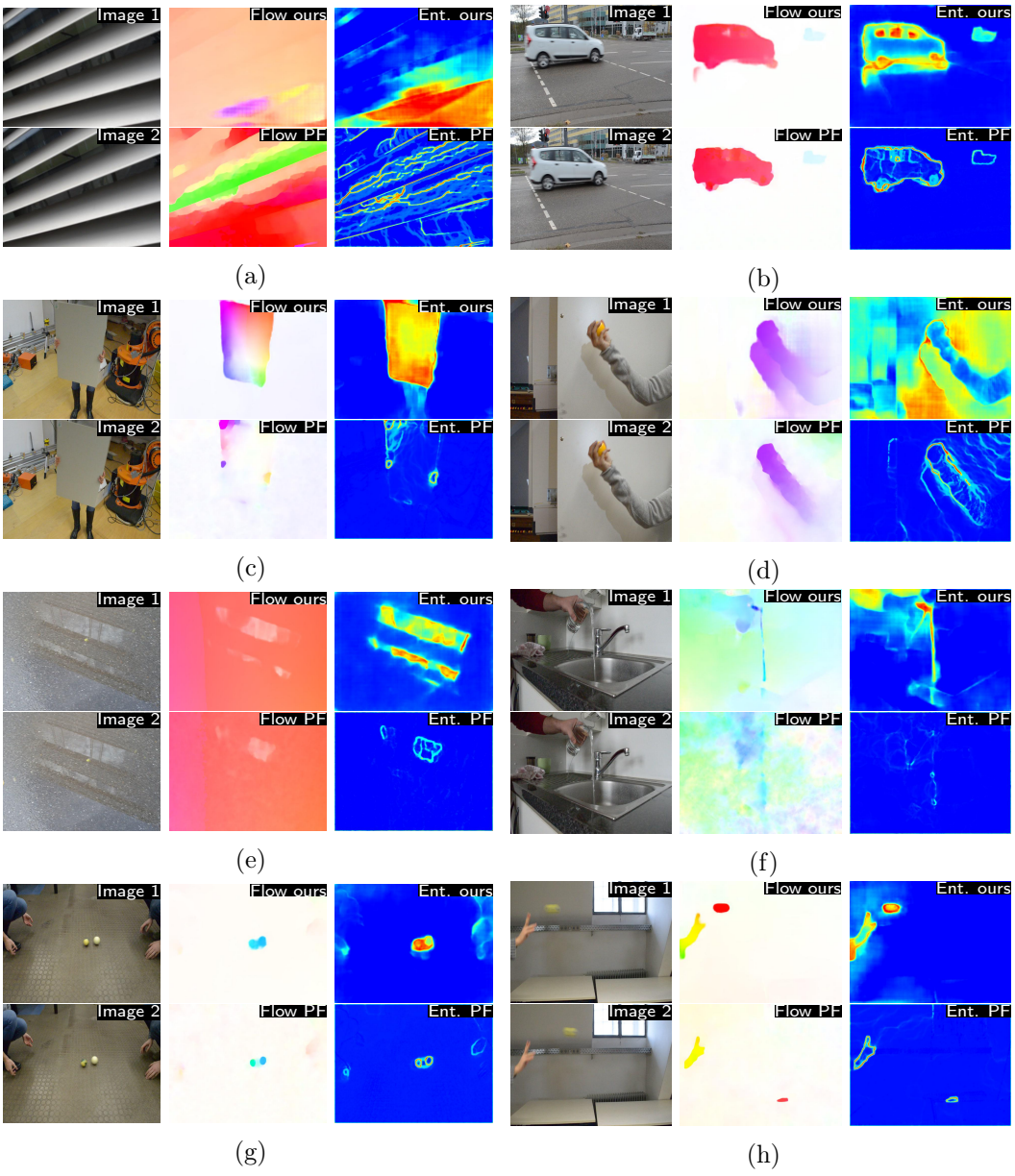
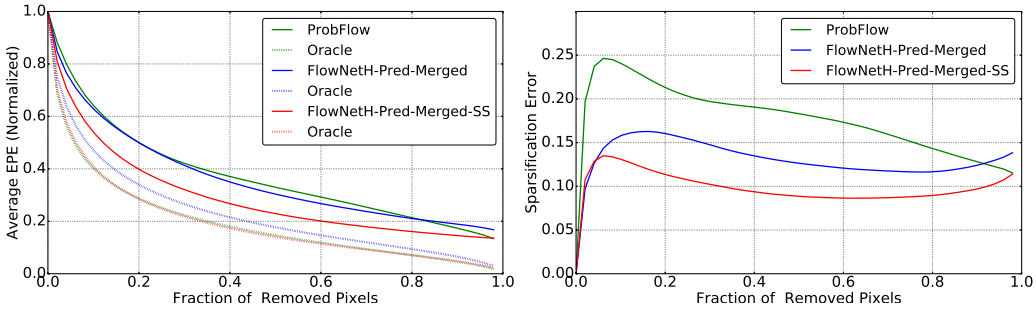


Figure 11.8: **Uncertainty estimation examples from real-world data.** Ours is FlowNetH-Pred-Merged, and "PF" stands for ProbFlow [WKR17]. ProbFlow mostly only predicts uncertainties due to image and flow boundaries, while our approach clearly predicts better uncertainties in general.

SS and train it on FlyingChairs and FlyingThings3D, similar to the stacks from Chapter 9 – with the exception that for FlowNetH and the merging network, we use Batch Normalization [IS15] as well as a continuously dropping cosine learning-rate schedule [LH17] and 1.2M iterations for FlyingChairs. For the full details of the training procedure, we refer to the supplemental material of [IÇG+18].



(a) (b)

Figure 11.9: **Comparing FlowNetH variants to ProbFlow.** Plots of the sparsification curves with their respective oracles **(a)** and of the sparsification errors **(b)** for ProbFlow, FlowNetH-Pred-Merged and FlowNetH-Pred-Merged-SS (version with two refinement networks stacked on top) on the Sintel train final dataset. KITTI versions are similar and are provided in the supplemental material of [IÇG+18].

	Sintel Clean		Sintel Final		KITTI		runtime
	AUSE	EPE	AUSE	EPE	AUSE	EPE	
ProbFlow [WKR17]	0.162	1.87	0.173	3.34	0.466	8.95	38.1s†
FlowNetH-Pred-Merged-FT-KITTI	-	-	-	-	**0.086**	3.12	**60**ms
FlowNetH-Pred-Merged	0.117	2.58	0.128	3.78	0.151	7.84	**60**ms
FlowNetH-Pred-Merged-S	0.091	2.29	0.098	3.51	0.102	6.86	86ms
FlowNetH-Pred-Merged-SS	**0.089**	2.19	**0.096**	3.40	0.091	6.50	99ms

Table 11.3: **Benchmark results for FlowNetH variants.** We compare FlowNetH variants to the state-of-the-art uncertainty estimation method ProbFlow [WKR17] on the Sintel train clean, the Sintel train final and our KITTI 2012+2015 validation split datasets. The "-FT-KITTI" version is fine-tuned on our KITTI 2012+2015 training split. FlowNetH-Pred-Merged, -S and -SS are all trained with the FlowNet2 schedule. Our method outperforms ProbFlow in AUSE by a large margin as well as in terms of EPE for the KITTI dataset. †runtime is taken from [WKR17]. Please see the supplemental material of [IÇG+18] for details on the computation of the ProbFlow outputs.

We compare our networks to the state-of-the-art method ProbFlow [WKR17] on the common Sintel and KITTI benchmarks. Figure 11.9 shows the sparsification plots for the Sintel train final. ProbFlow has almost the same oracle as FlowNetH-Pred-Merged, i.e., the flow field from ProbFlow can equally benefit from sparsification, but the actual sparsification error is higher due to its estimated uncertainty. This shows that FlowNetH-Pred-Merged has superior uncertainty estimates. In Table 11.3, we show that this also holds for the KITTI dataset (for fine-tuning purposes, we combine KITTI 2012 and 2015). FlowNetH also outperforms ProbFlow in terms of EPE in this case. This shows that the superior uncertainty estimates are not due to a weaker optical flow model, i.e., from obvious mistakes that are easy to predict.

Table 11.3 further shows that the uncertainty estimation is not limited to simple encoder-decoder networks but can also be applied successfully to state-of-the-art

stacked networks. The uncertainty estimation is not negatively influenced by the stacking, despite the improving flow fields. This again shows that the uncertainty estimation works reliably, notwithstanding of if the predicted optical flow is good or bad. Finally, note that the three bottom variants of Table 11.3 were trained only on synthetic data and are also able to predict good uncertainties on the totally different domain of the KITTI dataset.

11.6 Summary

This chapter has shown that:

- For FlowNet, SGDR and bootstrapped ensembles perform better than the commonly used dropout technique,
- predictive networks estimate uncertainty due to the input data $\mathbf{x}$, the network architecture f and the training process,
- for FlowNet, it is not necessary to model the epistemic uncertainty over the weights $\mathbf{w}$, (11.6.1)
- the multi-hypotheses FlowNetH network performs among the best methods and yields much faster runtimes than sampling-based approaches and ensembles,
- our FlowNetH-Pred-Merged qualitatively clearly outperforms the state-of-the-art energy based method ProbFlow, and
- FlowNetH variants trained on synthetic data are able to generalize well to real data also in the uncertainty estimations.

Chapter 12

Discussion

Chapter 2 defined the optical flow problem, and Statements 2.1.2 to 2.1.8 lead to the observation that optical flow estimation is a hard problem and ill-defined. In the following, the introduced approach will be related to traditional methods and interesting properties will be discussed.

