Towards Geometric Understanding of Motion

Towards Geometric Understanding of Motion

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

Anurag Ranjan
aus Bangalore, Indien

Tübingen
2019

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

Tag der mündlichen Qualifikation:	20.12.2019
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dr. Michael J. Black
2. Berichterstatter:	Prof. Dr. Daniel Scharstein
3. Berichterstatter:	Prof. Dr. Andrew Blake

For Mom, Papa and Ritu

Abstract

The motion of the world is inherently dependent on the spatial structure of the world and its geometry. Therefore, classical optical flow methods try to model this geometry to solve for the motion. However, recent deep learning methods take a completely different approach. They try to predict optical flow by learning from labelled data. Although deep networks have shown state-of-the-art performance on classification problems in computer vision, they have not been as effective in solving optical flow. The key reason is that deep learning methods do not explicitly model the structure of the world in a neural network, and instead expect the network to learn about the structure from data. We hypothesize that it is difficult for a network to learn about motion without any constraint on the structure of the world. Therefore, we explore several approaches to explicitly model the geometry of the world and its spatial structure in deep neural networks.

The spatial structure in images can be captured by representing it at multiple scales. To represent multiple scales of images in deep neural nets, we introduce a *Spatial Pyramid Network* (SpyNet). Such a network can leverage global information for estimating large motions and local information for estimating small motions. We show that SpyNet significantly improves over previous optical flow networks while also being the smallest and fastest neural network for motion estimation. SPyNet achieves a 97% reduction in model parameters over previous methods and is more accurate.

The spatial structure of the world extends to people and their motion. Humans have a very well-defined structure, and this information is useful in estimating optical flow for humans. To leverage this information, we create a synthetic dataset for human optical flow using a statistical human body model and motion capture sequences. We use this dataset to train deep networks and see significant improvement in the ability of the networks to estimate human optical flow.

The structure and geometry of the world affects the motion. Therefore, learning about the structure of the scene together with the motion can benefit both problems. To facilitate this, we introduce *Competitive Collaboration*, where several neural networks are constrained by geometry and can jointly learn about structure and motion in the scene without any labels. To this end, we show that jointly learning single view depth prediction, camera motion, optical flow and motion segmentation using Competitive Collaboration achieves state-of-the-art results among unsupervised approaches.

Our findings provide support for our hypothesis that explicit constraints on structure and geometry of the world lead to better methods for motion estimation.

Zusammenfassung

Bewegungen sind naturgemäß von der räumlichen Struktur der Welt sowie ihrer Geometrie abhängig. Daher versuchen klassische Methoden zur Schätzung des optischen Flusses seit Jahrzehnten, die Geometrie zu modellieren, um die Bewegung zu schätzen. Die jüngsten Methoden der tiefen Lernverfahren verfolgen jedoch einen ganz anderen Ansatz. Sie versuchen den optischen Fluss zu schätzen, indem sie aus annotierten Daten lernen. Obwohl tiefe Netze bei Klassifizierungsproblemen in der Bildverarbeitung derzeit die besten Resultate bringen, sind sie bei der Schätzung des optischen Flusses wenig effektiv. Der Hauptgrund dafür ist, dass tiefe Lernverfahren die Struktur der Welt nicht explizit modellieren, sondern erwarten, dass diese implizit aus Daten gelernt wird. Unsere Hypothese ist, dass es für ein neuronales Netz schwierig ist, etwas über die Bewegung zu erfahren, ohne zusätzliche Information über die Stuktur der Welt zu erhalten. Daher untersuchen wir verschiedene Ansätze, um 3D Geometrie und räumliche Struktur der Welt mit Hilfe tiefer neuronaler Netze explizit zu modellieren.

Die räumliche Struktur eines Bildes kann erfasst werden, indem das Bild in unterschiedlichen Skalierungen dargestellt wird. Um verschiedene Skalierungen in neuronalen Netzen zu nutzen, stellen wir ein *Spatial Pyramid Network (SpyNet)* vor. Ein solches Netzwerk verwendet globale Informationen zur Schätzung großer Bewegungen und lokale Informationen zur Schätzung kleiner Bewegungen. Wir zeigen, dass SpyNet den optischen Fluss viel besser abschätzt als bisherige Netz-werke und gleichzeitig das kleinste und schnellste neuronale Netz ist. SPyNet reduziert die Anzahl der Modellparameter um 97% und erzielt eine bessere Performance als frühere Methoden.

Auch menschliche Bewegungen unterliegen der räumlichen Struktur der Welt. Der Mensch hat eine wohldefinierte Geometrie, was nützlichlich ist, um den opti-schen Fluss für Menschen zu schätzen. Vor diesem Hintergrund erstellen wir einen synthetischen Datensatz für den optischen Fluss des Menschen unter Verwendung eines statistischen Körpermodells und Mocap Sequenzen. Wir verwenden diesen Datensatz um tiefe Netze zu trainieren und sehen eine signifikante Verbesserung hinsichtlich der Schätzung des optischen Flusses.

Die Struktur und Geometrie der Welt ist dauerhaft präsent und beeinflusst Bewegung. Daher kann das Lernen über die Struktur der Szene zusammen mit der Bewegung beiden Problemen zugute kommen. Um dies zu ermöglichen, stellen wir *Competitive Collaboration* vor, eine Methode, bei der mehrere neuronale Netze der Geometrie der Welt unterliegen und dadurch gemeinsam ohne Annotation die Struktur und Bewegung in einer Szene lernen. Zu diesem Zweck zeigen wir, dass das gemeinsame Schätzen von Tiefe aus einzelnen Bildern, Kamerabewegung, optischem Fluss und Bewegungsseg-

mentierung mittels Competitive Collaboration derzeit die besten Ergebnisse unter den unüberwachten Verfahren erzielt.

Unsere Ergebnisse stützen unsere Hypothese, dass explizite Einschränkungen der Struktur und Geometrie der Welt zu besseren Methoden zur Bewegungsschä-tzung führen.

Preface

The following works [103, 108, 106] form the part of this thesis.

1. Anurag Ranjan and Michael J Black. "Optical flow estimation using a spatial pyramid network." IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

2. Anurag Ranjan, Javier Romero, and Michael J Black. "Learning human optical flow." British Machine Vision Conference (BMVC), 2018.

3. Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. "Competitive Collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation." IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

The following works [104, 64, 29, 107, 83, 105] were done during my PhD but do not form a part of this thesis.

1. Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. "Generating 3D faces using convolutional mesh autoencoders." European Conference on Computer Vision (ECCV), 2018.

2. Joel Janai, Fatma Guney, Anurag Ranjan, Michael J Black, and Andreas Geiger. "Unsupervised learning of multi-frame optical flow with occlusions." European Conference on Computer Vision (ECCV), 2018.

3. Daniel Cudeiro, Timo Bolkart, Cassidy Laidlaw, Anurag Ranjan, and Michael J Black. "Capture, learning, and synthesis of 3D speaking styles." IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

4. Anurag Ranjan, Joel Janai, Andreas Geiger, and Michael J Black. "Attacking optical flow." Proceedings of the International Conference on Computer Vision (ICCV), 2019.

5. Qianli Ma, Siyu Tang, Sergi Pujades, Gerard Pons-Moll, Anurag Ranjan, and Michael J Black. "Dressing 3D humans using a conditional Mesh-VAE-GAN." arXiv preprint arXiv:1907.13615, 2019.

6. Anurag Ranjan, David T. Hoffmann, Dimitrios Tzionas, Siyu Tang, Javier Romero, and Michael J. Black. "Learning Multi-Human Optical Flow." arXiv preprint arXiv:1910.11667, 2019.

The data generation code for human optical flow was written by Javier Romero, Dimitrios Tzionas and David Hoffmann. The proof of convergence for Competitive Collaboration was provided by Lukas Balles.

Acknowledgements

The last four years have turned my world around. And, to think of it, I am surprised that this is not an understatement. Deep down, I think my biggest reasons for taking up this PhD were to experience this part of life and rediscover myself. Now, I stand upon this plank and see myself and this world in a different light.

Michael has been around all this time, and will forever remain my teacher. He is a wizard. How his incantations turn in to beautiful research papers is a mystery. What is not a mystery, are his perspectives on life, carefully planted in his conversations. Somewhere in them, you find deep inspiration, long journeys with powerful endings and notions of rediscovery.

Michael has created a brilliant group with beautiful people, and I have been fortunate to be a part of it. I will really miss Melanie, Rocko and Nicole. Melanie supported me throughout the *Kafkaesque* process of graduation. At Perceiving Systems, I found great friends – Timo, Soubhik, Partha and Dimitrios with whom research was so fun, I never realized when it turned into several publications.

During my time at Perceiving Systems, Andrew Blake visited us. A couple of years ago, I had read his book *Active Contours*, and some of the math was way over my head. Quite frankly, I never imagined he would be so fun. He became the *Hero of the day* (Michael's words) after becoming the external reviewer for my thesis on a short notice. He will continue to remain a Hero.

At Perceiving systems, we were also visited by Daniel Scharstein. He is truly the coolest Prof I have ever met, and the best slackliner I personally know. It is an honor for me to have him on my thesis committee.

I also thank Zeynep Akata for considering to examine my thesis defense. I feel lucky to have her on my committee given her expertize in machine learning.

Throughout my time in Tübingen, Andreas Geiger has been a great mentor and collaborator. I had an amazing time collaborating with him and his group – especially Joel Janai, who played a significant role in some of the most exciting projects that we undertook. Andreas supported me throughout my application process to come to the Max Planck Institute, without which I wouldn't be here. I am really happy to have him on my PhD committee.

Going back to early years of my research, I thank Tim Poston, who was considerably my first research advisor during my undergrad. I wish he is at peace. I am grateful to Dinesh Pai, my M.Sc. supervisor at UBC for providing me a great research environment that prepared for me for a successful PhD. I thank Debanga Neog for collaborating with

me on several projects. At UBC, Jim Little, Bob Woodham and David Lowe inspired me to pursue computer vision research. I still remember the computer vision lectures of Bob and David.

While doing my PhD, I also spent time at Facebook and NVIDIA. At Facebook, I thank Manohar Paluri and Jamie Ray who believed my ideas and provided all the support I needed to pursue them. At NVIDIA, Varun made me feel at home and a part of his family. He will continue to be one of my closest friend. I also thank Deqing Sun, Kihwan Kim and Jan Kautz for supporting my research at NVIDIA. During my stay in Cambridge while at NVIDIA, I felt really nice to be around Jonas, Sandra and Lea.

At MPI, Georgios Pavlakos and Carmel Kozlov became a special part of my life during their visit in the summer of 2018. They kept me motivated during some of the stressful times of my PhD. We continue to be close friends.

I will miss so many friends and collaborators – Despoina, Carolin, Simon, Cusuh, Vassilis, Haiwen, Siyu, Qianli, David, Peter Gehler, Sergi, Javier, Aamir, Sergey, Thomas and Joachim. Daniel Cudeiro lives in my memory. A lot of data and infrastructure support for my research was provided by – Senya, Naureen, Jon, Benjamin, Jojumon, Joan, Raffi, Jean-Claude, and Tsvetelina. Linda and Cornelia supported the public outreach of my research. Timo, Chun-Hao, Lea, Siyu and Marilyn provided constructive feedback on several chapters of this thesis.

Life in Tübingen would not have been so fun without some very special people – Chantal, Abhinanda, Priyanka, Vinod, Yamini, Manish, Rewati and Mehdi. Erika lent an immense support during some of my most difficult times.

I wouldn't have started this PhD without the support and motivation from Spoorthi Nayak. She believed in me before I could. Vaibhav Madhok helped me understand what it means to do a PhD.

During my PhD and throughout the years, I have been supported by Badal, Ankan, Sai Deepak, and Ashwin Nayak, whose friendships I cherish.

Mom, Papa and Ritu have given me all the love over all these years. I wouldn't be here without the love and support from Radhe Uncle.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The evolution of life on our planet was shaped by two important processes – motion and vision [75]. The motion here represents the locomotion ability of the organisms, whereas the vision represents the development of the visual systems. Throughout the course of evolution, the synergistic interaction between motion and vision helped in the development of complex organs for human vision and locomotion [75]. Motion, as it seems, has played a crucial role in the development of visual system across the species of the animal kingdom. Early visual systems, categorized by Class 1 type [96], consisted of a few non-directional photo-receptor cells used for estimating the water depth, therefore providing a stimulus for locomotion. The present human visual system, categorized by Class 4 type [96], also uses the synergy between motion and vision for the development of visual cortex. In addition, the middle temporal visual area (MT) in the human visual cortex plays an important role in the perception of this motion in the world [18]. To formalize our notion of motion, Gibson introduced *optical flow* [45], or the apparent motion of pixels in a sequence of images. This is illustrated in Figure 1.1.