12.1 Architecture Choice

Feature Refinement Traditional methods [BBPW04, BM11] implement coarse-to-fine estimation to avoid local minima. The drawback is that small objects with large displacements are lost in the coarse resolution and cannot be recovered during the refinement (Statement 3.2.7). FlowNet also employs a coarse-to-fine refinement in the decoder. However, this refinement happens in feature instead of image space, which carries more structure and is also able to transport multiple motion hypotheses per location. It therefore does not suffer from the coarse-to-fine limitations. The estimation of the deep supervision flow fields are only an auxiliary component and do not constrain the main feature stream (see also Figure 6.5).

FlowNet2 Refinement Note that although the FlowNet2 pipeline introduces several refinement steps on warped images, this does not necessarily imply a coarse-to-fine scheme. In general, a refinement network in FlowNet2 can estimate any solution from the supplied images or regularize spatially close or distant solutions given by the earlier network. In fact, we see that the first network in the stack commonly outputs some noise (that might also contain different possible hypotheses) and the second network then selects the correct solution. This is illustrated in Figure 12.1. The behaviour can be considered to be similar to the propagation in PatchMatch [BSFG09] (Section 3.4.3).

Coarse-to-Fine Although the limitations of coarse-to-fine approaches are well-known, the recently published works PWC-Net [SYLK18] and LiteFlowNet [HTL18] have proposed to implement strict coarse-to-fine approaches with FlowNet. While this improves overall performance, it also has the obvious drawback of being unable to estimate large motions of small objects. An experiment to illustrate the limitations of coarse-to-fine approaches is given in Figure 12.2 (see next page).

Figure 12.1: **Results during the refinement pipeline.** The figure uses the Middlebury visualization for best visibility. The refinement happens mainly due to propagating correct solutions.

The traditional coarse-to-fine energy minimization [BBPW04] method in the first result column succeeds only for a very large object size and small displacement. LDOF [BM11] and DeepFlow [WRHS13] integrate descriptor matching and are able to solve up to the smallest object size. All traditional methods (first three result columns) have problems with the weak texture in the background. Results could likely be improved by tuning the trade-off hyper-parameter α of Equation 3.2.3. PWC-Net has problems with fine structures (which also represent small objects) with large displacements and fails for the four bottom rows with smaller object sizes. Note that PWC-Net was also trained on FlyingChairs and should in principle be able to deal with the given data very well. Also note that missing fine structures and missing small objects do not affect the benchmarks significantly because they contribute very little to the overall error. FlowNet2 yields the best results for small objects and retains fine structures throughout the results.

Despite the limitation, PWC-Net and LiteFlowNet offer remarkable benchmark results with much smaller network sizes than FlowNet2 ($\sim$ 9M vs. $\sim$ 160M parameters) and faster runtimes. Arguably, this brings us back to the search for best heuristic to solve the optical flow problem (see also Section 3.4). An important observation is that this search is actually data-driven. If nothing is known about the data's properties, FlowNetS provides the most general approach. FlowNetC increases performance on the given benchmark by making assumptions about possible displacements and object sizes that suit the data well. These assumptions are built into the architecture design with the correlation layer. PWC-Net and LiteFlowNet exploit the data further by integrating the coarse-to-fine approach as well as by being able to estimate flows in large areas quickly and accurately.

(12.1.1) Concluding, there is not a single best architecture for all possible optical flow applications. If nothing is known about the domain, FlowNetS can be applied to any kind of training data and provides the most versatile solution. If, for example, one knows the minimum object sizes and the maximum possible displacement, PWC-Net and LiteFlowNet [HTL18] can be adapted to provide the best and fastest solution. This is illustrated in Figure 12.3 (see next page).

Figure 12.2: **Evaluation of coarse-to-fine estimation abilities.** Each row of the figure shows an example with two images and flow estimations from multiple methods. "Tr." is the traditional coarse-to-fine energy minimization from Brox et al. [BBPW04], "LDOF' is Large Displacement Optical Flow [BM11], "DF" is DeepFlow with DeepMatching [WRHS13], "PWC" is PWC-Net [SYLK18] and "FN2" is FlowNet2. In the second row, we increase the displacement from 30 to 150px and then keep it constant while reducing the object size. Tr. already fails for the larger displacement in the second row. PWC-Net fails from row 5 onwards.

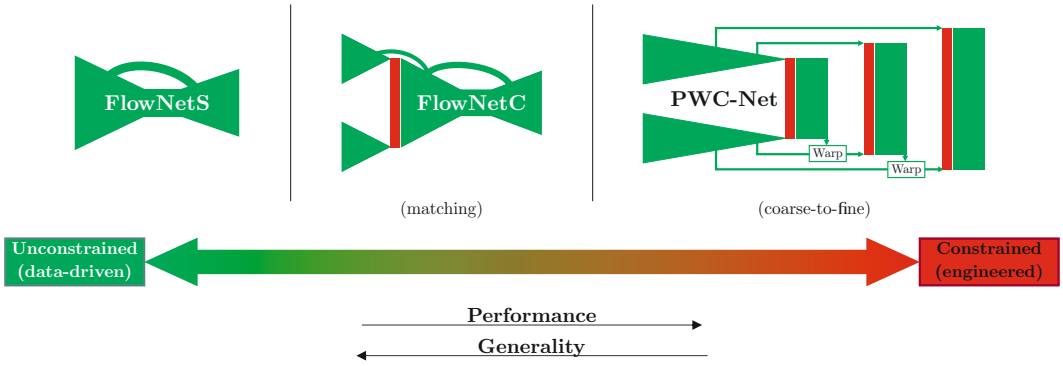


Figure 12.3: **General and specialized architectures.** Red bars indicate correlation layers. Architectures with more engineering perform better on public benchmarks, but sacrifice generality.

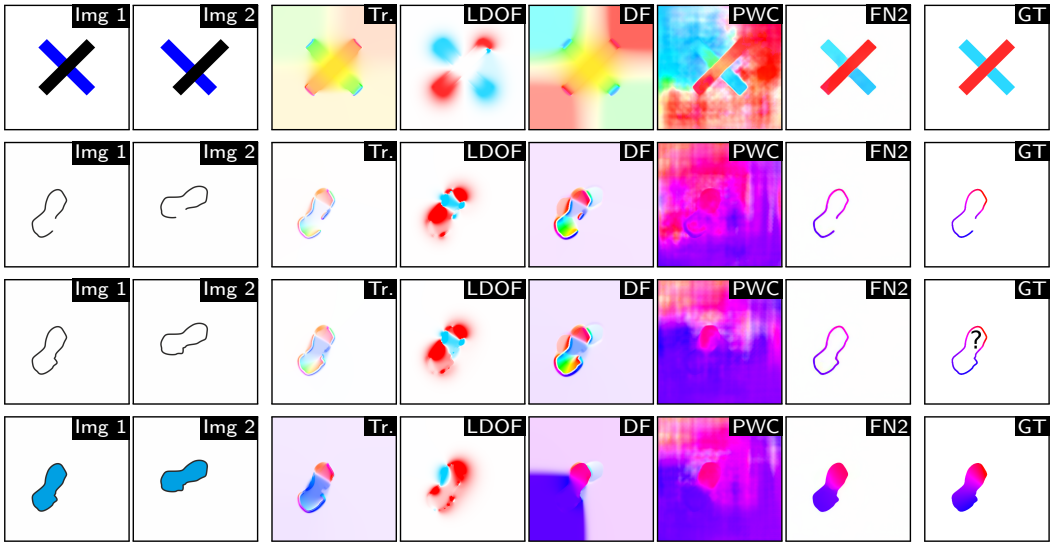


Figure 12.4: **Evaluation of regularization challenges.** Each row shows an example with two images and flow estimations from multiple methods. "Tr." is the traditional coarse-to-fine energy minimization from Brox et al. [BBPW04], "LDOF" is Large Displacement Optical Flow [BM11], "DF" is DeepFlow with DeepMatching [WRHS13], "PWC" is PWC-Net [SYLK18] and "FN2" is FlowNet2. Note that the third case is ambiguous: the inside of the contour could be moving with the contour or with the background. FlowNet2 succeeds to solve all the cases.

12.2 Regularization

Next, we perform a set of experiments in order to test regularization capabilities. The first row of Figure 12.4 shows a common example of the aperture problem. If one considers only a small window around the center, the edges where the objects touch would appear to be moving downwards. Considering the whole image, one can observe that the blue bar is moving to the left and the black bar is moving to the right.

Nonlocal regularization

All approaches succeed in finding the motion at the edges of the bars; however, all traditional approaches fail to determine the motion in the center area. Note that the FlowNet2 training includes the ChairsSDHom dataset, which contains priors for homogeneous backgrounds. The estimation of PWC-Net on the background could likely be improved when also fine-tuning on this dataset. Finding the motion of the bars requires nonlocal regularization. The CNN approaches succeed in this case.

In the second row, an open contour is moving. In contrast to the closed contours, its inside clearly belongs to the background and should not move. However, in traditional approaches, the solution must diffuse through the contour gap, which is problematic. In the third row, the contour is closed, but the inside color is the same as the background. Thus, the inside of the contour could be interpreted as moving with the contour or with the background. FlowNet2 establishes a nonlocal regularization across the contour line and assigns the same motion to the inside as to the background.

In the third example, the inside of the contour is colored differently. Now the inside clearly moves with the contour, as FlowNet2 also correctly determines. In summary:

(12.2.1) From the examples, we observe that CNNs use priors and perform nonlocal regularization, which is much more sophisticated than in traditional methods. In particular, CNNs prove to make much better use of image statistics (as has been shown before in other tasks [KSH12, EPF14, RDGF16]). The results match human perception very closely.

12.3 Comparison of Algorithm Implementations

PatchMatch

As already mentioned in Section 12.1, the solution propagation in the FlowNet2 pipeline appears similar to PatchMatch [BSFG09]. Whether such solution propagation also happens inside a single encoder-decoder architecture is possible but unknown.

LDOF

The correlation layer of FlowNetC can be seen as being similar to the initial nearest-neighbor matching in LDOF [BM11]. However, the FlowNetS without the correlation is also able to solve for large displacements. FlowNetS and FlowNetC each comprise 10 convolutions + ReLUs in the encoder and four upconvolutions + ReLUs in the decoder (when neglecting the deep supervision). LDOF operates on a coarse-to-fine refinement and can hence be seen as being similar to the decoder, although LDOF only refines flow information and not features. The operations of the Gauss-Newton method in LDOF with SOR and lagged diffusivity include additions, multiplications, derivatives, powers of two and nonlinearities. Most of these operations would require multiple convolution and ReLU layers to approximate them. On the examples in this chapter, LDOF performs 88 iterations, where each iteration consists of multiple operations. When comparing to the four layers of the FlowNet decoder, this leads to the conclusion that an algorithm similar to LDOF cannot be implemented in FlowNet.

DeepFlow DeepMatching [WRHS13] operates already on response maps (equivalent to cost volumes and correlations). It could be compared to the regularization part of FlowNetC after the correlation, although the correlation output is much more sparse than the initial response map from DeepMatching. The regularization in DeepMatching is implemented by using sparse convolutions, max-pooling and nonlinearities. Although DeepMatching also implements a backtracking algorithm through the max-pooling operation and FlowNet does not use max-pooling at all, DeepMatching can be seen as an example that regularization can be implemented with CNN operations. Notably, the convolutions in this implementation only use input weights with the values 0 and 1.

The resulting matches from DeepFlow then require a dense interpolation which can be carried out with a variational approach [WRHS13] or with much simpler interpolation [RWHS15]. Since a CNN can make very good use of image statistics, one can expect the dense interpolation to be an easy task for the CNN.

PCA-Flow If sparse matches are available, another way to obtain a dense flow field is by finding a representation of the matches through a learned basis [WB15]. As FlowNet is convolutional, it is not able to store any global basis vectors. However, one could still imagine some local basis representation. In the upconvolutions, the weights can be seen as learned basis vectors $\boldsymbol{b}_i$ and the input features as coefficients α_i (compare to Equation 3.4.2). However, note that the combination of basis vectors leads to a patch for every feature location. These patches overlap, and in order to obtain the result for an output location, the values from the neighboring patches are summed up. Thus, the upconvolution does not exactly correspond to a basis combination.

(12.3.1) In general, we observe that there are some similarities between the presented CNNs in this work and traditional approaches. However, the FlowNets contain very few and simple operations and are able to learn a much more efficient heuristic, producing much better results quantitatively and qualitatively.

This concludes that the traditional methods and past research has not determined the most important principles of optical flow estimation and that much better heuristics do exist.

Chapter 13

Outlook

This chapter presents ideas for future research directions by the author. EWTA and the sampling/fitting framework are from recent joint work [MIÇB19]. While the author contributed to the idea of EWTA, the original concepts of EWTA and the sampling/fitting framework were developed by Osama Makansi.

As listed in Section 1.3, FlowNet and FlowNet2 have already been extended by many other works and have been used in numerous applications. Besides speed and accuracy, reliability and robustness are most important for the use in real-world scenarios. The uncertainty estimates presented in Chapter 11 constitute a milestone towards this goal.

Networks that only provide point estimates have no means of indicating difficult cases. The results of FlowNetH show that networks can be constructed to inform about their own uncertainty and even about failure cases very well. However, in uncertain situations and more complex applications, one might actually be interested in multiple plausible alternative solutions. While FlowNetH already provides such *hypotheses*, combining them with uncertainty prediction would mean to predict a full *mixture distribution*:

$$p(\mathbf{y}|\mathbf{x}) = \sum_{i=1}^{M} \pi_i p(\mathbf{y}|\mu_i, \sigma_i)\,. \tag{13.1}$$

This is illustrated in Figure 13.1. While FlowNetC presented only a single point estimate μ, FlowNetH is able to produce multiple point estimates μ_i or distribution estimates (μ_i, σ_i). What is missing are the relative importances π_i to predict the parameters (μ_i, σ_i, π_i) of a full mixture distribution. The well-known approach to estimate such a mixture density from a single forward pass is Mixture Density Networks [Bis94] which are trained by minimizing the negative log-likelihood of Equation 13.1. However, optimizing for general, unconstrained mixture distributions requires special initialization as well as training procedures and badly suffers from mode collapse [RLD+17, CRC+18, CR18, MFS18, Gra13, HN99].

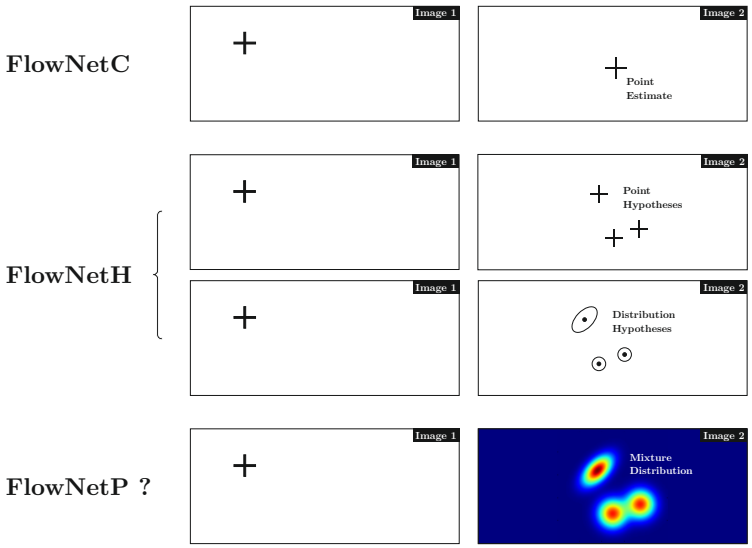


Figure 13.1: **From point estimates to mixture distributions.** While FlowNetC provided only point estimates for correspondences, FlowNetH extended the approach to predicting multiple point or distribution hypotheses. In future, this could be extended to an approach called FlowNetP providing multimodal distributions.

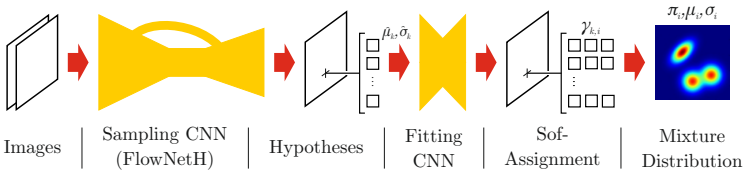


Figure 13.2: **Sampling and fitting framework.** The first network produces a set of point $\hat{\mu}_k$ or distribution samples $(\hat{\mu}_k, \hat{\sigma}_k)$ referred to as *hypotheses*. The second network estimates soft assignments $\gamma_{k,i}$ from which mixture components (π_i, μ_i, σ_i) can be computed. Note that in general, there are many more hypotheses $(\hat{\mu}_k, \hat{\sigma}_k)$, e.g. $k = 1..20$, than mixture components (π_i, μ_i, σ_i), e.g. $i = 1..4$.

Recent joint work with Osama Makansi and Özgün Çiçek [MIÇB19] has shown that the limitations of mixture density networks can be overcome by combining the WTA approach (as also used in FlowNetH) to produce diverse samples with a separate distribution fitting stage. This is illustrated in Figure 13.2. The first network predicts point $\hat{\mu}_k$ or distribution samples $(\hat{\mu}_k, \hat{\sigma}_k)$, which resemble the hypotheses. The second network then fits a mixture distribution to these hypotheses, as is normally done by the EM algorithm [Bis06]. For details of the approach, the reader is referred to the publication [MIÇB19].

Since the work with Osama Makansi and Özgün Çiçek proves that with the sampling and fitting framework from Figure 13.2, high-quality multimodal distributions can be obtained in the task of future prediction, this multimodal prediction could also be implemented for FlowNet, basically extending it to the *FlowNetP* shown in Figure 13.1. This network would predict a mixture distribution over possible correspondences of each pixel in the first image.

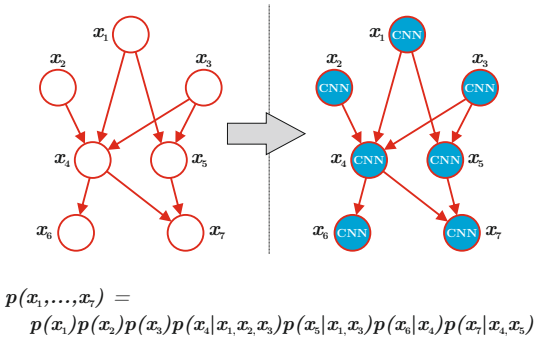


Figure 13.3: **Using CNNs to build graphical models.** In robotics, graphical models are commonly used to engineer robust systems. Providing CNNs that output probability distributions allows for using them as a building block in such probabilistic systems.