1.1 What is Optical Flow?

Consider an image sequence $I(x,y,t)$ parametrized by Euclidean coordinates (x,y) and time t. Then, the optical flow $V = (u,v)$ is defined as the apparent motion of the pixel at (x,y) over one unit of time. In an ideal world, we assume that the brightness of the pixel does not change when it moves. Therefore, the optical flow $V = (u,v)$ satisfies the constraint

$$I(x,y,t) = I(x+u, y+v, t+1). \tag{1.1}$$

Optical flow has been found to be useful in a variety of applications spanning computer vision, computer graphics and robotics. Optical flow has been shown to be useful in two-stream deep networks for action recognition [123] where optical flow is encoded by one of the streams, and images are encoded by the other stream. Sevilla et al. [118] show a case study of how various optical flow methods influence the performance when used in action recognition systems. Optical flow provides weak supervision for training neural networks to segment objects [98]. Optical flow features are also useful in temporal

Figure 1.1: An image from the Sintel optical flow dataset [22] showing the optical flow in the scene marked by arrows and color coded optical flow for each of the pixels in the image on the right. Illustration by J. Wulff and L. Sevilla-Lara.

interpolation of videos [69]. Amiranashvili et al. [6] show that using optical flow as an additional input leads to better performance of robotic agents in a control task. Optical flow is also useful for training networks to solve other computer vision tasks – depth, camera motion and motion segmentation [106].

1.2 Computing Optical Flow

1.2.1 Energy Minimization

Classical methods such as Horn and Schunck [59] approach this problem using non-linear optimization to minimize the energy function

$$E = \sum_x \sum_y f\Big(I(x,y,t), I(x+u_{xy}, y+v_{xy}, t+1)\Big) + g(u_{xy}, v_{xy}) \tag{1.2}$$

where f is an error function, g is a regularization function and (u_{xy}, v_{xy}) is the optical flow at (x,y). The first term f is known as the *brightness consistency* term and the regularizer g is often referred to as the *spatial smoothness* term. The brightness consistency term assumes that the pixel values corresponding to objects or scenes do not change when they move across the image. The constraint provided by the brightness consistency term is insufficient to obtain a unique solution for optical flow. Therefore, an additional constraint is introduced by the smoothness term g that assumes that the optical flow (u,v) is spatially smooth.

Horn and Schunck [59] use an alternating minimization approach over the brightness

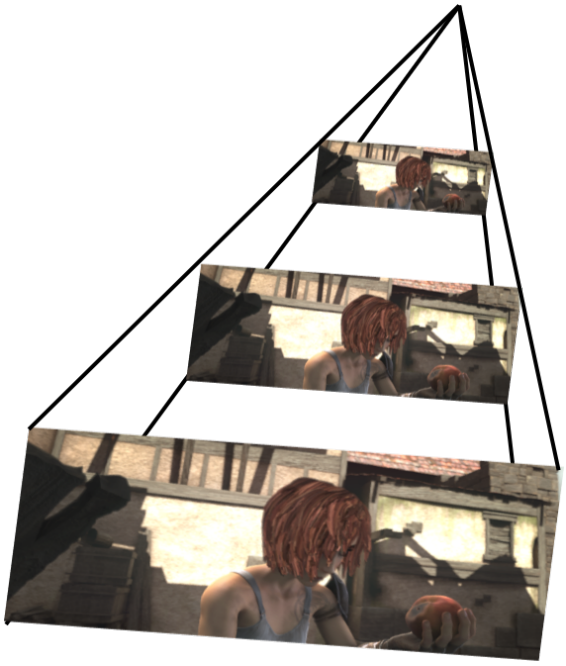


Figure 1.2: A spatial image pyramid consists of an image sampled at multiple scales.

consistency and spatial smoothness term in an energy functional given by

$$E = \sum_x \sum_y ||I(x,y,t) - I(x+u_{xy}, y+v_{xy}, t+1)||^2 + (\nabla^2 u_{xy} + \nabla^2 v_{xy}) \tag{1.3}$$

to solve for optical flow. However, this energy term is convex only if the values of (u,v) are small. Therefore, this formulation makes it intractable to solve for optical flow if the motion is not small. To deal with this, Anandan [7] introduces a coarse-to-fine strategy for optimizing for optical flow values. Under this, the image pairs are sampled in a *spatial pyramid* (see Figure 1.2) at multiple spatial scales of decreasing resolutions. The images at a coarse scale are first used to obtain rough estimates of optical flow using Eq. (1.3), which are then refined by using images at finer scales.

Following Horn and Schunck, several other methods [13, 20, 82] approached the problem of estimating optical flow using optimization and spatial pyramids. These methods improve upon the energy functional Eq. (1.2) by using robust functions [13] to model data f and smoothness g terms. These methods that are based on the brightness consistency and smoothness assumptions are commonly known as *classical methods* for computing optical flow.

1.2.2 Fallacies of Optical Flow Energy

Brightness consistency assumption. The assumptions of brightness consistency and spatial smoothness in classical methods are weak and do not hold true in several cases for general scenes. The brightness consistency holds true only if the surfaces are *Lambertian*, i.e., their reflectence does not depend on the viewing angle. This is generally not the case for real world objects. The specular reflections from the surfaces violate the assumption that the points on an object have the same pixel intensity when the corresponding pixels move in an image. The brightness consistency is also violated in case of occlusions when objects move over each other. In such cases, the object pixels visible in one frame are not visible in other frames to satisfy the data constraint modeled by f in Eq. (1.2). The cases that violate brightness consistency also include transparent objects for which the pixel values depend on several factors such as camera motion, illumination etc. Another case violating this assumption is motion blur introduced by fast moving objects or low frame rate cameras. Classical methods find it difficult to incorporate all these failure cases in the brightness consistency term f in Eq. (1.2) which leads to large errors in optical flow estimation.

Spatial smoothness assumption. The other fallacy of classical methods is the spatial smoothness term g in the energy functional, Eq. (1.2). The spatial smoothness term generally holds true over small regions in the image. However, this assumption is violated in several cases, such as occlusions and motion boundaries. During occlusions, the occluded pixels do not conform to similar optical flow across the occlusion boundaries. In case of motion boundaries, the optical flow values across the boundaries have a discontinuity, and therefore enforcing spatial smoothness at these boundaries compromises the accuracy of the optical flow estimates. Similarly, deformable objects such has human bodies, faces and hands do not conform to smooth flow fields in the self-occluding regions.

Classical methods try to address these two assumptions by using extra terms in the energy functional. However, these assumptions are still present to constrain Eq. (1.2) and therefore estimate optical flow.

1.2.3 Deep Learning Approach

In order to get around the assumptions of brightness consistency and smoothness, we can avoid optimization of the optical flow energy functional Eq. (1.2), and instead learn from data. Consider that we have large samples of consecutive image pairs I^1, I^2 and their corresponding ground-truth flow $V = (u, v)$, making a dataset $\mathcal{D} = \{(I_i^1, I_i^2, V_i), i = [1, 2, 3, ...]\}$. We can use a deep neural network $F(\theta)$ parametrized by θ and train it on the dataset $\mathcal{D}$ to regress to optical flow given a pair of images (see Figure 1.3). The neural

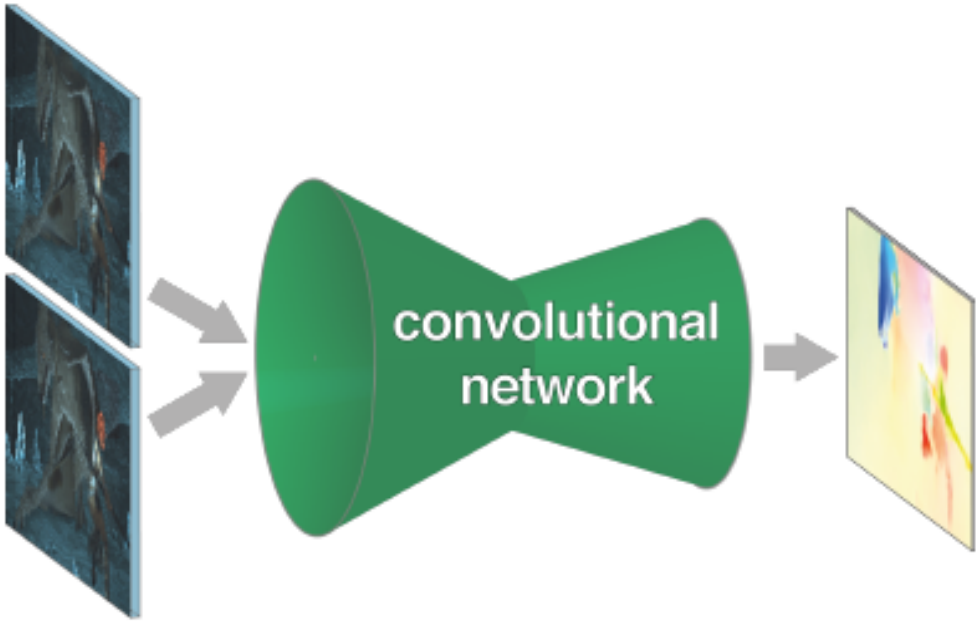


Figure 1.3: An alternative to minimizing optical flow energy is to use a convolutional neural network (CNN). A representative CNN regresses to optical flow using a pair of image frames. Reproduced from [33].

network F is trained by minimizing the L1-norm,

$$\min_{\theta} \mathbb{E}_{I_i^1, I_i^2, V_i \sim \mathcal{D}} ||F(I_i^1, I_i^2; \theta) - V_i||_1 \tag{1.4}$$

using stochastic gradient descent. FlowNet [33] was the first method to compute optical flow following this approach by using deep convolutional networks. Although this attempt was promising, the results were not as accurate as the classical methods. Particularly, FlowNet estimates noisy optical flow even when identical images are supplied wherein the true optical flow between the images is zero. This is shown in Figure 1.4.

1.2.4 Spatial Pyramid Network

FlowNet employs a deep-learning approach to learn to regress optical flow using a large synthetic dataset. We argue that there is an alternative approach that combines the best approaches of classical methods and deep learning. We introduce a *Spatial Pyramid Network* or *SPyNet* [103] that learns to compute optical flow by combining a classical spatial-pyramid formulation with deep learning. This estimates large motions in a coarse-to-fine approach by warping one image of a pair at each pyramid level by the current flow estimate and computing an update to the flow. Instead of the standard minimization of an

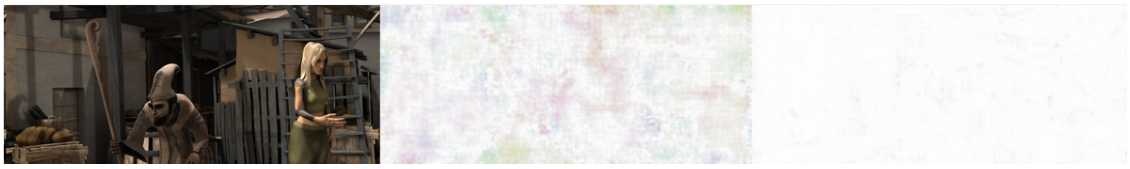


Image $I^1 = I^2$ FlowNet SPyNet

Figure 1.4: When identical images are input to FlowNet [33], the network hallucinates noisy flow estimates. In contrast, SPyNet [103] correctly predicts zero flow.

objective function at each pyramid level, we train one deep network per level to compute the flow update. Unlike the FlowNet approach, the networks do not need to deal with large motions; these are dealt with by the pyramid. This has several advantages.

1. The Spatial Pyramid Network is simpler and 96% smaller than FlowNet [33] in terms of model parameters. This makes it more efficient and appropriate for embedded applications.

2. Since the flow at each pyramid level is small ($<$ 1 pixel), a convolutional approach applied to pairs of warped images is appropriate.