This step would advance networks from point estimates to multimodal distributional outputs. In robotics, it has long been common practice to model systems in a probabilistic way [TBF06]. Providing distributions from CNNs would allow for using CNNs as building blocks in such systems. This is illustrated in Figure 13.3. One very well-known method is the *Bayes filter*, as illustrated in Figure 13.4. Here, the robot state is modelled probablistically. While the robot position is completely unknown in the beginning the robot is able to localize itself from multiple *observations*. The important issue to notice is that the robot position cannot be inferred from any single observation alone but only by carrying and refining a distribution over all possible positions across time.

For FlowNet, the concept of belief does not apply since optical flow does not keep track of the states of objects. While for a tracker, implementing the belief over states might be of importance and improve performance, for optical flow, an implementation is difficult due to the dense prediction and the integral over past states of the Bayes filter update step. However, the general concept is still interesting, and one can make use of the insight that multiple observations help narrow down uncertain predictions. Multiple observations could be obtained by:

1. Predicting a distribution over possible motions from a single image. This yields motion priors that really only stem from “typical behaviour”.

2. Predicting a distribution over possible motions from the past trajectory. This yields priors for typical object trajectories.

3. Predicting a distribution over possible motions from multiple sensor inputs. While one sensor might produce ambiguous results in certain situations, more certainty could be obtained from other sensors.

4. Fusing predictions from different models. E.g., next to the two-frame image input, one may maintain a 3D reconstruction of the scene and use it in combination with the future frame to determine a distribution over possible motions. This would couple optical flow estimation with higher level representations.

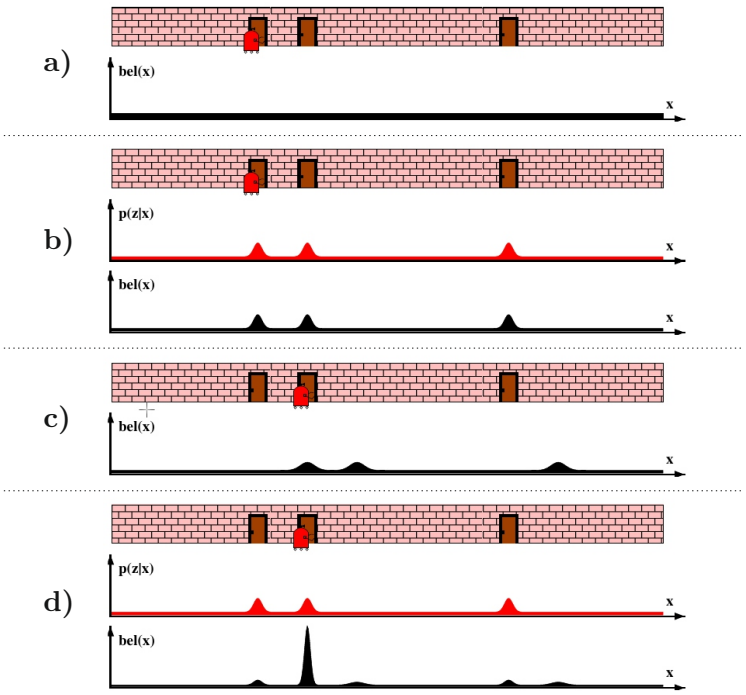


Figure 13.4: **Illustration of Bayes filter.** The figure illustrates a robot using the Bayes filter [TBF06] for localization. The robot state is modeled with a probability distribution called "belief" (bel(x)). In the beginning (a), the belief is uniform, resembling complete uncertainty about the position of the robot. Observation $p(z|x)$ of a door then shapes the belief to the three possible door locations (b). After taking an action, the robot is even more uncertain about its state (c), and observing another door finally leads to a highly certain localization (d). Source: [TBF06].

Section 2.1.4 explained that priors are very important for optical flow estimation, and a major observation in this thesis is that the capability of CNNs to learn such priors well boosts optical flow performance. The first two above items would serve to develop even more sophisticated priors ("learning about how different objects typically move, given a snapshot or a history"), while the second two items would allow for using different sources of information, such as from different sensors or as an inference from a more complex internal state.

This is motivated by the fact that humans also maintain an internal state. As the optical flow estimation is ill-posed (Statement 2.1.6), humans use information, such as the previous motion and semantics about the current scene. Up to today, optical flow algorithms have brought performance to a very high level without using any such knowledge. However, the performance of such algorithms must be limited by the uninformedness and is also likely to reach a limit on the benchmarks at some point. Integrating the above points could approach the complex problem by constructing more informed algorithms, which are more similar to sources of information humans use to reason about scenes, and could be the step for the next generation of algorithms.

Chapter 14

Conclusion

In the past, many algorithms have been proposed for optical flow estimation. An important observation is that the optical flow problem cannot be solved optimally. As described in Section 2.1.4, because of the aperture problem, optical flow estimation is ill-posed and prior knowledge is required to obtain a solution. Furthermore, determining the solution involves complex joint optimization of the flow in combination with occlusions and motion boundaries.

Past methods have provided heuristics, thereby each yielding different performance with respect to the hyper-parameters object size and appearance, displacement size, amount of occlusions and amount of required invariance. A very important insight is that accounting for all possible values of these hyper-parameters is not feasible and that one commonly chooses the best parameters by hand depending on the data, which makes the search for the best algorithm data-driven.

This is where the work of this thesis stroke. An alternative approach that learns the heuristic from training data end-to-end was presented by using convolutional neural networks. The results show that such learned heuristic is significantly better and more efficient than the traditional engineered methods. In particular, the work has shown...

- For the first time, that end-to-end estimation of optical flow with CNNs is possible.
- That such CNNs learn the concept of correspondence and generalize very well from simplistic synthetic data.
- That the CNNs learn priors that correspond to human perception very closely and outperform engineered approaches in this respect by far.
- That such approaches run in real time.
- That it is possible to obtain state-of-the-art results. The approach was extended to a refinement pipeline with a stack of networks, thus achieving state of the

art in KITTI 2012 and 2015 optical flow as well as in KITTI 2015 disparity benchmarks.

- That the approach can be extended to full scene flow.
- That the results of network stacks deliver smooth flow fields as well as crisp motion boundaries and are able to estimate fine details while not suffering from coarse-to-fine limitations.
- That in comparison to traditional methods, CNNs are far superior in occlusion and motion boundary estimation. For both, state of the art was achieved and led to a new state of the art in motion segmentation.
- That infinite amounts of unlabeled real data can be successfully integrated into the training procedure.
- That the networks are even able to inform about their own predictions' uncertainty very well and outperform engineered methods in this respect by far, too. State of the art was achieved in uncertainty estimation for optical flow.

Particularly the abilities of the approach to be fine-tuned to the target domain, to run in real time and to provide uncertainty estimates have made it the method of choice for applications. As a whole, these contributions have revolutionized optical flow estimation and changed the direction of research in the field. A promising direction of further research is extending the approach to full mixture distributions and using it to create larger and more complex AI systems.

Acknowledgements

First, I would like to express my sincere thanks to my advisor Professor Thomas Brox for the continuous support of my PhD study and related research, for his motivation, good advice and especially his patience.

Besides my advisor, the extent of this work would never have been possible on my own, so I would like to thank all coauthors of the publications. In particular, I would like to thank Alexey Dosovitskiy for being a mentor, Philipp Fischer for initiating the first Hackathon, Nikolaus Mayer for the great work on training data, Margret Keuper for the discussions as well as the work on occlusions and motion segmentation, Özgün Çiçek for the joint work on uncertainty and finally Osama Makansi and Silvio Galesso for being enthusiastic and prospering students. I have sincerely enjoyed working together with all of you and will miss you.

As in every research career, I also had frustrating times and times of doubt. In this respect, I'd like to thank Nikolaus Hensler who encouraged me to live with temporal uncertainty and nevertheless to continue. Only this made it possible to bring this PhD project to such great success. In this respect, I would also like to thank Ricardo Bartra and my mother for believing in me and giving me emotional support.

At last, I'd like to thank the DFG and Trimbot 2020 for the funding and making this research possible in the first place.

Bibliography

[ADPS07] Luis Alvarez, Rachid Deriche, Théo Papadopoulo, and Javier Sánchez. Symmetrical dense optical flow estimation with occlusions detection. *Int. Journal of Computer Vision (IJCV)*, 75(3):371–385, Dec 2007.

[AHPB13] Osin Mac Aodha, Ahmad Humayun, Marc Pollefeys, and Gabriel J. Brostow. Learning a confidence measure for optical flow. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(5):1107–1120, May 2013.

[AME+14] Mathieu Aubry, Daniel Maturana, Alexei Efros, Bryan Russell, and Josef Sivic. Seeing 3D chairs: Exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[AMFM11] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(5), May 2011.

[AP16] Aria Ahmadi and Ioannis Patras. Unsupervised convolutional neural networks for motion estimation. In *Int. Conference on Image Processing (ICIP)*, 2016.

[BA96] Michael J. Black and Padmanabhan Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding (CVIU)*, 63(1):75–104, 1996.

[BBM09] Thomas Brox, Christoph Bregler, and Jitendra Malik. Large displacement optical flow. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[BBPW04] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision (ECCV)*, 2004.

[BCKW15] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *Int. Conference on Machine Learning (ICML)*, 2015.

[BETVG08] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, Jun 2008.

[BF00a] Michael J. Black and David J. Fleet. Probabilistic detection and tracking of motion boundaries. *Int. Journal of Computer Vision (IJCV)*, 38(3):231–245, Jul 2000.

[BF00b] Michael J. Black and David J. Fleet. Probabilistic detection and tracking of motion boundaries. *Int. Journal of Computer Vision (IJCV)*, 38(3):231–245, Jul 2000.