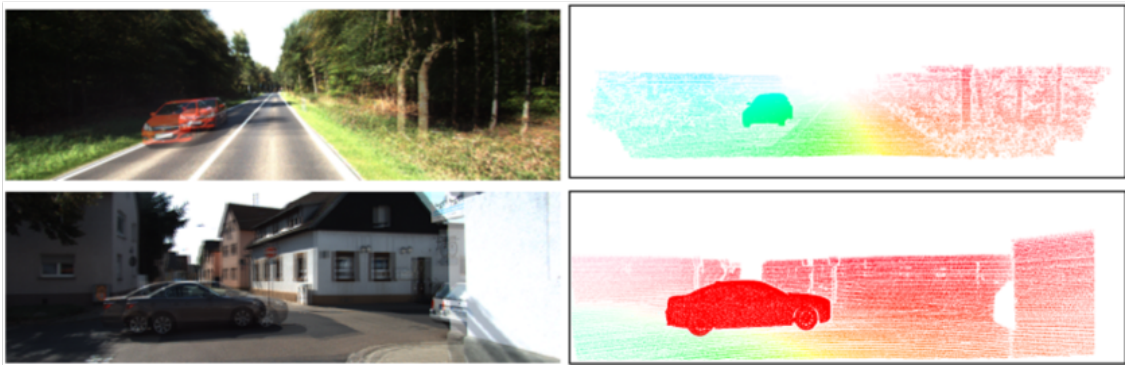
3. Unlike FlowNet, the learned convolution filters appear similar to classical spatio-temporal filters, giving insight into the method and how to improve it.

The results obtained using SPyNet are more accurate than FlowNet on standard benchmarks, suggesting a new direction of combining classical methods with deep learning.

1.3 Optical Flow Datasets

The progress of optical flow methods is also attributed to standard datasets in the field. The Middlebury dataset [10] was the first standard dataset to serve as a benchmark for optical flow methods. The dataset consists of 8 image pairs and their ground-truth optical flow. Geiger et al. [43] introduced the KITTI dataset which has sparse ground truth for driving scenes. The KITTI dataset was constructed using LIDAR and provides 200 images and their ground-truth flow fields. Butler et al. [22] introduced the Sintel optical flow benchmark that consists of synthetically generated optical flow fields from a CGI movie. Sintel provides 1041 image pairs with their ground-truth optical flow. However, these datasets have several limitations. Real-world datasets like Middlebury and KITTI are still very small, and are not diverse. Synthetic datasets like Sintel are also small for large-scale training of deep neural networks. Moreover, there is a domain gap between synthetic datasets and the real world that can affect the generalization ability of optical flow networks trained on synthetic data. In Figure 1.5, we show examples from Sintel and KITTI datasets.

Sintel



KITTI

Figure 1.5: Image and corresponding optical flow samples from Sintel [22] and KITTI [43] datasets.

1.3.1 Large-Scale Datasets

To address the problem of curating large-scale data for optical flow, Dosovitskiy et al. [33] introduced the *Flying Chairs* dataset. This is a synthetic dataset containing 22,872 image pairs with their optical flow ground truth. The dataset is simple. It is constructed by placing chairs of different shapes, poses, and colors on a random background image. A 2D affine motion is then applied to the chair to synthesize image pairs and corresponding optical flow fields. Although the dataset is simple, it is large enough for training deep neural networks. Dostovitsky et al. [33] used this dataset to train the first neural network, FlowNet to compute optical flow. Since FlowNet is trained using this synthetic dataset, it does not generalize well on real-world sequences due to the domain gap.

1.3.2 Human Optical Flow Dataset

To close the domain gap between synthetic and real datasets, we tackle the specific case of human motion. Human motion is very special. Humans are deformable and do not conform to rigid motion. As a result, the Flying Chairs dataset, which was constructed by applying rigid motions to rigid objects (chairs), does not resemble the characteristics of deformable human motion. Moreover, other characteristics of human motion such as self-occlusion due to moving limbs, shape and pose changes are not present in the Flying Chairs dataset. To address the limited availability of optical flow datasets containing human motion, we construct the *Human Optical Flow* dataset [108]. For this, we use a 3D model of the human body and motion capture data to synthesize realistic flow fields in both single- and multi-person images.

We then train a convolutional neural network to estimate human flow fields from pairs of images. We demonstrate that our trained networks are more accurate than a wide range of top methods on held-out test data and that they generalize well on real image sequences.

Learning optical flow for humans shows that enforcing structure on scenes can improve performance of optical flow methods. In this case, the structure is imposed by using a realistic human body model [81] and human motion capture data [62]. Such constraints about the structure can also be enforced on general scenes. Using contraints about the structure of the world such as depth and rigidity, MR-Flow [153] achieves state of the art results on optical flow benchmarks. However, Wulff et al. [153] approach the problem by energy minimization and therefore their method is very slow. In contrast, using such reasoning about geometry and structure of the world with deep networks can lead to better accuracy and faster running time.

1.4 Learning Geometric Reasoning

To incorporate the geometry of the world in the problem of motion estimation, we need to jointly model structure of the 3D world along with its motion. The 3D structure of the world can be represented using depth maps. The motion has two components – the motion of the observer and the motion of independent objects. Several works have approached these problems independently.

Zhou et al. [159] show that view synthesis of neighboring frames in a video sequence is sufficient for unsupervised learning of single view depth prediction and camera odometry. Following this, Yin et al. [156] show that one can learn optical flow in addition to depth and odometry. However, these methods do not account for independently moving objects in the scene. By segmenting the static scene from independently moving objects, we can reason about the static scene using depth and odometry, and reason about the independently moving objects using optical flow. Therefore, there are four problems that can be solved here – single view depth prediction, camera motion estimation, optical

flow, and segmentation of a video into the static scene and moving regions. In practice, we need four neural networks to solve each of the problems.

1.4.1 Competitive Collaboration

Training multiple neural networks using a single objective is tricky and can be seen as a multi-player game. In order to train these networks, we introduce *Competitive Collaboration*, a framework that facilitates the coordinated training of multiple specialized neural networks. In this framework, two networks act as adversaries and compete with each other. The competition between the networks is moderated by another network, the Moderator. Concurrently, these competitors form a consensus in alternating cycles to collaborate and train the moderator. By doing so, all networks in the framework can be trained to solve specific tasks.

1.4.2 Joint Unsupervised Learning of Depth, Camera Motion, Optical Flow and Motion Segmentation

Using Competitive Collaboration, we address the unsupervised learning of four interconnected problems – single view depth prediction, camera motion estimation, optical flow and segmentation of a video into the static scene and moving regions. As shown in Figure 1.6, given an image, we can segment it into static scene and moving regions. The motion in the static scene is a result of the depth of the scene and the camera motion exclusively. Therefore, using a brightness consistency loss on the pixels of the static scene in a video sequence, we can estimate the depth and camera motion. The independently moving regions cannot be accounted for by using depth and camera motion across the frames of a video sequence. Therefore, we account for these pixels using optical flow. However, these problems are interconnected, and we can solve for motion segmentation by looking at the consistency between depth, camera motion and optical flow.

Our key insight is that these four fundamental vision problems are coupled through geometric constraints. Consequently, learning to solve them together simplifies the problem because the solutions can reinforce each other. In the Competitive Collaboration framework, neural networks act as both competitors to explain pixels that correspond to static or moving regions, and as collaborators through a moderator that assigns pixels to be either static or independently moving. In this way, our novel method integrates all these problems in a common framework and simultaneously reasons about the segmentation of the scene into moving objects and the static background, the camera motion, depth of the static scene structure, and the optical flow of moving objects. Our model is trained without any supervision and achieves state-of-the-art performance among joint unsupervised methods on all sub-problems.

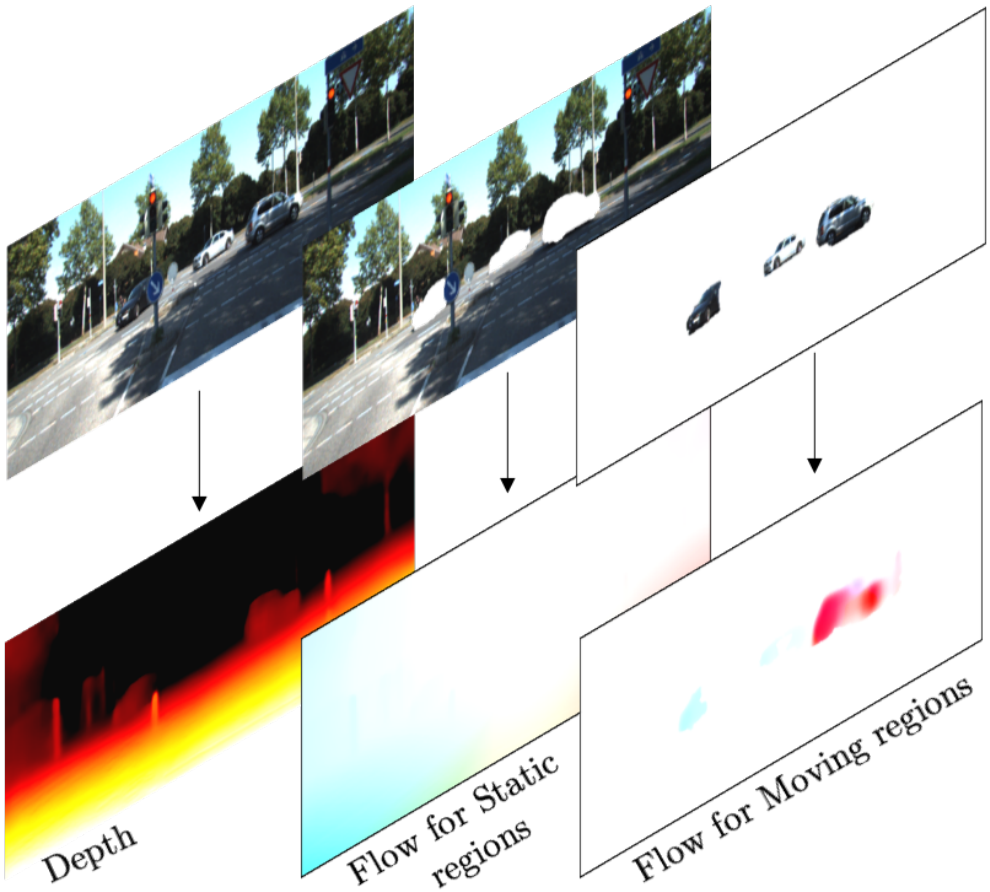


Figure 1.6: An image can be segmented into static regions and independently-moving regions. The static regions are accounted for by depth and camera motion, and the moving regions are accounted for by optical flow. Using Competitive Collaboration [106], we can jointly train for depth, camera motion, optical flow and motion segmentation in a completely unsupervised framework.

1.5 Thesis Organization

The thesis is organized as follows.

Chapter 2 surveys related work. We discuss classical optical flow methods, and their problems, and various ways of dealing with these problems in the improved versions of classical methods. We then go over end-to-end training of deep neural networks using synthetic datasets. We discuss unsupervised learning of optical flow and the explicitly modeling geometry in a deep learning framework.

In Chapter 3, we introduce the Spatial Pyramid Network and show how to train this network to estimate optical flow. We experimentally evaluate our method on optical flow benchmarks such as Middlebury [10], KITTI [43], and Sintel [22]. We also evaluate the performance of SPyNet in terms of speed and memory.

In Chapter 4, we discuss generation of the Human Optical Flow dataset using a realistic 3D human body model and captured MoCap sequences. We improve upon SPyNet and train this network on our Human Optical Flow dataset. We evaluate the performance on our generated dataset and show qualitative results on real-world videos consisting of human motion.

In Chapter 5, we introduce the Competitive Collaboration framework with three neural networks, where two adversaries compete with each other through a moderator network that moderates the competition. We study the convergence properties and generalization of Competitive Collaboration.

In Chapter 6, we show joint unsupervised learning of single-view depth prediction, camera odometry, optical flow and motion segmentation. We train four different networks jointly to solve each of the subproblems using Competitive Collaboration. We evaluate the performance of all the subproblems on the raw KITTI [43] dataset.

In Chapter 7, we conclude by discussing the importance of geometrical constraints in training deep neural networks for computer vision tasks. We also discuss ways to make these systems more robust and accurate for the future.

Chapter 2

Related Work

2.1 Classical Optical Flow

The classical formulation of the optical flow problem involves optimizing the sum of a data term based on brightness constancy and a spatial smoothness term [59]. Classical methods suffer from the fact that they make assumptions about the image brightness change and the spatial structure of the flow that do not match reality. Early efforts make the formulation more robust [13] using robust M-estimation. Many methods focus on improving robustness by changing the assumptions. These optical flow methods include pre-processing the images by filtering [147] and augmenting them with gradients [20]. Different scaling factors in the spatial pyramid are used [128]. A median filter [126, 147] is used to significantly boost performance. Feature matching [19] and semantic segmentation [119] are integrated more recently, resulting in performance improvement.

Spatial Pyramids. The use of spatial pyramids for flow estimation has a long history [21, 46] with its first use in the classical flow formulation appearing in Glazer [46]. Typically, Gaussian or Laplacian pyramids are used to deal with large motions. Generally, gradient-based optical flow estimation relies on the computation of spatial and temporal derivatives of the image sequence, implemented as filters [58]. This formulation assumes that the image motion is small. Through spatial smoothing and subsampling, this small motion assumption holds at high levels of the pyramid. A coarse-to-fine approach then estimates the flow such that, at each pyramid level, the motion is updated with an estimate that is assumed to be small.