[BFB94] John L. Barron, David J. Fleet, and Steven S. Beauchemin. Performance of optical flow techniques. *Int. Journal of Computer Vision (IJCV)*, 12(1):43–77, Feb 1994.

[BG05] Michael Bleyer and Margrit Gelautz. A layered stereo matching algorithm using image segmentation and global visibility constraints. *Journal of Photogrammetry and Remote Sensing (ISPRS)*, 59:128–150, 05 2005.

[Bis94] Christopher M. Bishop. Mixture density networks. Technical report, Aston University, Birmingham, UK, 1994.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2nd edition, 2006.

[BM10] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *European Conference on Computer Vision (ECCV)*, 2010.

[BM11] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(3):500–513, 2011.

[BR78] Andrew Burton and John Radford. *Thinking in Perspective: Critical Essays in the Study of Thought Processes*. Methuen, 1978.

[Bro05] Thomas Brox. *From pixels to regions: Partial differential equations in image analysis*. PhD thesis, Faculty of Mathematics and Computer Science, Saarland University, April 2005.

[Bro18] Thomas Brox. Computer Vison class 8, Optical Flow II. `https://lmb.informatik.uni-freiburg.de/lectures/computer_vision_I/slides/ComputerVision08.pdf`, 2018.

[BSFG09] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (TOG)*, 28(3), Aug 2009.

[BSL+09] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. Technical Report MSR-TR-2009-179, December 2009.

[BTS15] Christian Bailer, Bertram Taetz, and Didier Stricker. Flow Fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. *Int. Conference on Computer Vision (ICCV)*, 2015.

[BVS17] Christian Bailer, Kiran Varanasi, and Didier Stricker. CNN-based patch matching for optical flow with thresholded hinge embedding loss. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[BW06] Andrés. Bruhn and Joachim Weickert. A confidence measure for variational optic flow methods. In *Geometric Properties for Incomplete Data*, pages 283–298, 2006.

[BWL18] Linchao Bao, Baoyuan Wu, and Wei Liu. CNN in MRF: Video object segmentation via inference in a CNN-based higher-order spatio-temporal MRF. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[BWSB12] Daniel J. Butler, Jonas Wulff, Garett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision (ECCV)*, 2012.

[CAA17] Aaron Chadha, Alhabib Abbas, and Yiannis Andreopoulos. Compressed-domain video classification with deep neural networks: There's way too much information to decode the matrix. In *Int. Conference on Image Processing (ICIP)*, 2017.

[CAA19] Aaron Chadha, Alhabib Abbas, and Yiannis Andreopoulos. Video classification with CNNs: Using the codec as a spatio-temporal activity sensor. *Transactions on Circuits and Systems for Video Technology (TCSVT)*, 29(2):475–485, Oct 2019.

[CCL+18] Yu-Ting Chen, Wen-Yen Chang, Hai-Lun Lu, Tingfan Wu, and Min Sun. Leveraging motion priors in videos for improving human segmentation. In *European Conference on Computer Vision (ECCV)*, 2018.

[CFG14] Tianqi Chen, Emily B. Fox, and Carlos Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *Int. Conference on Machine Learning (ICML)*, 2014.

[CHM+15] Anna Choromanska, Mikael Henaff, Michaël Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. *Journal of Machine Learning Research (JMLR)*, 38:192–204, 2015.

[CK14] Qifeng Chen and Vladlen Koltun. Fast MRF optimization with application to depth reconstruction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[CK17] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *Int. Conference on Computer Vision (ICCV)*, 2017.

[COR+16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[CP17] Yunjin Chen and Thomas Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(6):1256–1272, Jun 2017.

[CR18] Joseph Curro and John Raquet. Deriving confidence from artificial neural networks for navigation. In *Position, Location and Navigation Symposium (PLANS)*, 2018.

[CRC+18] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. *arXiv pre-print*, 1809.10732, 2018.

[Cyb89] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems (MCSS)*, 2(4):303–314, 1989.

[DB15] Benjamin Drayer and Thomas Brox. Combinatorial regularization of descriptor matching for optical flow estimation. In *British Machine Vision Conference (BMVC)*, 2015.

[DFI+15] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *Int. Conference on Computer Vision (ICCV)*, 2015.

[DFS+16] Alexey Dosovitskiy, Philipp Fischer, Jost T. Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 38(9):1734–1747, Oct 2016.

[DGI16] Ürün Doğan, Tobias Glasmachers, and Christian Igel. A unified view on multi-class support vector classification. *Journal of Machine Learning Research (JMLR)*, 17(1):1550–1831, Jan 2016.

[DLHT16] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 38(02):295–307, Feb 2016.

[DSB15] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[DYLT07] Yi Deng, Qiong Yang, Xueyin Lin, and Xiaoou Tang. Stereo correspondence with occlusion handling in a symmetric patch-based graph-cuts model. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(6):1068–1079, Jun 2007.

[DZ13] Piotr Dollár and C. Lawrence Zitnick. Structured forests for fast edge detection. In *Int. Conference on Computer Vision (ICCV)*, 2013.

[EMW04] Geoffrey Egnal, Max Mintz, and Richard P. Wildes. A stereo confidence metric using single view imagery with comparison to five alternative approaches. *Image and Vision Computing*, 22(12):943–957, 2004.

[ENT18] Alaaeldin El-Nouby and Graham W. Taylor. Real-time end-to-end action detection with two-stream networks. In *Conference on Computer and Robot Vision (CRV)*, 2018.

[EPF14] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Int. Conference on Neural Information Processing Systems (NIPS)*, 2014.

[FAFM15] Katerina Fragkiadaki, Pablo Arbeláez, Panna Felsen, and Jitendra Malik. Learning to segment moving objects in videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[FAR18] Kevin J. Shih Robert Kirby Jon Barker David Tarjan Andrew Tao Bryan Catanzaro Fitsum A. Reda, Guilin Liu. SDC-Net: Video prediction using spatially-displaced convolution. In *European Conference on Computer Vision (ECCV)*, 2018.

[FBK15] Denis Fortun, Patrick Bouthemy, and Charles Kervrann. Optical flow modeling and computation: A survey. *Computer Vision and Image Understanding (CVIU)*, 134:1–21, May 2015.

[FDB14] Philipp Fischer, Alexey Dosovitskiy, and Thomas Brox. Descriptor matching with convolutional neural networks: A comparison to SIFT. *arXiv pre-print*, 1405.5769, 2014.

[FVT+93] Olivier Faugeras, Thierry Viéville, Eric Theron, Jean Vuillemin, Bernard Hotz, Zhengyou Zhang, Laurent Moll, Patrice Bertin, Herve Mathieu, Pascal Fua, Gérard Berry, and Catherine Proy. Real-time correlation-based stereo: algorithm, implementations and applications. Research Report RR-2013, INRIA, 1993.

[GAB17] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[GG15] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with Bernoulli approximate variational inference. In *Int. Conference on Learning Representations (ICLR) Workshop*, 2015.

[GG16a] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Int. Conference on Machine Learning (ICML)*, 2016.

[GG16b] Fatma Güney and Andreas Geiger. Deep discrete flow. In *Asian Conference on Computer Vision (ACCV)*, 2016.

[GGZY18] Chang Gao, Derun Gu, Fangjun Zhang, and Yizhou Yu. ReCoNet: Real-time coherent video style transfer network. In *Asian Conference on Computer Vision (ACCV)*, 2018.

[GK17] Spyros Gidaris and Nikos Komodakis. Detect, replace, refine: Deep structured prediction for pixel wise labeling. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[GLY95] Davi Geiger, Bruce Ladendorf, and Alan Yuille. Occlusions and binocular stereo. *Int. Journal of Computer Vision (IJCV)*, 14(3):211–226, Apr 1995.

[Gra11] Alex Graves. Practical variational inference for neural networks. In *Int. Conference on Neural Information Processing Systems (NIPS)*, 2011.

[Gra13] Alex Graves. Generating sequences with recurrent neural networks. *arXiv pre-print*, 1308.0850, 2013.

[GRBK12] Abner Guzman-Rivera, Dhruv Batra, and Pushmeet Kohli. Multiple choice learning: Learning to produce multiple structured outputs. In *Int. Conference on Neural Information Processing Systems (NIPS)*, 2012.

[HA15a] J. Hernández-Lobato and R. Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *Int. Conference on Machine Learning (ICML)*, 2015.

[HA15b] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *Int. Conference on Learning Representations (ICLR) Workshop*, 2015.

[HAB11] Ahmad Humayun, Oisin Mac Aodha, and Gabrial J. Brostow. Learning to find occlusion regions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[HBK+14] Michael Hornacek, Frederic Besse, Jan Kautz, Andrew W. Fitzgibbon, and Carsten Rother. Highly overparameterized optical flow using PatchMatch belief propagation. In *European Conference on Computer Vision (ECCV)*, 2014.

[HC16] Sojeong Ha and Seungjin Choi. Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors. In *Int. Joint Conference on Neural Networks (IJCNN)*, 2016.

[HD07] Frédéric Huguet and Frédéric Devernay. A variational method for scene flow estimation from stereo sequences. In *Int. Conference on Computer Vision (ICCV)*, 2007.

[HDW13] David Hafner, Oliver Demetz, and Joachim Weickert. Why is the Census Transform good for robust optic flow computation? In *Scale Space and Variational Methods in Computer Vision (SSVM)*, 2013.