The methods based on spatial pyramid structure are well known to have problems when small objects move quickly. To deal with them, Brox et al. [19] incorporate long range matching into the traditional optical flow objective function. This approach of combining image matching to capture large motions, with a variational [110] or discrete optimization [51] for fine motions, can produce accurate results. Alternatively, Sevilla et al. [120] replace the classical pyramid with a channel representation in which the spatial pyramid is effectively applied to segmented images, preserving small structures at high levels of the pyramid.

Layered Methods. The brightness consistency term of the optical flow energy Eq. (1.2) does not hold at the motion boundaries when objects are moving over each other accompanied by occlusions. To deal with this, Jepson and Black [67] model optical flow over a patch using a mixture of smoothly varying layers. Wang and Adelson [143] decompose an image into foreground and background layers using a mask and independently estimate motion for each of the layers. Ayer and Sawhney [8] formulate this problem of layered representation using maximum-likelihood estimation to model multiple object motions in different layers of an image. Cremers and Soatto [28] use a variational approach for segmenting the image plane into a set of regions of parametric motion.

Sun et al. [129] formulate a layered model to segment the image into foreground and background using a Conditional Random Field (CRF). The optical flow for each of the segments is computed by constraining it to the segments. Sevilla et al. [119] build upon the layered formulation to segment image in semantically meaningful layers for computing flow. Further, Wulff et al. [153] segment the image into static regions and moving objects to obtain the optical flow using a plane+parallax approach.

Geometric Methods. Modeling the spatial structure of images has proven useful in estimation of motion. One can also model the structure of the world, in terms of its geometry and this has also been shown to be useful for motion estimation. Ullman [137] shows that we can recover three-dimensional scene structure and relative motion from a single monocular video sequence. Several works [146, 157] also combine estimation of depth together with motion under an approach known as motion-stereo. Vedula et al. [141] propose three-dimensional scene flow combining depth and optical flow to estimate 3D velocities of all the points in images.

Using a prior on the geometry has also been shown to be useful for optical flow estimation. Guney et al. [50] model the geometry of cars in their approach to compute scene flow. Sevilla et al. [119] incorporate the geometry of the scene using its semantics by classifying the scene in to static regions, regions having affine motion, and deformable regions. However, the scene geometry is not explicitly modeled. Wulff et al. [153] explicitly model geometry by factoring the scene in to static regions and independently moving regions. For estimating the optical flow of static regions, they simply compute the depth and camera motion which simplifies the optimization and therefore improves accuracy. Optical flow is estimated using depth and camera motion of a static scene using a closed form solution (see Appendix A.1). The moving regions do not follow any geometric contraints and therefore their optical flow is independently estimated.

2.2 Optical Flow meets Machine Learning

The key advantage of learning to compute flow is that we do not hand craft assumptions. Rather, the variation in image brightness and spatial smoothness are embodied in the learned model.

Possibly the first attempt to learn a model to estimate optical flow is the work of Freeman et al. [39] using a Markov Random Field (MRF). Freeman et al. consider a simple synthetic world of uniform moving blobs with ground-truth flow. They vector-quantize training patches to compute neighborhood co-occurrence statistics. Then, they perform belief propagation in an MRF to estimate flow using the learned model. However, their training data is not realistic and they do not apply the method to real image sequences. Black et al. [15] use a PCA basis to model the distribution of optical flow. They estimate optical flow using linear combination of these basis flows.

Roth and Black [115] learn a field-of-experts (FoE) model to capture the spatial statistics of optical flow. The model is trained using flow fields generated from laser scans of real scenes and natural camera motions. However, the data includes only the flow fields without the corresponding images. Consequently, their method only learns the spatial statistics of the flow. Sun et al. [128] describe the first fully learned model that can be interpreted as a shallow convolutional neural network. They formulate a classical flow problem with a data term and a spatial term. The spatial term uses the FoE model from Roth and Black [115], while the data term replaces traditional derivative filters with a set of learned convolutional image filters. With limited training data and a small set of filters, their method does not show the full promise of learning flow. Rosenbaum et al. [114] model the spatial statistics of optical flow using Gaussian Mixture Models. Wulff and Black [151] learn the spatial statistics of optical flow by applying robust PCA [54] to real (noisy) optical flow computed from natural movies. While this produces a global flow basis and overly smooth flow, they use the model to compute reasonable flow relatively quickly.

Several methods rely on matching image features using deep networks, and then use those matches as a constraint in the optical flow energy function Eq. (1.2). Weinzaepfel et al. [149] use convolutional features in a variational optimization framework thereby improving the optical flow accuracy. In contrast, Guney et al. [51] use deep features in discrete optimization of optical flow gaining even better performance. However, deep matching methods [51, 110, 149] do not fully solve the problem, since they resort to classical methods to compute the final flow field.

Spatio-temporal Filters and Human Perception. Learning spatial statistics of optical flow shows a correlation between learned spatiotemporal filters and motion models from neuroscience [30]. Various methods show that spatio-temporal filters emerge from learning, for example using independent component analysis [139], sparseness [97], and multi-layer models [23]. Burt and Adelson [3] lay out the theory of spatiotemporal models for motion estimation and Heeger [57] provides a computational embodiment. While inspired by human perception, such methods do not perform well [12].

Memisevic and Hinton learn a simple spatial transformations with a restricted Boltzmann machine [88], and find a variety of filters. Taylor et al. [133] use synthetic data to learn "flow-like" features using a restricted Boltzmann machine but do not evaluate flow

accuracy. Han et al. [52] propose a rich set of filters for representing a "video primal sketch" but do not learn the filters or use them for estimating optical flow.

2.3 Optical Flow meets Deep Learning

The above learning methods suffer from limited training data and the use of shallow models. In contrast, deep convolutional neural networks emerge as a powerful class of models for solving recognition [56, 132] and dense estimation [25, 79] problems.

2.3.1 Optical Flow Networks

Early work use neural networks trained for other tasks such as segmentation to constrain the optimization objective of optical flow. This is mainly because of unavailability of large scale optical flow datasets at the time.

With the release of Flying Chairs dataset, FlowNet [33] was the first deep convolutional architecture for flow estimation that is trained end-to-end. FlowNet introduced two different types of networks – FlowNetS and FlowNetC. Both networks have a simple U-Net architecture consisting of an encoder and a decoder, each of which are made from a block of convolutional layers. The encoder and decoder have skip connections so that the features computed at the encoder are also available to the decoder. For FlowNetS, the input frames are stacked together and passed as input. In contrast, in FlowNetC, the input frames are not stacked but a cost-volume between their features is computed. This is contrasted in Table 2.1.

The cost-volume of input features is obtained by cross-correlating them. The cost-volume is estimated using

$$c(x_1,x_2) = \sum_{o\in[-k,k]\times[-k,k]} \langle f_1(x_1+o), f_2(x_2+o)\rangle, \tag{2.1}$$

where f_1, f_2 are input features maps. For each point (x_1,x_2), the cross-correlation is defined by dot product of elements in the $k-$distance neighborhood. Once the cost-volume is obtained, it is processed through the network to get optical flow prediction. The cost volume helps in getting long range correspondences and therefore it is better for capturing large motions in image sequences.

Both FlowNetS and FlowNetC networks show promising results, despite being trained on Flying Chairs dataset, which is a synthetic dataset of chairs flying over randomly selected images. Despite promising results, the method lags behind the state-of-the-art in terms of accuracy. The major limitations come from – (a) synthetic dataset on which it is trained, and (b) the architecture of the neural networks.

To deal with the domain gap between synthetic data and real world videos, Tran et al. [135] use a traditional flow method to create "semi-truth" training data. Then, they

	Stacked Frames	Correlation
U-Net	FlowNetS [33]	FlowNetC [33]
Spatial Pyramids	SPyNet [103]	PWC-Net [130]

Table 2.1: Contrasting different optical flow network architectures based on input processing pipeline.

train a 3D convolutional network on this data. However, the performance of their method does not exceed the performance of the traditional flow method that was used to create the training data.

The Spatial Pyramid Network (SPyNet) [103] discussed in Chapter 3 approaches the problem by bringing together ideas from classical methods [46] and integrating them with deep networks. SPyNet consists of multiple small networks in a spatial pyramid. The image pair is downsampled at multiple scales in the pyramid and a network computes residual flow at each level of the pyramid. SPyNet is trained on the same data as FlowNet and is more accurate. Furthermore, it is a simpler architecture with 96% fewer parameters than FlowNet.

Ilg et al. [61] propose FlowNet 2.0, an architecture composed by cascading together multiple FlowNetC and FlowNetS networks. This network is trained on even more data, including animated sequences to close the domain gap with Sintel [22], and synthetic driving scenes to close the domain gap with KITTI [43]. FlowNet 2.0 is a big improvement over FlowNet and subsequently SPyNet, but it is a large network with a complex architecture.

It is evident from SPyNet [103] that small networks in a pyramid can outperform large networks even when trained on the same data. Sun et al. [130] propose PWC-Net consisting of a correlation block similar to FlowNetC [33] but integrate it in a spatial pyramid. This architecture is simpler than FlowNet2.0 with fewer parameters but achieves similar performance. In Table 2.1, we show how different ideas from classical methods like correlation and spatial pyramids integrate themselves in deep networks. Recently, Hur and Roth [60] propose iterative refinement of optical flow over coarse-to-fine pyramid levels using the same network in recurrent fashion.

2.3.2 Unsupervised Learning of Optical Flow

The problem with training optical flow networks is that they need large scale datasets. These datasets are synthetic and therefore always have some domain gap with the real world. Therefore, a group of methods formulate the problem from an unsupervised learning perspective.

The simplest approach is to use a deep network and minimize the optical flow energy Eq. (1.2) on the flow predictions of the network given the unlabeled image sequences [5,

66]. These optical flow estimates are reasonable but are far behind the state-of-the-art optical flow methods. Meister et al. [87] add forward-backward constraint to the energy term by using a Siamese network. In addition, they also reason about occlusions using a heuristic to deal with areas where the brightness consistency term from Eq. (1.2) does not hold. This method achieves better performance on KITTI using unsupervised training followed by supervised fine-tuning on just 400 images of the KITTI dataset.

Further improvements are obtained by using multiple frames [64] and explicitly modelling the occlusion term by predicting it from a network. Janai et al. [64] show that using multiple frames can lead to significant improvement in optical flow predictions. More recently, Wulff et al. [152] show that one can also learn optical flow by training a network for temporal interpolation and then fine tuning it only on a small number of images.

Although unsupervised learning of optical flow network shows promising results, the methods that use supervision using synthetic data supersede in performance.

2.3.3 Small and Fast Optical Flow Networks

Traditionally, optical flow has been treated as an optimization problem that is approached by minimizing the optical flow energy Eq. (1.2). This makes it very slow to compute. Several recent methods attempt to balance speed and accuracy, with the goal of real-time processing and reasonable (though not top) accuracy. GPU-flow [150] began this trend but several methods now outperform it. PCA-Flow [151] runs on a CPU, is slower than frame rate, and produces overly smooth flow fields. EPPM [11] achieves similar, middle-of-the-pack, performance on Sintel (test), with similar speed on a GPU. DIS-Fast [73] is a GPU method that is significantly faster than previous methods but is also significantly less accurate.

In contrast, CNN methods have an advantage over traditional variational methods in that they do not perform iterative optimization to propagate information spatially. All the neural networks viz. FlowNet, SPyNet, PWC-Net can process video frames in real time. SPyNet and PWC-Net are really small in terms of memory footprint which makes their deployment easy on embedded systems. Deep learning has led to a class of optical flow methods that obtain highly accurate real-time optical flow using a small computational cost.

2.4 Large Scale Datasets

Several datasets have been developed to facilitate training and benchmarking of optical flow methods. However, early datasets are not large or diverse enough to support training of deep neural networks. The Middlebury dataset [10] contains 8 pairs of training frames with ground truth, and is limited to small motions. Since then, optical flow datasets have become larger and more challenging. The KITTI dataset [43] is larger but is focused on

rigid scenes and automotive motions [43]. The KITTI dataset has 200 pairs of images and ground-truth flow. However, the optical flow ground truth in KITTI is sparse and only exists for roads and cars.

The Sintel dataset [22] is larger than KITTI with 1041 pairs of images and ground truth flow. The dataset contains complex deformations as well, such as motion of an animated character, wings of a dragon, etc. The low-level statistics of optical flow in Sintel is representative of the real-world motion [22]. However, since the data is generated using computer graphics, complex and high-level structure of the world is not well represented. These datasets are mainly used for evaluation of optical flow methods and are generally too small to support training neural networks.