[HE08] James Hays and Alexei A. Efros. im2gps: estimating geographic information from a single image. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[HHC+18] Po-Yu Huang, Wan-Ting Hsu, Chun-Yueh Chiu, Ting-Fan Wu, and Min Sun. Efficient uncertainty estimation for semantic segmentation in videos. In *European Conference on Computer Vision (ECCV)*, 2018.

[Hir05] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[HLP17] Gao Huang, Yixuan Li, and Geoff Pleiss. Snapshot ensembles: Train 1, get M for free. In *Int. Conference on Learning Representations (ICLR)*, 2017.

[HLvdMW17] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[HM12] Xiaoyan Hu and Philippos Mordohai. A quantitative evaluation of confidence measures for stereo vision. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(11):2121–2133, Nov 2012.

[HMLL18] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. Learning to steer by mimicking features from heterogeneous auxiliary networks. *arXiv pre-print*, 1811.02759, 2018.

[HN99] Lars U. Hjorth and Ian T. Nabney. Regularization of mixture density networks. In *IEE Conference Publication*, 1999.

[Hor91] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, Mar 1991.

[HR17] Junhwa Hur and Stefan Roth. MirrorFlow: Exploiting symmetries in joint optical flow and occlusion estimation. In *Int. Conference on Computer Vision (ICCV)*, 2017.

[HS81] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[HTL18] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. LiteFlowNet: A lightweight convolutional neural network for optical flow estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[HWK+18] Ping Hu, Gang Wang, Xiangfei Kong, Jason Kuen, and Yap-Peng Tan. Motion-guided cascaded refinement network for video object segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[IÇG+18] Eddy Ilg, Özgün Çiçek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In *European Conference on Computer Vision (ECCV)*, 2018.

[IMS+17] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Int. Conference on Machine Learning (ICML)*, 2015.

[ISB18] Eddy Ilg, Tonmoy Saikia, and Thomas Brox. Occlusions, motion and depth boundaries with a generic network for optical flow, disparity, or scene flow estimation. In *European Conference on Computer Vision (ECCV)*, 2018.

[JGW+17] Joel Janai, Fatma Güney, Jonas Wulff, Michael Black, and Andreas Geiger. Slow Flow: Exploiting high-speed cameras for accurate and diverse optical flow reference data. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[KAB15] Margret Keuper, Brun Andrés, and Thomas Brox. Motion trajectory segmentation via minimum cost multicuts. In *Int. Conference on Computer Vision (ICCV)*, 2015.

[KB15] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Int. Conference on Learning Representations (ICLR)*, 2015.

[KBI+19] Anna Khoreva, Rodrigo Benenson, Eddy Ilg, Thomas Brox, and Bernt Schiele. Lucid data dreaming for multiple object tracking. *Int. Journal of Computer Vision (IJCV)*, Mar 2019.

[KDD18] Dimitrios Konstantinidis, Kosmas Dimitropoulos, and Petros Daras. A deep learning approach for analyzing video and skeletal features in sign language recognition. In *Int. Conference on Imaging Systems and Techniques (IST)*, pages 1–6, 2018.

[Keu17] Margret Keuper. Higher-order minimum cost lifted multicuts for motion segmentation. In *Int. Conference on Computer Vision (ICCV)*, 2017.

[KG17] Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *Int. Conference on Neural Information Processing Systems (NIPS)*, 2017.

[KK12] Philipp Krähenbühl and Vladlen Koltun. Efficient nonlocal regularization for optical flow. In *European Conference on Computer Vision (ECCV)*, 2012.

[KKJG07] Claudia Kondermann, Daniel Kondermann, Bernd Jähne, and Christoph Garbe. An adaptive confidence measure for optical flows based on linear subspace projections. In *German Conference on Pattern Recognition (GCPR)*, 2007.

[KMD+17] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *Int. Conference on Computer Vision (ICCV)*, 2017.

[KMF18] Moritz Kampelmühler, Michael G Müller, and Christoph Feichtenhofer. Camera-based vehicle velocity estimation from monocular video. *arXiv pre-print*, 1802.07094, 2018.

[KMG08] Claudia Kondermann, Rudolf Mester, and Christoph Garbe. A statistical confidence measure for optical flows. In *European Conference on Computer Vision (ECCV)*, 2008.

[KMT14] Vladimir Kolmogorov, Pascal Monasse, and Pauline Tan. Kolmogorov and Zabih's graph cuts stereo matching algorithm. *Image Processing On Line*, 4:220–251, Oct 2014.

[KN11] Jan Kybic and Claudia Nieuwenhuis. Bootstrap optical flow confidence and uncertainty measure. *Computer Vision and Image Understanding (CVIU)*, 115(10):1449–1462, 2011.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Int. Conference on Neural Information Processing Systems (NIPS)*, 2012.

[KSK06] Andreas Klaus, Mario Sormann, and Konrad Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *Int. Conference on Pattern Recognition (ICPR)*, 2006.

[KZ01a] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions using graph cuts. In *Int. Conference on Computer Vision (ICCV)*, 2001.

[KZ01b] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions using graph cuts. In *Int. Conference on Computer Vision (ICCV)*, 2001.

[LAC08] Sebastien Lefebvre, Sebastien Ambellouis, and Francois Cabestaing. A colour correlation-based stereo matching using 1D windows. In *Conference on Signal-Image Technologies and Internet-Based Systems (SITIS)*, 2008.

[LB98] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The Handbook of Brain Theory and Neural Networks*, pages 255–258, 1998.

[LC18] Xin Li and Mooi Choo Chuah. ReHAR: Robust and efficient human activity recognition. In *Winter Conference on Applications of Computer Vision (WACV)*, 2018.

[LFG+18] Zhengfa Liang, Yiliu Feng, Yulan Guo, Hengzhu Liu, Linbo Qiao, Wei Chen, Li Zhou, and Jianfeng Zhang. Learning for disparity estimation through feature constancy. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[LH17] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *Int. Conference on Learning Representations (ICLR)*, 2017.

[LHY17a] Wei-Sheng Lai, Jia-Bin Huang, and Ming-Hsuan Yang. Semi-supervised learning for optical flow with generative adversarial networks. In *Int. Conference on Neural Information Processing Systems (NIPS)*. 2017.

[LHY17b] Wei-Sheng Lai, Jia-Bin Huang, and Ming-Hsuan Yang. Semi-supervised learning for optical flow with generative adversarial networks. In *Int. Conference on Neural Information Processing Systems (NIPS)*, 2017.

[LK81] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Int. Joint Conference on Artificial Intelligence (IJCAI)*, 1981.

[LL18a] Xiaoxiao Li and Chen Change Loy. Video object segmentation with joint re-identification and attention-aware mask propagation. In *European Conference on Computer Vision (ECCV)*, 2018.

[LL18b] Xiaoxiao Li and Chen Change Loy. Video object segmentation with joint re-identification and attention-aware mask propagation. In *European Conference on Computer Vision (ECCV)*, 2018.

[LLR18] Yin Li, Miao Liu, and James M. Rehg. In the eye of beholder: Joint learning of gaze and actions in first person video. In *European Conference on Computer Vision (ECCV)*, 2018.

[LNSC14] Song-Lin Liu, Zhao-Dong Niu, Gang Sun, and Zeng-Ping Chen. Gabor filter-based edge detection: A note. *Optics*, 125(15):4120–4123, 2014.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision (IJCV)*, 60(2):91–110, Nov 2004.

[LPB16] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Int. Conference on Neural Information Processing Systems (NIPS) Workshop*, 2016.

[LPC+16] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. Stochastic multiple choice learning for training diverse deep ensembles. In *Int. Conference on Neural Information Processing Systems (NIPS)*, 2016.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[LSS12a] Marius Leordeanu, Rahul Sukthankar, and Cristian Sminchisescu. Efficient closed-form solution to generalized boundary detection. In *European Conference on Computer Vision (ECCV)*, 2012.

[LSS12b] Marius Leordeanu, Rahul Sukthankar, and Cristian Sminchisescu. Efficient closed-form solution to generalized boundary detection. In *European Conference on Computer Vision (ECCV)*, 2012.

[LSV+18] Siyang Li, Bryan Seybold, Alexey Vorobyov, Xuejing Lei, and C. C. Jay Kuo. Unsupervised video object segmentation with motion-based bilateral networks. In *European Conference on Computer Vision (ECCV)*, 2018.

[LTC17] Marc T. Law, Nicolas Thome, and Matthieu Cord. Learning a distance metric from relative comparisons between quadruplets of images. *Int. Journal of Computer Vision (IJCV)*, 121(1):65–94, Jan 2017.

[LYT11] Ce Liu, Jenny Yuen, and Antonio Torralba. SIFT Flow: Dense correspondence across scenes and its applications. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(5):978–994, May 2011.

[LZS13] Marius Leordeanu, Andrei Zanfir, and Cristian Sminchisescu. Locally affine sparse-to-dense matching for motion and occlusion estimation. In *Int. Conference on Computer Vision (ICCV)*, 2013.

[LZXW18] Qiuyu Li, Shu Zhan, Liangfeng Xu, and Congzhong Wu. Facial micro-expression recognition based on the fusion of deep learning and enhanced optical flow. *Multimedia Tools and Applications*, pages 1–16, 2018.

[Mac92] David J. C. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, May 1992.

[Mat92] Larry Matthies. Stereo vision for planetary rovers: Stochastic modeling to near real-time implementation. *Int. Journal of Computer Vision (IJCV)*, 8(1):71–91, Jul 1992.