To learn optical flow using neural networks, more datasets have emerged that contain examples on the order of tens of thousands of frames. The Flying Chairs [33] dataset contains about $22,000$ samples of chairs moving against random backgrounds. Although it is not very realistic or diverse, it provides training data for neural networks [33, 103] that achieve reasonable results on optical flow benchmarks. More synthetic datasets [86, 41] for optical flow are especially designed for training deep neural networks and improve the realism in the dataset. Flying Things [86] contains tens of thousands of samples of random 3D objects in motion. The Monkaa and Driving scene datasets [86] contain frames from animated scenes and virtual driving respectively. Virtual KITTI [41] uses graphics to generate scenes like those in KITTI and is two orders of magnitude larger. Recent synthetic datasets [40] show that synthetic data can train networks that generalize to real scenes.

These datasets have a common property. The scenes generally have rigid objects and rigid motions. In contrast, human bodies are non-rigid and undergo deformation. Therefore, networks trained on these datasets perform poorly [108] when humans are present in the scene. To address this Varol et al. [140] introduce the SURREAL dataset with human bodies. The dataset uses 3D human meshes rendered on top of color images to train networks for 2D pose estimation, depth estimation, and body part segmentation. While not fully realistic, they show that this data is sufficient to train methods that generalize to real data. In a similar fashion, [160, 91] render synthetic color images for 3D hand pose estimation. In Chapter 4, we go beyond these works to address the problem of optical flow by synthesizing a dataset focused at human motion.

2.5 Human Motion

Using images to analyze human motion dates to the origins of photography. In 1884, Marey [85], a French physician captured the change in joint positions across time by capturing multiple photos in a single photographic plate. A hundred years later, Johansson [70] established the basis for modern motion capture (MoCap) in his work using point lights attached to the body. To this day, reasoning about human motion using sparse set of points is a very common approach in fields like human pose estimation [95].

However, other holistic approaches that reason about the full extent of the body have existed too. Parametric models using generalized local cones [93], superquadrics [42], 2d part templates [37] or more recently triangulated meshes [17] have been used to model full body motion.

Human motion can be understood from 2D motion. Early work shows that simple motion history images can be used to recognize actions without computing dense flow [31]. Motion history images are sensitive to viewpoint changes. To address this, Winland et al. [148] generalize motion history images to motion history volumes using multiple camera views. Black et al. [15] use principal component analysis (PCA) to parametrize human motion but use noisy flow computed from image sequences for training data. They learn linear motion models of specific motions like walking and mouth movement for the estimation of optical flow. Blank et al. [16] represent human motion as three-dimensional shapes induced by the silhouettes in the space-time volume. Yilmaz and Shah [155] use moving contours to represent human motion. Laptev et al. [76] compute optical flow in a local neighborhood and aggregate it in histograms, called histograms of optical flow to represent complex human motion.

More similar to us, Fablet and Black [35] use a 3D articulated body model and motion capture data to project 3D body motion into 2D optical flow. They then learn a view-based PCA model of the flow fields. They detect and track human motion by estimating the parameters of this flow model given image sequences. In contrast to these methods, we generate more realistic training data using a statistical model of the human body together with realistic image sequences. Rather than training a model of the flow statistics, we use the images and flow to train a CNN to directly estimate flow from images.

Only a few works in pose estimation exploit human motion and in particular several methods [38, 162] use optical flow constraints to improve 2D human pose estimation in videos. Similar work [24, 99] propagates pose results temporally using optical flow to encourage time consistency of the estimated bodies. Apart from its application in warping between frames, the structural information existing in optical flow is used for pose estimation alone [112].

Wang and Mori [144] propose max-margin hidden conditional random field (MHCRF) on optical flow features for action recognition. Recent state-of-the-art action recognition methods use two-stream deep neural networks [36, 123], where one of the streams is used to process images and the other is used to process precomputed optical flow across the image sequences.

Although motion has been used for a number of applications related to human perception, there has been far less attention [15] to estimate human motion itself. Human-specific patterns are not used as much for estimation of optical flow.

2.6 Geometry meets Deep Learning

Although curating large scale datasets has led to success of deep-learning-based methods on optical flow, they still lag in accuracy compared to methods [119, 153] that exploit structure of the scene to compute motion. Joint estimation frameworks have been shown to be useful in the past where problems could reinforce each other by constraining the geometry of the world. Optical flow estimation can provide information about camera motion [53] and vice versa. Depth and camera motion can provide information about optical flow if there are no moving objects in the scene [159]. Optical flow can be used to reason about motion segmentation [4, 109] and vice versa [14].

Flow estimation provides correspondences between views or consecutive frames which are essential to unveil the relative camera pose between the views [53]. If the observed scene only contains rigid static background, the flow with a known pose can be seen as a camera ego-motion, which can provide correct disparities for 3D scene structure [94]. Because all these cues are tied and constrain each other, often times, solving one problem can help other problems [94].

Recently, it has been shown that these problems can be effectively combined by segmenting image into static scene and moving regions (e.g. [153]). However, these methods are based on classical optimization and tends to be very slow.

More recent works [84, 138, 142, 156, 159] approach joint estimation of structure and motion by coupling two or more problems together in an unsupervised learning framework. Zhou et al. [159] introduce joint unsupervised learning of ego-motion and depth from multiple unlabeled frames. To account for moving objects, they learn an explainability mask. However, these masks also capture model failures such as occlusions at depth discontinuities, and are therefore not useful for motion segmentation.

Mahjourian et al. [84] use a more explicit geometric loss, an ICP constraint to jointly learn depth and camera motion for rigid scenes. Yin et al. [156] add a refinement network to Zhou et al. [159] to also estimate residual optical flow. The estimation of residual flow is designed to account for moving regions, but there is no coupling of the optical flow network with the depth and camera motion networks. Residual optical flow is obtained using a cascaded refinement network, thus preventing other networks from using flow information to improve themselves. Therefore, recent works show good performance either on depth and camera motion [84, 156, 159] or on optical flow [87], but not on both.

Zou et al. [161] exploit consistency between depth and optical flow to improve performance on both the problems. In Chapter 6, we add the key missing piece to jointly learn the segmentation of the scene into static and independently-moving regions. This allows the networks to use geometric constraints where they apply and generic flow where they do not. This introduces a framework where motion segmentation, flow, depth and camera motion models can be coupled and solved jointly to reason about the complete geometric structure and motion of the scene.

Chapter 3

Spatial Pyramid Networks

Recent years have seen significant progress on the problem of accurately estimating optical flow, as evidenced by improving performance on increasingly challenging benchmarks. Despite this, most flow methods are derived from a "classical formulation" that makes a variety of assumptions about the image, from brightness constancy to spatial smoothness. These assumptions are only coarse approximations to reality and this likely limits performance. The recent history of the field has focused on improving these assumptions or making them more robust to violations [13]. This has led to steady but incremental progress.

3.1 Introduction

An alternative approach abandons the classical formulation altogether and starts over using recent neural network architectures [33, 135]. Such an approach takes a pair (or sequence) of images and learns to directly compute flow from them. Ideally such a network would learn to solve the correspondence problem (short and long range), learn filters relevant to the problem, learn what is constant in the sequence, and learn about the spatial structure of the flow and how it relates to the image structure. The first attempts are promising but are not yet as accurate as the classical methods.

Goal. We argue that there is an alternative approach that combines the best of both approaches. Decades of research on flow has produced well-engineered systems and principles that are effective. But there are places where these methods make assumptions that limit their performance. Consequently, here we apply machine learning to address the weak points, while keeping the engineered architecture, with the goal of

- improving performance over existing neural networks and the classical methods upon which our work is based;
- achieving real-time flow estimates with accuracy better than the much slower classical methods;

- reducing memory requirements to make flow more practical for embedded, robotic, and mobile applications.

Problem. Computing flow requires the solution to two problems. One is to solve for long-range correlations while the other is to solve for detailed sub-pixel optical flow and precise motion boundaries. The first neural network method, FlowNet [33], attempts to learn both of these at once. In contrast, we tackle the latter using deep learning and rely on existing methods to solve the former.

Approach. To deal with large motions, we adopt a traditional coarse-to-fine approach [46] using a spatial pyramid[1]. At the top level of the pyramid, the assumption is that the motions between frames are smaller than a few pixels and that, consequently, the convolutional filters can learn meaningful temporal structure. At each level of the pyramid we solve for the flow using a convolutional network and up-sample the flow to the next pyramid level. As is standard with classical formulations [127], we warp one image towards the other using the current flow, and repeat this process at each pyramid level. Instead of minimizing a classical objective function at each level, we learn a convolutional network to predict the *flow increment* at that level. We train the network from coarse to fine to learn the flow correction at each level and add this to the flow output of the network above. The idea is that the displacements are then always less than one (or a few) pixels at each pyramid level.

We call the method *SPyNet*, for Spatial Pyramid Network, and train it using the same Flying Chairs data as FlowNet [33]. We report similar performance as FlowNet on Flying Chairs and Sintel [22] but are significantly more accurate than FlowNet on Middlebury [10] and KITTI [43] after fine tuning. The total size of SPyNet is 96% smaller than FlowNet, meaning that it runs faster, and uses much less memory. The expensive iterative propagation of classical methods is replaced by the non-iterative computation of the neural network.

We do not claim to solve the full optical flow problem with SPyNet; we address the same problem as traditional approaches and inherit some of their limitations. For example, it is well known that large motions of small or thin objects are difficult to capture with a pyramid representation. We see the large-motion problem as separate, requiring different solutions. Rather, what we show is that the traditional problem can be reformulated, portions of it can be learned, and performance improves in many scenarios.

Additionally, because our approach connects past methods with new tools, it provides insights into how to move forward. In particular, we find that SPyNet learns spatio-temporal convolutional filters that resemble traditional spatio-temporal derivative or Gabor filters [3, 57]. The learned filters resemble biological models of motion-processing filters in cortical areas, MT and V1 [122]. The MT-V1 filters have also been shown useful for optical flow estimation [26, 124]. This is in contrast to the random-looking

[1] This, of course, has well-known limitations, which we discuss later.

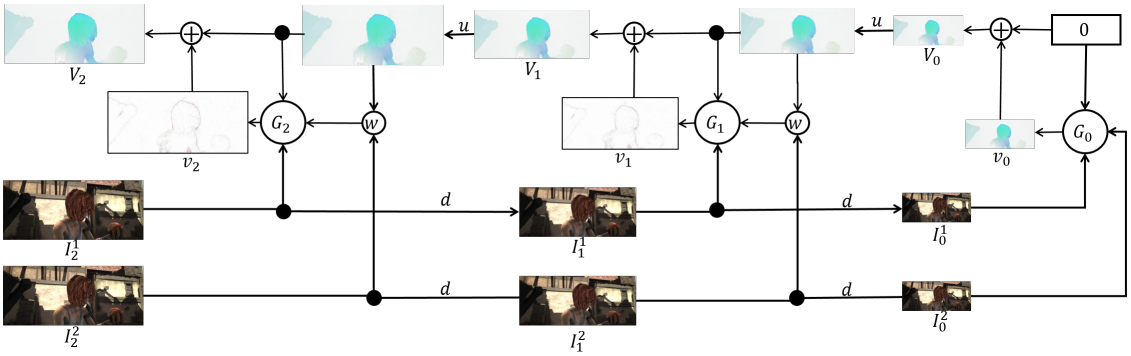


Figure 3.1: Inference in a 3-Level Pyramid Network [32]: The network G_0 computes the residual flow v_0 at the highest level of the pyramid (smallest image) using the low resolution images $\{I_0^1, I_0^2\}$. At each pyramid level, the network G_k computes a residual flow v_k which propagates to each of the next lower levels of the pyramid in turn, to finally obtain the flow V_2 at the highest resolution. u is upsampling operator and d is downsampling operator.

filters learned by FlowNet (see Figure 3.7). This suggests that it is time to reexamine older spatio-temporal filtering approaches with new tools.

In summary our contributions are the following.

1. We combine traditional coarse-to-fine pyramid methods with deep learning for optical flow estimation.
2. We introduce a new SPyNet model that is 96% smaller and is faster than FlowNet.
3. SPyNet achieves comparable or lower error than FlowNet on standard benchmarks – Sintel, KITTI and Middlebury.
4. The learned spatio-temporal filters provide insight about what filters are needed for flow estimation.
5. The trained network and related code are publicly available for research[2] purposes. Several methods [130, 64] have built on our approach.