[MAW+07] Paul Merrell, Amir Akbarzadeh, Liang Wang, Philippos Mordohai, Jan-Michael Frahm, Ruigang Yang, David Nistér, and Marc Pollefeys. Real-time visibility-based fusion of depth maps. In *Int. Conference on Computer Vision (ICCV)*, 2007.

[MD11] Robert C. Moore and John DeNero. L1 and L2 regularization for multiclass hinge loss models. In *Symposium on Machine Learning in Speech and Natural Language Processing*, 2011.

[MDSL17] Muhammad Habib Mahmood, Yago Díez, Joaquim Salvi, and Xavier Lladó. A collection of challenging motion segmentation benchmark datasets. *Pattern Recognition*, 61:1–14, 2017.

[MFS18] Safa Messaoud, David Forsyth, and Alexander G. Schwing. Structural consistency and controllability for diverse colorization. In *European Conference on Computer Vision (ECCV)*, 2018.

[MG15] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[MHN13] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Int. Conference on Machine Learning (ICML) Workshop*, 2013.

[MHR18] Simon Meister, Junhwa Hur, and Stefan Roth. UnFlow: Unsupervised learning of optical flow with a bidirectional census loss. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2018.

[MIB17] Osama Makansi, Eddy Ilg, and Thomas Brox. End-to-end learning of video super-resolution with motion compensation. In *German Conference on Pattern Recognition (GCPR)*, 2017.

[MIB18] Osama Makansi, Eddy Ilg, and Thomas Brox. FusionNet and AugmentedFlowNet: Selective proxy ground truth for training on unlabeled image. *arXiv pre-print*, 1808.06389, 2018.

[MIÇB19] Osama Makansi, Eddy Ilg, Özgün Çiçek, and Thomas Brox. Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[MIF+18] Nikolaus Mayer, Eddy Ilg, Philipp Fischer, Caner Hazirbas, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. What makes good synthetic training data for learning disparity and optical flow estimation? *Int. Journal of Computer Vision (IJCV)*, 126(9):942–960, Sep 2018.

[MIH+16] Nikolaus Mayer, Eddy Ilg, Philip Häusser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[MP98] Étienne Mémin and Patrick Pérez. A multigrid approach for hierarchical motion estimation. In *Int. Conference on Computer Vision (ICCV)*, 1998.

[MS05] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27(10):1615–1630, Oct 2005.

[MT99] Roberto Manduchi and Carlo Tomasi. Distinctiveness maps for image matching. In *Int. Conference on Image Analysis and Processing (ICIAP)*, 1999.

[Nea96] Radford M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1996.

[NF16] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision (ECCV)*, 2016.

[NH10] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *Int. Conference on Machine Learning (ICML)*, 2010.

[NS18] David Nilsson and Cristian Sminchisescu. Semantic video segmentation by gated recurrent flow propagation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[NŠM18] Michal Neoral, Jan Šochman, and Jiří Matas. Continual occlusions and optical flow estimation. In *Asian Conference on Computer Vision (ACCV)*, 2018.

[NW94] David A. Nix and Andreas S. Weigend. Estimating the mean and variance of the target probability distribution. In *World Congress on Computational Intelligence*, 1994.

[NYD16] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision (ECCV)*, 2016.

[OMB14] Peter Ochs, Jitendra Malik, and Thomas Brox. Segmentation of moving objects by long term video analysis. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(6):1187–1200, Jun 2014.

[ON06] Michael Otte and Hans-Hellmut Nagel. Optical flow estimation: Advances and comparisons. In *European Conference on Computer Vision (ECCV)*, 2006.

[PPSHS18] Eduardo Pérez-Pellitero, Mehdi SM Sajjadi, Michael Hirsch, and Bernhard Schölkopf. Photorealistic video super resolution. In *European Conference on Computer Vision (ECCV) Workshop*, 2018.

[PRCBP16] Juan-Manuel Perez-Rua, Tomas Crivelli, Patrick Bouthemy, and Patrick Perez. Determining occlusions from space and time image reconstructions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[Pro17] Blender Project. Open movies (Agent, Caminandes, Cosmos, Sintel and Big Buck Bunny). `https://www.blender.org/about/projects`, 2017.

[PSA10] Mathias Perrollaz, Anne Spalanzani, and Didier Aubert. Probabilistic representation of the uncertainty of stereo-vision and application to obstacle detection. In *Intelligent Vehicles Symposium (IV)*, 2010.

[PSR+17] Jiahao Pang, Wenxiu Sun, Jimmy S. J. Ren, Chengxi Yang, and Qiong Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *Int. Conference on Computer Vision (ICCV) Workshop*, 2017.

[PTM17] Matteo Poggi, Fabio Tosi, and Stefano Mattoccia. Quantitative evaluation of confidence measures in a machine learning world. In *Int. Conference on Computer Vision (ICCV)*, 2017.

[RB17] Anurag Ranjan and Michael J. Black. Optical flow estimation using a spatial pyramid network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[RDB18] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos and spherical images. *Int. Journal of Computer Vision (IJCV)*, 126(11):1199–1219, Nov 2018.

[RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[RKLS07] Carsten Rother, Vladimir Kolmogorov, Victor Lempitsky, and Martin Szummer. Optimizing binary MRFs via extended roof duality. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[RLD+17] Christian Rupprecht, Iro Laina, Robert DiPietro, Maximilian Baust, Federico Tombari, Nassir Navab, and Gregory D. Hager. Learning in an uncertain world: Representing ambiguity through multiple hypotheses. In *Int. Conference on Computer Vision (ICCV)*, 2017.

[RWHS15] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. EpicFlow: Edge-preserving interpolation of correspondences for optical flow. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[RYN+17] Zhe Ren, Junchi Yan, Bingbing Ni, Bin Liu, Xiaokang Yang, and Hongyuan Zha. Unsupervised deep learning for optical flow estimation. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2017.

[SAMR18] Hajar Sadeghi Sokeh, Vasileios Argyriou, Dorothy Monekosso, and Paolo Remagnino. Superframes, a temporal video segmentation. In *Int. Conference on Pattern Recognition (ICPR)*, 2018.

[SBM+11] Patrick Sundberg, Thomas Brox, Michael Maire, Pablo Arbelaez, and Jitendra Malik. Occlusion boundary detection and figure/ground assignment from optical flow. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[SCH15] Manolis Savva, Angel X. Chang, and Pat Hanrahan. Semantically-enriched 3D models for common-sense knowledge. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[SEG+18] Mennatullah Siam, Sara Eikerdawy, Mostafa Gamal, Moemen Abdel-Razek, Martin Jagersand, and Hong Zhang. Real-time segmentation with appearance, motion and geometry. In *Int. Conference on Intelligent Robots and Systems (IROS)*, 2018.

[SHLC09] Sumit Srivastava, Seong Jong Ha, Sang Hwa Lee, and Nam Ik Cho. Stereo matching using hierarchical belief propagation along ambiguity gradient. In *Int. Conference on Image Processing (ICIP)*, 2009.

[SJ04] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. In *Int. Conference on Neural Information Processing Systems (NIPS)*, 2004.

[SKH+19] A. F. M. Saifuddin Saif, Akib Shahriar Khan, Abir Mohammad Hadi, Rahul Prashad Karmoker, and Joy Julian Gomes. Aggressive action estimation: A comprehensive review on neural network based human segmentation and action recognition. *Int. Journal of Education and Management Engineering (IJEME)*, 9:9–19, Jan 2019.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[SLD17] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(4):640–651, Apr 2017.

[SLP14] Deqing Sun, Ce Liu, and Hanspeter Pfister. Local layering for joint motion estimation and occlusion detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[SN19] Pat Hanrahan Shree Nayar, Ravi Ramamoorthi. Basic Principles of Surface Reflectance. `hhttps://www.cs.cmu.edu/afs/cs/academic/class/15462-f09/www/lec/lec8.pdf`, 2019. Accessed 25-Jan-2019.

[SP16] Akihito Seki and Marc Pollefeys. Patch based confidence prediction for dense disparity map. In *British Machine Vision Conference (BMVC)*, 2016.

[SS98] Daniel Scharstein and Richard Szeliski. Stereo matching with nonlinear diffusion. *Int. Journal of Computer Vision (IJCV)*, 28(2):155–174, Jun 1998.

[SSB12] Deqing Sun, Erik Sudderth, and Michael J. Black. Layered segmentation and optical flow estimation over time. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[STB+18] Nicola Strisciuglio, Radim Tylecek, Michael Blaich, Nicolai Petkov, Peter Biber, Jochen Hemming, Eldert van Henten, Torsten Sattler, Marc Pollefeys, Theo Gevers, Thomas Brox, and Robert B. Fisher. TrimBot2020: An outdoor robot for automatic gardening. In *50th Int. Symposium on Robotics (ISR)*, 2018.

[SW17] Amit Shaked and Lior Wolf. Improved stereo matching with constant highway networks and reflective confidence learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[SWS+13] Deqing Sun, Jonas Wulff, Erik Sudderth, Hanspeter Pfister, and Michael Black. A fully-connected layered model of foreground and background flow. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[SYLK18] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[SZ14a] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Int. Conference on Neural Information Processing Systems (NIPS)*, 2014.

[SZ14b] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Int. Conference on Learning Representations (ICLR)*, 2014.

[SZB17] Nima Sedaghat, Mohammadreza Zolfaghari, and Thomas Brox. Hybrid learning of optical flow and next frame prediction to boost optical flow in the wild. *arXiv pre-print*, 1612.03777, 2017.

[Sze10] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag, 1st edition, 2010.

[SZS13] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv pre-print*, 1212.0402, 2013.

[TBF06] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. Cambridge University Press, 1st edition, 2006.

[TBL18] Adam M Terwilliger, Garrick Brazil, and Xiaoming Liu. Recurrent flow-guided semantic forecasting. In *Winter Conference on Applications of Computer Vision (WACV)*, 2018.

[TCM17] Pauline Tan, Antonin Chambolle, and Pascal Monasse. Occlusion detection in dense stereo estimation with convex optimization. In *Int. Conference on Image Processing (ICIP)*, 2017.

[TDDT+18] Minh-Triet Tran, Tung Dinh-Duy, Thanh-Dat Truong, Vinh Ton-That, Thanh-Nhon Do, Quoc-An Luong, Thanh-An Nguyen, Vinh-Tiep Nguyen, and Minh N. Do. Traffic flow analysis with multiple adaptive vehicle detectors and velocity estimation with landmark-based scanlines. In *Conference on Computer Vision and Pattern Recognition (CVPR) Workshop*, 2018.

[Tka18] Dmytro Tkachenko. Human action recognition using fusion of modern deep convolutional and recurrent neural networks. In *Int. Conference on System Analysis & Intelligent Computing (SAIC)*, 2018.

[TLF10] Engin Tola, Vincent Lepetit, and Pascal Fua. Daisy: An efficient dense descriptor applied to wide baseline stereo. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(5):815–830, 2010.

[TLZ+19] Zhigang Tu, Hongyan Li, Dejun Zhang, Justin Dauwels, Baoxin Li, and Junsong Yuan. Action-stage emphasized spatio-temporal VLAD for video action recognition. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28:2799–2812, 2019.

[TPBM18] Fabio Tosi, Matteo Poggi, Antonio Benincasa, and Stefano Mattoccia. Beyond local reasoning for stereo confidence estimation with deep learning. In *European Conference on Computer Vision (ECCV)*, 2018.

[TXD+19] Zhigang Tu, Wei Xie, Justin Dauwels, Baoxin Li, and Junsong Yuan. Semantic cues enhanced multi-modality multi-stream CNN for action recognition. *Transactions on Circuits and Systems for Video Technology (TCSVT)*, 29(5):1423–1437, May 2019.

[UZU+17] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. DeMoN: Depth and motion network for learning monocular stereo. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[VBR+05] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27(3):475–480, 2005.

[VSF+18] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *European Conference on Computer Vision (ECCV)*, 2018.

[VSR15] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3D scene flow estimation with a piecewise rigid scene model. *Int. Journal of Computer Vision (IJCV)*, 115(1):1–28, 2015.

[VVB17] Johan Vertens, Abhinav Valada, and Wolfram Burgard. SMSnet: Semantic motion segmentation using deep convolutional neural networks. In *Int. Conference on Intelligent Robots and Systems (IROS)*, 2017.

[WB15] Jonas Wulff and Michael J. Black. Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[WFR+16] Shenlong Wang, Sean Ryan Fanello, Christoph Rhemann, Shahram Izadi, and Pushmeet Kohli. The global patch collider. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[WKR17] Anne S. Wannenwetsch, Margret Keuper, and Stefan Roth. ProbFlow: Joint optical flow and uncertainty estimation. In *Int. Conference on Computer Vision (ICCV)*, 2017.

[WL15] Paul Wohlhart and Vincent Lepetit. Learning descriptors for object recognition and 3D pose estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[WLZ+18] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *Int. Conference on Neural Information Processing Systems (NIPS)*, 2018.

[WRHS13] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. DeepFlow: Large displacement optical flow with deep matching. In *Int. Conference on Computer Vision (ICCV)*, 2013.

[WRHS15] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Learning to detect motion boundaries. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[WS85] David H. Warren and Edward R. Strelow. *Electronic Spatial Sensing for the Blind: Contributions from Perception, Rehabilitation, and Computer Vision*. Springer Netherlands, 1985.

[WS09a] Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest-neighbor classification. *Journal of Machine Learning Research (JMLR)*, 10:207–244, Jun 2009.

[WS09b] Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research (JMLR)*, 10:207–244, Jun 2009.

[WSL+14] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[WSLB17] Jonas Wulff, Laura Sevilla-Lara, and Michael J. Black. Optical flow in mostly rigid scenes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[WT11] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Int. Conference on Machine Learning (ICML)*, 2011.

[WTP+09] Manuel Werlberger, Werner Trobin, Thomas Pock, Andreas Wedel, Daniel Cremers, and Horst Bischof. Anisotropic Huber-L1 optical flow. In *British Machine Vision Conference (BMVC)*, 2009.

[XBZ18] Shuangjie Xu, Linchao Bao, and Pan Zhou. Class-agnostic video object segmentation without semantic re-identification. In *European Conference on Computer Vision (ECCV)*, 2018.

[XFL+18] Huaxin Xiao, Jiashi Feng, Guosheng Lin, Yu Liu, and Maojun Zhang. MoNet: Deep motion exploitation for video object segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[XFYL18] Yu-Syuan Xu, Tsu-Jui Fu, Hsuan-Kung Yang, and Chun-Yi Lee. Dynamic video segmentation network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[XJM12] Li Xu, Jiaya Jia, and Yasuyuki Matsushita. Motion detail preserving optical flow estimation. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(9):1744–1757, 2012.

[XRK17] Jia Xu, René Ranftl, and Vladlen Koltun. Accurate optical flow via direct cost volume processing. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[XWW18] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *European Conference on Computer Vision (ECCV)*, 2018.

[XXFH19] Christopher Xie, Yu Xiang, Dieter Fox, and Zaid Harchaoui. Object discovery in videos as foreground motion clustering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[YHD16] Jason J. Yu, Adam W. Harley, and Konstantinos G. Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *European Conference on Computer Vision (ECCV)*, 2016.

[YL15] Jiaolong Yang and Hongdong Li. Dense, accurate optical flow estimation with piecewise parametric model. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[YMU14] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *European Conference on Computer Vision (ECCV)*, 2014.

[YMYS18] Takuma Yagi, Karttikeya Mangalam, Ryo Yonetani, and Yoichi Sato. Future person localization in first-person videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[ZCZ+18] Zheng Zhang, Dazhi Cheng, Xizhou Zhu, Stephen Lin, and Jifeng Dai. Integrated object detection and tracking with tracklet-conditioned detection. *arXiv pre-print*, 1811.11167, 2018.

[ZDYW18] Xizhou Zhu, Jifeng Dai, Lu Yuan, and Yichen Wei. Towards high performance video object detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[ZF14] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, 2014.

[ZIE17] Richard Zhang, Phillip Isola, and Alexei A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[ŽL14] Jure Žbontar and Yann LeCun. Computing the stereo matching cost with a convolutional neural network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[ZLL18] Qi Zhao, Fangmin Li, and Xinhua Liu. Real-time visual odometry based on optical flow and depth learning. In *Int. Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, 2018.

[ZLNH17] Yi Zhu, Zhenzhong Lan, Shawn Newsam, and Alexander G. Hauptmann. Guided optical flow learning. In *Conference on Computer Vision and Pattern Recognition (CVPR) Workshop*, 2017.

[ZLX18] Haochen Zhang, Dong Liu, and Zhiwei Xiong. Convolutional neural network-based video super-resolution for action recognition. In *Automatic Face & Gesture Recognition (FG)*, 2018.

[ZS01] Zhengyou Zhang and Ying Shan. A progressive scheme for stereo matching. In *3D Structure from Images – SMILE 2000*, 2001.

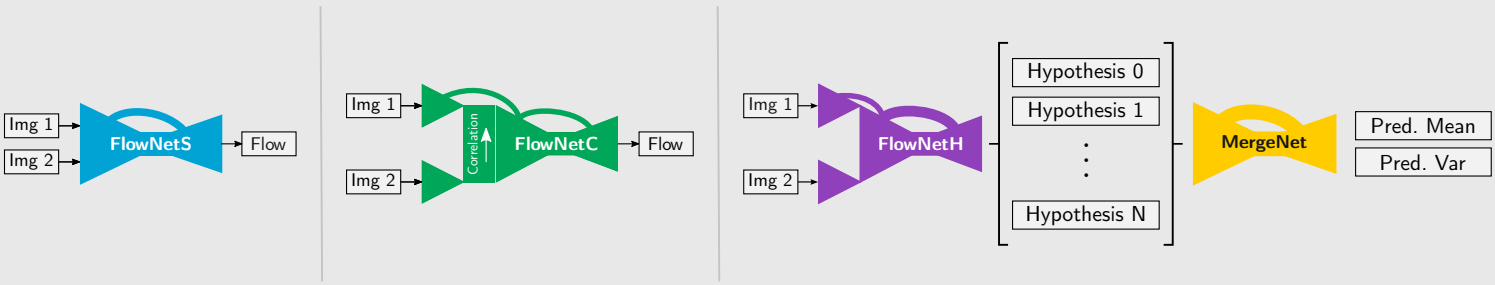
[ZSP+18] Cheng Zhao, Li Sun, Pulak Purkait, Tom Duckett, and Rustam Stolkin. Learning monocular visual odometry with dense 3D mapping from dense 3D flow. In *Int. Conference on Intelligent Robots and Systems (IROS)*, 2018.

[ZW94] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In *European Conference on Computer Vision (ECCV)*, 1994.

Notes

[o] Vision
COMPUTER VISION University of Freiburg

May 2019



Dekanin: Prof. Dr. Hannah Bast

Erstgutachter und Betreuer: Prof. Dr. Thomas Brox
Zweitgutachterin: Prof. Dr. Cordelia Schmid