3.2 Spatial Pyramid Network

Our approach uses the coarse-to-fine spatial pyramid structure of [32] to learn residual flow at each pyramid level. Here we describe the network and training procedure. We introduce a small change in notation in this chapter to improve readability. While we use V to represent optical flow, and we use u to represent an upsampling operator and v to represent the intermediate residual flows.

[2] `https://github.com/anuragranj/spynet`

3.2.1 Spatial Sampling

Let I be an $m \times n$ image with dimensions that are powers of 2. Let $d(.)$ be the downsampling function that decimates I to the corresponding image $d(I)$ of size $m/2 \times n/2$. Let $u(.)$ be the reverse operation that upsamples images by a factor of 2. These operators are bilinear and are also used for resampling the the optical flow field, V. We also define a warping operator $w(I,V)$ that warps the image, I according to the flow field, V, using bilinear interpolation.

3.2.2 Inference

Let $\{G_0,...,G_K\}$ denote a set of trained convolutional neural network (convnet) models, each of which computes residual flow, v_k

$$v_k = G_k(I_k^1, w(I_k^2, u(V_{k-1})), u(V_{k-1})) \tag{3.1}$$

at the k-th pyramid level. The convnet G_k computes the residual flow v_k using the upsampled flow from the previous pyramid level, V_{k-1}, and the frames $\{I_k^1, I_k^2\}$ at level k. The second frame I_k^2 is warped using the flow as $w(I_k^2, u(V_{k-1}))$ before feeding it to the convnet G_k. The flow, V_k at the k-th pyramid level is then

$$V_k = u(V_{k-1}) + v_k. \tag{3.2}$$

As shown in Fig. 3.1, we start with downsampled images $\{I_0^1, I_0^2\}$ and an initial flow estimate that is zero everywhere to compute the residual flow $v_0 = V_0$ at the top of the pyramid. We upsample the resulting flow, $u(V_0)$, and pass it to the network G_1 along with $\{I_1^1, w(I_1^2, u(V_0))\}$ to compute the residual flow v_1. At each pyramid level, we compute the flow V_k using Equation ((3.2)). The flow V_k is similarly propagated to higher resolution layers of the pyramid until we obtain the flow V_K at full resolution. Figure 3.1 shows the working of our approach using a 3-level pyramid. In practice, we use a 5-level pyramid ($K = 4$).

3.2.3 Training and Network Architecture

We train each of the convnets $\{G_0,...,G_K\}$ independently and sequentially to compute the residual flow v_k given the inputs $\{I_k^1, w(I_k^2, u(V_{k-1})), u(V_{k-1})\}$. We compute target residual flows $\hat{v}_k$ as a difference of target flow $\hat{V}_k$ at the k-th pyramid level and the upsampled flow, $u(V_{k-1})$ obtained from the trained convnet of the previous level

$$\hat{v}_k = \hat{V}_k - u(V_{k-1}). \tag{3.3}$$

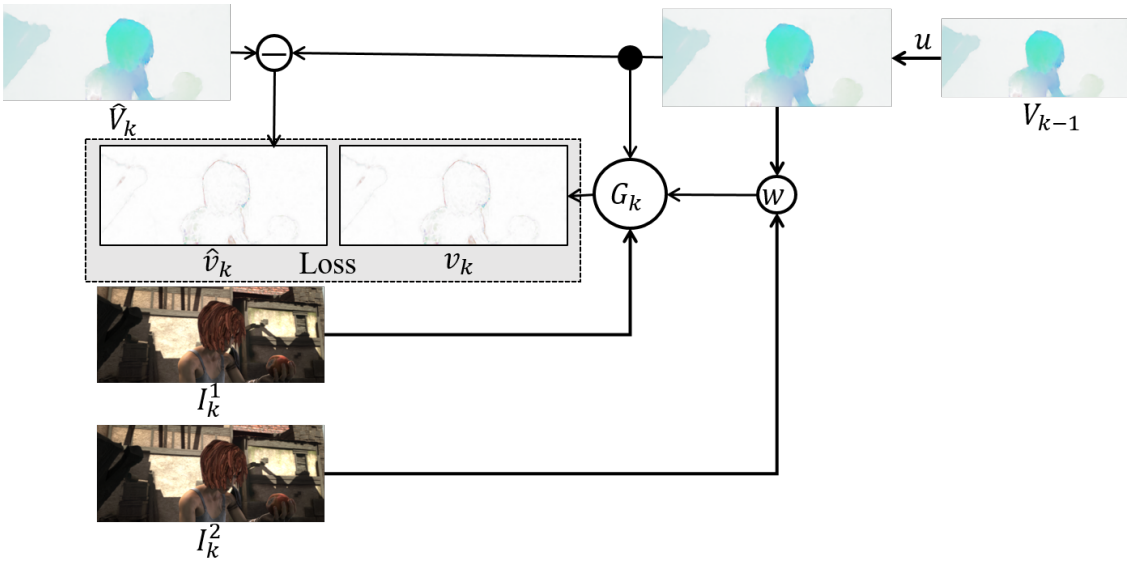


Figure 3.2: Training network G_k requires trained models $\{G_0 ... G_{k-1}\}$ to obtain the initial flow $u(V_{k-1})$. We obtain ground truth residual flows $\hat{v}_k$ by subtracting downsampled ground truth flow $\hat{V}_k$ and $u(V_{k-1})$ to train the network G_k using the EPE loss.

As shown in Fig. 3.2, we train each of the networks, G_k, to minimize the average End Point Error (EPE) loss between the residual flow v_k and $\hat{v}_k$

$$\frac{1}{m_k n_k} \sum_{x,y} \sqrt{(v_k^x - \hat{v}_k^x)^2 + (v_k^y - \hat{v}_k^y)^2}$$

where the $m_k \times n_k$ is the image dimension at level k and the x and y superscripts indicate the horizontal and vertical components of the flow vector.

Each level in the pyramid has a simplified task relative to the full optical flow estimation problem; it only has to estimate a small-motion update to an existing flow field. Consequently each network can be simple. Here, each G_k has 5 convolutional layers; this gave the best combination of accuracy, size, and speed. We train five convnets $\{G_0, ..., G_4\}$ at different resolutions of the Flying Chairs dataset. The network G_0 is trained with 24x32 images. We double the resolution at each pyramid level and finally train the convnet, G_4 with a resolution of 384x512.

Each convolutional layer is followed by a Rectified Linear Unit (ReLU), except the last one. We use 7x7 convolutional kernels for each layer; these worked better than smaller filters. The number of feature maps in each convnet G_k are {32, 64, 32, 16, 2}. The image I_k^1 and the warped image $w(I_k^2, u(V_{k-1}))$ have 3 channels each (RGB). The upsampled flow $u(V_{k-1})$ is 2 channel (horizontal and vertical). We stack image frames together with the upsampled flow to form an 8 channel input to each G_k. The output is 2 channel flow corresponding to velocity in x and y.

Using Torch7[3], we train five networks $\{G_0, ..., G_4\}$ such that each network G_k uses the previous network G_{k-1} as initialization. The networks are trained using Adam [71] optimization with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We use a batch size of 32 across all networks with 4000 iterations per epoch. We use a learning rate of 1e-4 for the first 60 epochs and decrease it to 1e-5 until the networks converge. We use the Flying Chairs [33] dataset and MPI Sintel [22] for training. We trained G_0 for three days and $\{G_1, G_2, G_3, G_4\}$ for one day each on a single Titan X.

We include various types of data augmentation during training. We randomly scale images by a factor between $[1, 2]$ and apply rotations at random within $[-17°, 17°]$. We then apply a random crop to match the resolution of the convnet, G_k being trained. We include additive white Gaussian noise sampled uniformly from $\mathcal{N}(0, 0.1)$. We apply color jitter with additive brightness, contrast and saturation sampled from a Gaussian, $\mathcal{N}(0, 0.4)$. We finally normalize the images using a mean and standard deviation computed from a large sample of ImageNet [116] data in [56].

3.3 Experiments

We evaluate our performance on standard optical flow benchmarks and compare with FlowNet [33] and Classic+NLP [127], a traditional pyramid-based method. We compare performance using average end-point error (EPE) in Table 3.1. We evaluate on all the standard benchmarks and find that SPyNet is the most accurate overall, with and without fine tuning (details below). Additionally SPyNet is faster than all other methods as shown in Table 3.2.

Note that the FlowNet results reported on the MPI-Sintel website are for a version that applies variational refinement ("+v") to the convnet results. Here we are not interested in the variational component and only compare the results of the convnet output.

Flying Chairs. SPyNet achieves better performance than FlowNetS [33] on the Flying Chairs dataset, however FlowNetC [33] performs better than ours. We show the qualitative results on Flying Chairs dataset in Figure 3.3 and compare the performance in Table 3.1.

MPI-Sintel. The resolution of Sintel images is 436x1024. To use SPyNet, we scale the images to 448x1024, and use 6 pyramid levels to compute the optical flow. The networks used on each pyramid level are $\{G_0, G_1, G_2, G_3, G_4, G_4\}$. We repeat the network G_4 at the sixth level of pyramid for experiments on Sintel. Because Sintel has extremely large motions, we found that this gives better performance than using just five levels.

We evaluate the performance of our model on MPI-Sintel [22] in two ways. First, we directly use the model trained on the Flying Chairs dataset and evaluate our performance

[3] `http://torch.ch/`

Method	Sintel Clean		Sintel Final		KITTI		Middlebury		Chairs
	train	test	train	test	train	test	train	test	test
Classic+NLP	4.13	6.73	5.90	8.29	-	-	0.22	0.32	3.93
FlowNetS	4.50	7.42	**5.45**	**8.43**	**8.26**	-	1.09	-	2.71
FlowNetC	4.31	7.28	5.87	8.81	9.35	-	1.15	-	**2.19**
SPyNet	**4.12**	**6.69**	5.57	**8.43**	9.12	-	**0.33**	**0.58**	2.63
FlowNetS+ft	3.66	6.96	4.44	**7.76**	7.52	9.1	0.98	-	3.04
FlowNetC+ft	3.78	6.85	5.28	8.51	8.79	-	0.93		**2.27**
SPyNet+ft	**3.17**	**6.64**	**4.32**	8.36	8.25	10.1	**0.33**	**0.58**	3.07
SPyNet+ft*	-	-	-	-	**3.36**	**4.1**	-	-	-

Table 3.1: Average end-point errors (EPE). Results are divided into methods trained with (+ft) and without fine tuning. SPyNet+ft* uses additional training data compared to FlowNet+ft. Bold font indicates the most accurate results among the convnet methods.

	Classic+NLP	FlowNetS	FlowNetC	SPyNet
Time (s)	102	0.080	0.150	**0.069**

Table 3.2: Average running time of optical flow methods. All run times are measured on Flying Chairs and exclude image loading time.

on both the training and the test sets. Second, we extract a validation set from the Sintel training set, using the same partition as [33]. We fine tune our model independently on the Sintel Clean and Sintel Final split, and evaluate the EPE. The fine-tuned models are listed as "+ft" in Table 3.1. We show the qualitative results on MPI-Sintel in Figure 3.4.

Table 3.3 compares our fine-tuned model with FlowNet [33] for different velocities and distances from motion boundaries. We observe that SPyNet is more accurate than FlowNet for all velocity ranges except the largest displacements (over 40 pixels/frame). SPyNet is also more accurate than FlowNet close to motion boundaries (see Figure 3.4), which is important for many problems.

KITTI and Middlebury. We evaluate KITTI [43] scenes using the base model SPyNet trained on Flying Chairs. Here we also fine-tune the model using Driving and Monkaa scenes from [86] and evaluate the fine-tuned model SPyNet+ft*. Fine tuning results in a significant improvement in accuracy by about 5 pixels. The large improvement suggests that better datasets are needed and that these could improve the accuracy of SPyNet further on general scenes.

For the Middlebury [10] dataset, we evaluate the sequences using the base model SPyNet as well as SPyNet+ft, which is fine-tuned on the Sintel-Final dataset; the Middlebury dataset itself is too small for fine-tuning. SPyNet is significantly more accurate

Figure 3.3: Visualization of optical flow estimates using our model (SPyNet) and the corresponding ground-truth flow fields on the Flying Chairs dataset.

on Middlebury, where FlowNet has trouble with the small motions. Both learned methods are less accurate than Classic+NL on Middlebury but both are also significantly faster.

3.4 Analysis

In this section, we analyse our method with previous work with respect to size of our model, performance-accuracy trade off and the characteristics of filters learned by the network.

Model Size. Combining spatial pyramids with convnets results in a huge reduction in model complexity. At each pyramid level, a network, G_k, has 240,050 learned parameters. The total number of parameters learned by the entire network is 1,200,250, with 5

Figure 3.4: Visual comparison of optical flow estimates using our SPyNet model with FlowNet on the MPI Sintel dataset. The top five rows are from the Sintel Final set and the bottom five rows are from the Sintel Clean set. Each set of 5 rows is sorted in the order of increasing average displacements. SPyNet performs particularly well when the motions are relatively small.

Method	$d_{0\text{-}10}$	$d_{10\text{-}60}$	$d_{60\text{-}140}$	$s_{0\text{-}10}$	$s_{10\text{-}40}$	s_{40+}
FlowNetS+ft	7.25	4.61	**2.99**	1.87	5.83	43.24
FlowNetC+ft	7.19	4.62	3.30	2.30	6.17	**40.78**
SpyNet+ft	**6.69**	**4.37**	3.29	**1.39**	**5.53**	49.71

(a) Errors on Sintel Final

Method	$d_{0\text{-}10}$	$d_{10\text{-}60}$	$d_{60\text{-}140}$	$s_{0\text{-}10}$	$s_{10\text{-}40}$	s_{40+}
FlowNetS+ft	5.99	3.56	2.19	1.42	3.81	40.10
FlowNetC+ft	5.57	3.18	1.99	1.62	3.97	**33.37**
SpyNet+ft	**5.50**	**3.12**	**1.71**	**0.83**	**3.34**	43.44

(b) Errors on Sintel Clean

Table 3.3: Comparison of FlowNet and SpyNet on the Sintel benchmark for different velocities, s, and distances, d, from motion boundaries.

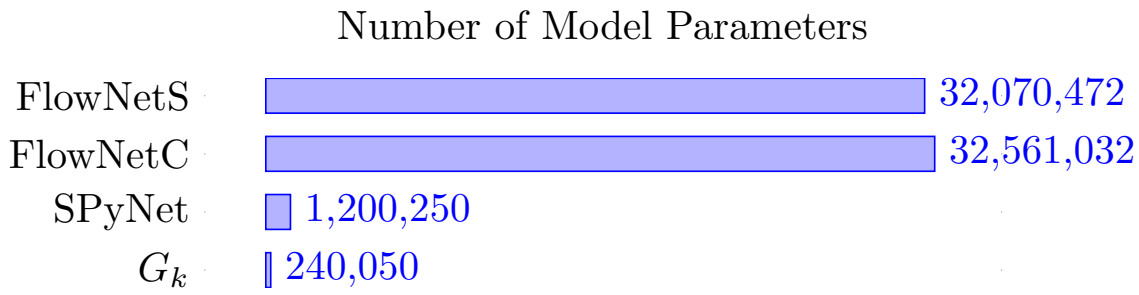


Figure 3.5: Model size of various methods. Our model is 96% smaller than FlowNet.

spatial pyramid levels. In comparison, FlowNetS and FlowNetC [33] have 32,070,472 and 32,561,032 parameters respectively. SPyNet is about 96 % smaller than FlowNet (Figure 3.5).

The spatial pyramid approach enables a significant reduction in model parameters without sacrificing accuracy. There are two reasons – the warping function and learning of residual flow. By using the warping function directly, the convnet does not need to learn it. More importantly, learning residual flows restricts the range of flow fields in the output space. Each network only has to model a smaller range of velocities at each level of the spatial pyramid.

SPyNet also has a small memory footprint. The disk space required to store all the model parameters is 9.7 MB. This simplifies deployment on mobile or embedded devices with GPU support; this is future work.

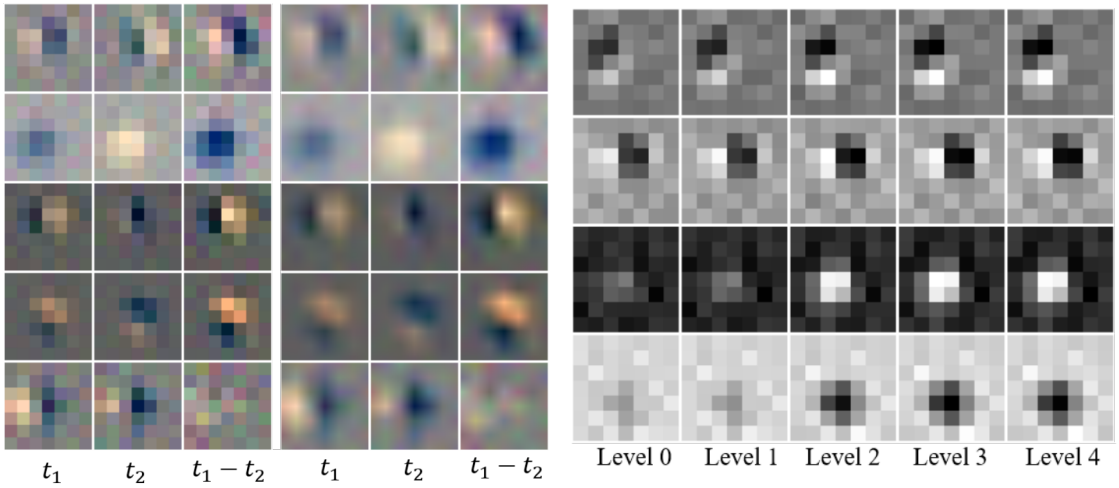


Figure 3.6: (a) Visualization of filter weights in the first layer of G_2 showing their spatiotemporal nature on RGB image pairs using nearest-neighbor (left) and bilinear (right) interpolation. (b) Evolution of some example filters across the pyramid levels (from low resolution (0) to high resolution (4))

Visualization of Learned Filters. Figure 3.6(a) shows examples of filters learned by the first layer of the network, G_2. In each row, the first two columns show the spatial filters that operate on the RGB channels of the two input images respectively. The third column is the difference between the two spatial filters, hence representing the temporal features learned by our model. We observe that most of the spatio-temporal filters in Figure 3.6(a) are equally sensitive to all color channels, and hence appear mostly grayscale. Note that the actual filters are 7×7 pixels and are upsampled for visualization.

We observe that many of the spatial filters appear to be similar to traditional Gaussian derivative filters used by classical methods and have a center-surround structure. These classical filters are hand crafted and typically are applied in the horizontal and vertical direction. Here we observe a greater variety of derivative-like filters of varied scales and orientations. We also observe that some of filters spatially resemble second derivative or Gabor filters [3]. The temporal filters show a clear derivative-like structure in time.

Figure 3.6(b) illustrates how filters learned by the network at each level of the pyramid differ from each other. Recall that, during training, each network is initialized with the network before it in the pyramid. The filters, however, do not stay exactly the same with training. Most of the filters in our network look like rows 1 and 2, where the filters become sharper with increasing contrast as we progress towards the finer-resolution levels of the pyramid. However, there are some filters that are similar to rows 3 and 4. These filters become more defined at higher resolution levels of the pyramid.

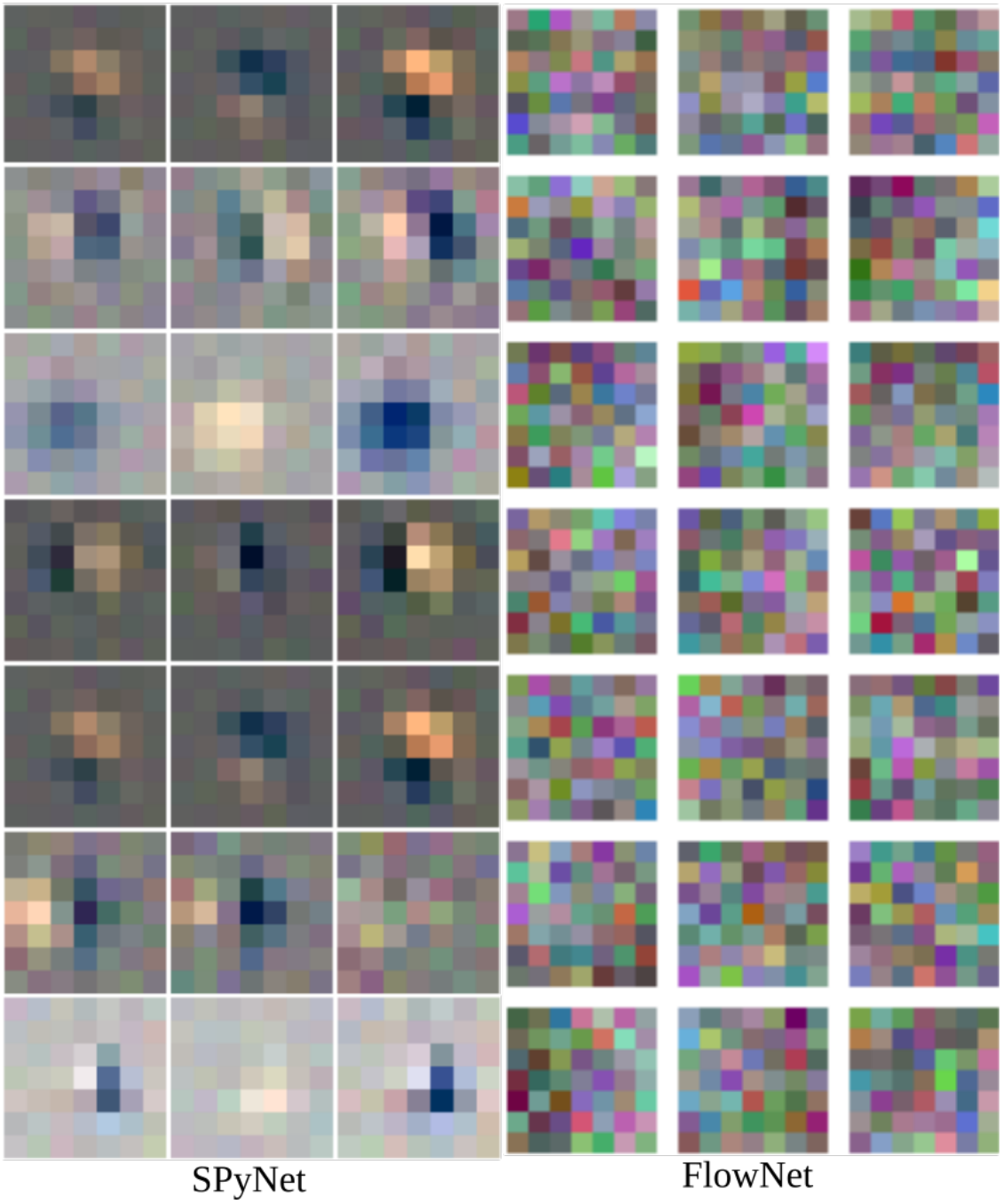


Figure 3.7: Comparison of learned spatio-temporal filters of SPyNet (left) and FlowNet (right)

In Figure 3.7, we compare our learned spatio-temporal filters with the spatial filters learned by FlowNet [33]. We observe that first layer filters of FlowNet [33] appear more random than the Gabor-like filters common to most neural networks or previous spatio-temporal filters. We suspect this is because first layer windows do not overlap common image patches when the motion exceeds a few pixels. In contrast, our pyramid structure means that the motions are always small and the convolutional windows are always looking at related patches in two neighboring frames. We find that this leads to filters, even in the first layer, that resemble classical spatio-temporal filters.

Speed. Optical flow estimation is traditionally viewed as an optimization problem involving some form of variational inference. This is computationally expensive, often taking seconds or minutes per frame. This has limited the application of optical flow in robotics, embedded systems, and video analysis. Using a GPU can speed up traditional methods [131, 150] but with reduced accuracy. Feed forward deep networks [33] leverage fast GPU convolutions and avoid iterative optimization.

Figure 3.8 shows the speed-accuracy comparisons of several well known methods. All times shown are measured with the images already loaded in the memory. The errors are computed as the average EPE of both the clean and final MPI-Sintel sequences. SPyNet offers a good balance between speed and accuracy; no faster method is as accurate.

Optical flow on identical frames. Computing optical flow on identical frames can provide a lot of information about the characteristics of the trained network. In order to validate our network performance on zero motion, we experiment by passing identical frames as the input to our network. We use the test set of Sintel Clean, which was never seen by our network as inputs. We observe an average flow magnitude of 0.06 pixels for identical frames, which is significantly smaller than the average flow magnitude in Sintel. The average flow magnitude in Sintel is 18.926. We show qualitative results in Figure 3.9.

3.5 Discussion

Traditional flow methods linearize the brightness constancy equation resulting in an optical flow constraint equation implemented with spatial and temporal derivative filters. Sometimes methods adopt a more generic filter constancy assumption [2, 20]. Our filters are somewhat different. The filters learned by SPyNet are used in the direct computation of the flow by the feed-forward network.

SPyNet is small compared with other recent optical flow networks. Examination of the filters, however, suggests that it might be possible to make it significantly smaller still. Many of the filters resemble derivative of Gaussian filters or Gabor filters at various scales, orientations, spatial frequencies, and spatial shifts. Given this, it may be possible

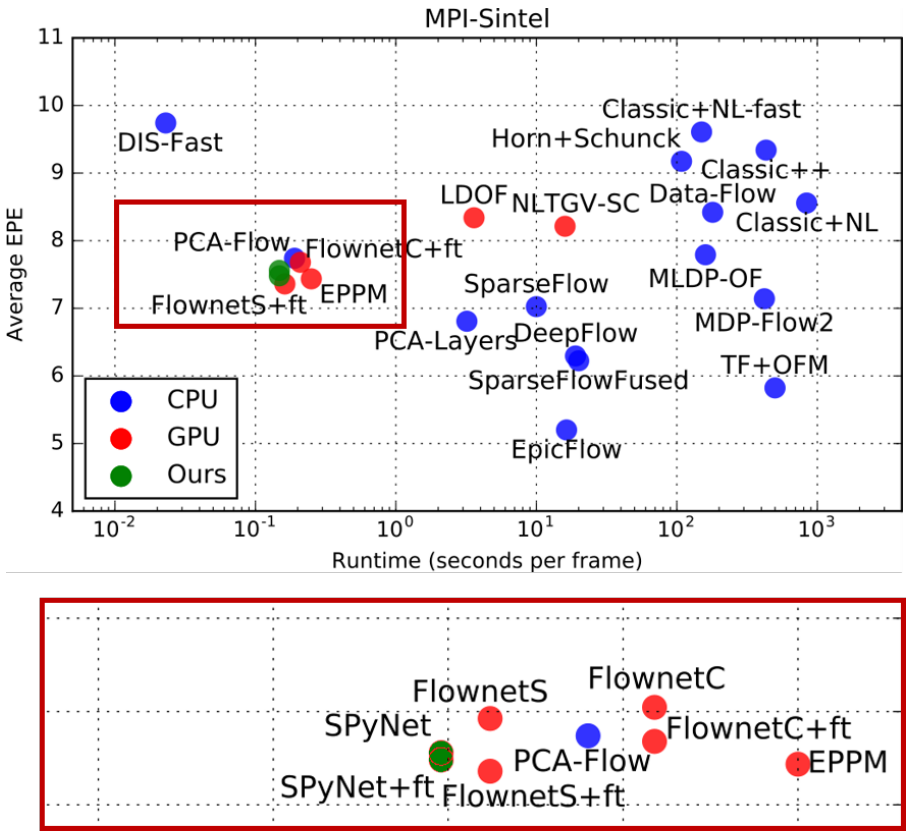


Figure 3.8: Average EPE vs. runtime on MPI-Sintel. Zoomed in version on the bottom shows the fastest methods as of December 2016. Times were measured by us. Adapted from [151].

to significantly compress the filter bank by using dimensionality reduction or by using a set of analytic spatio-temporal features. Some of the filters may also be separable.

Early methods for optical flow used analytic spatio-temporal features but, at the time, did not produce good results and the general line of spatio-temporal filtering decayed. The difference from early work is that our approach suggests the need for a large filter bank of varied filters. Note also that these approaches considered only the first convolutional layer of filters and did not seek a "deep" solution. This all suggests the possibility that a deep network of analytic filters could perform well. This could vastly reduce the size of the network and the number of parameters that need to be learned.

Note that pyramids have well-known limitations for dealing with large motions [19, 120]. In particular, small or thin objects that move quickly effectively disappear at coarse pyramid levels, making it impossible to capture their motion. Recent approaches for dealing with such large motions use sparse matching to augment standard pyramids [19, 149]. Future work should explore adding long-range matches to SPyNet. Alternatively Sevilla et al. [120] define a channel constancy representation that preserves fine structures

Figure 3.9: Optical flow using both frames and identical frames. Note that flow for the latter case for SPyNet is nearly zero. The flow on identical frames has been magnified by 10x for visualization.

in a pyramid. The channels effectively correspond to filters that could be learned.

A spatial pyramid can be thought of as the simple application of a set of linear filters. Here we take a standard spatial pyramid but one could learn the filters for the pyramid itself. SPyNet also uses a standard warping function to align images using the flow computed from the previous pyramid level. This too could be learned.

An appealing feature of SPyNet is that it is small enough to fit on a mobile device. Future work will explore a mobile implementation and its applications. Additionally we will explore extending the method to use more frames (e.g. 3 or 4). Multiple frames could enable the network to reason more effectively about occlusion.

3.6 Conclusion

In summary, we have described a new optical flow method that combines features of classical optical flow algorithms with deep learning. In a sense, there are two notions of "deepness" here. First we use a "deep" spatial pyramid to deal with large motions. Second we use deep neural networks at each level of the spatial pyramid and train them to estimate a flow *update* at each level. This approach means that each network has less work to do than a fully generic flow method that has to estimate arbitrarily large

motions. At each pyramid level we assume that the motion is small (on the order of a pixel). This is borne out by the fact that the network learns spatial and temporal filters that resemble classical derivatives of Gaussians. Because each sub-task is so much simpler, our network needs many fewer parameters than previous methods like FlowNet. This results in a method with a small memory footprint that is faster than existing methods. At the same time, SPyNet achieves an accuracy comparable to FlowNet, surpassing it in several benchmarks. This opens up the promise of optical flow that is accurate, practical, and widely deployable.

Chapter 4

Learning Human Optical Flow

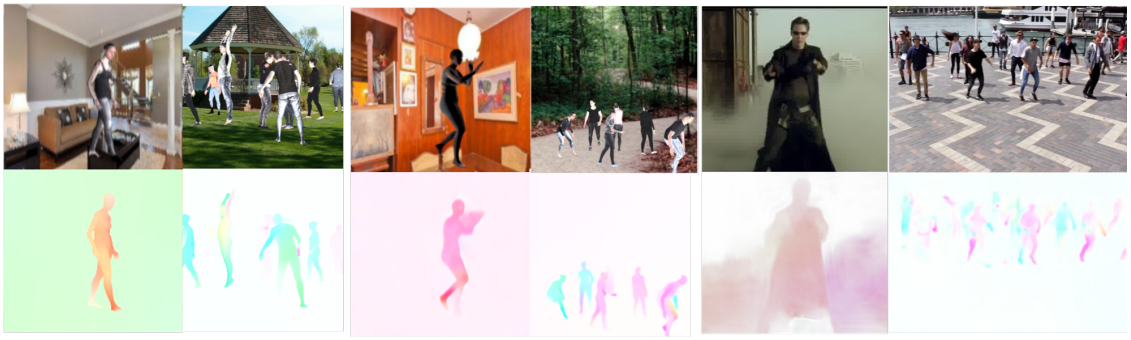
A significant fraction of videos on the Internet contain people moving [44] and the literature suggests that optical flow plays an important role in understanding human action [68, 125]. Action recognition datasets [74, 125] contain human motion as a major component. The 2D motion of humans in video, or *human optical flow*, is an important feature that provides a building block for systems that can understand and interact with humans. Human optical flow is useful for various applications including analyzing pedestrians in road sequences, motion-controlled gaming, activity recognition, human pose estimation etc.

Despite this, optical flow has previously been treated as a generic, low-level vision problem. Given the importance of people, and the value of optical flow in understanding them, we develop a flow algorithm that is specifically tailored to humans and their motion. Such motions are non-trivial since humans are complex, articulated objects that vary in shape, size and appearance. They move quickly, adopt a wide range of poses, and self-occlude or occlude in multi-person scenarios.

Our goal is to obtain more accurate 2D motion estimates for human bodies by training a flow algorithm specifically for human movement. To do so, we create a large and realistic dataset of humans moving in virtual worlds with ground-truth optical flow (Figure 4.1(a)), called the Human Optical Flow dataset. This is comprised of two parts; the Single-Human Optical Flow dataset (SHOF), where the image sequences contain only one person in motion and the Multi-Human Optical Flow dataset (MHOF) where images contain multiple people involving significant occlusion between them.

We evaluate the generalization of both our datasets by using them to train optical flow networks. We train SPyNet [103] using SHOF dataset to obtain a baseline and then train SPyNet as well as PWC-Net [130] on the MHOF dataset. In both cases, we see a significant improvement in human optical flow estimation. The networks trained obtain state-of-the-art optical flow on the test sequences of this dataset (Figure 4.1(b)). The trained networks also generalize to real video sequences (Figure 4.1(c)).

Several datasets and benchmarks [10, 22, 43] have been established to drive the progress in optical flow. We argue that these datasets are insufficient for the task of human motion estimation and, despite its importance, no attention has been paid to datasets and algorithms for human optical flow. One of the main reasons is that dense human motion



(a) Our dataset (b) Results on synthetic scenes (c) Results on real scenes

Figure 4.1: (a) We simulate human motion in virtual worlds creating an extensive dataset with images (top row) and flow fields (bottom row); color coding from [10]. (b) We train an existing deep network for human motion estimation and show that it performs better when trained on our dataset and (c) generalizes to human motions in real world scenes. Columns show single-person and multi-person cases alternately.

is extremely difficult to capture accurately in real scenes. Without ground truth, there has been little work focused specifically on estimating human optical flow. To advance research on this problem, we provide a dataset tailored to human optical flow.

A key observation is that recent work has shown that optical flow methods trained on synthetic data [33, 61, 103] generalize relatively well to real data. Additionally, these methods obtain state-of-the-art results with increased realism of the training data [41, 86]. This motivates our effort to create a dataset designed for human motion. To that end, we use the SMPL body model [81] to generate about a hundred thousand different human shapes. We place them on random indoor backgrounds and simulate human activities like running, walking, dancing etc. using motion capture data [80]. We extend this by using the more holistic SMPL+H [113] body and hands model to include finger motion, and render multiple humans in each image. Thus, we create a large virtual dataset that captures the statistics of natural human motion in both single and multi-person scenarios. We then train optical flow networks on our datasets and evaluate their performance for estimating human motion. While our dataset can be used to train any flow method, we choose SpyNet and PWC-Net because they are compact and computationally efficient. In summary, our major contributions are:

1. We provide the Single-Human Optical Flow dataset (SHOF) of human bodies in motion with realistic textures and backgrounds, having $146,020$ frame pairs for single-person scenarios.

2. We extend this by providing the Multi-Human Optical Flow dataset (MHOF), with $111,312$ frame pairs of multiple human bodies in motion, with improved textures

and realistic visual occlusions, but without (self-)collisions or intersection of body meshes. These two datasets together comprise the *Human Optical Flow dataset*.

3. We show that networks trained using our dataset outperform previous optical flow methods by 30% on the single-person (SHOF) and 10-20 % on the multi-person (MHOF) Human Optical Flow dataset, and they generalize to real world scenes.

4. We provide the data, code, and trained models at `http://humanflow.is.tue.mpg.de` for research purposes.

4.1 Human Optical Flow Dataset

Our approach generates a realistic dataset of synthetic human motions by simulating them against different realistic backgrounds. We use parametric models [81, 113] to generate synthetic humans with a wide variety of different human shapes and appearances. We then use the Cycles rendering engine of Blender[1] to generate realistic synthetic image frames and optical flow, to create the Human Optical Flow dataset. This is comprised of two parts. We first create the Single-Human Optical Flow (SHOF) dataset using the body-only SMPL model [81] in images containing a single synthetic human. However, image statistics are different for the single- and multi-person case, as multiple people tend to occlude each other in complicated ways. For this reason we also create the Multi-Human Optical Flow (MHOF) dataset to better capture this realistic interaction. To make images even more realistic, we replace SMPL [81] with the SMPL+H [113] that models the body together with fingers, to have richer motion variation. In the following section, we describe the components of our rendering pipeline, shown in Figure 4.2. For clarity, for each component we first describe the setup for the SHOF dataset, followed by the changes for the MHOF dataset.

Body Model and Rendering Engine. The main challenge in the generation of realistic human optical flow is modeling realistic articulated motions. We use the SMPL [81] and SMPL+H [113] models, parameterized by pose and shape parameters to change the body posture and identity, as shown in Figure 4.2.

The models also contain a UV appearance map that allows us to change the skin tone, face features and clothing texture of the resulting virtual humans. A key component of Blender in this project is its *Vector pass*. This render pass is typically used for producing motion blur, and it produces the motion in image space for every pixel; i.e. ground truth optical flow. We are mainly interested in the result of this pass, together with the color rendering of the textured bodies.

[1] `https://www.blender.org`

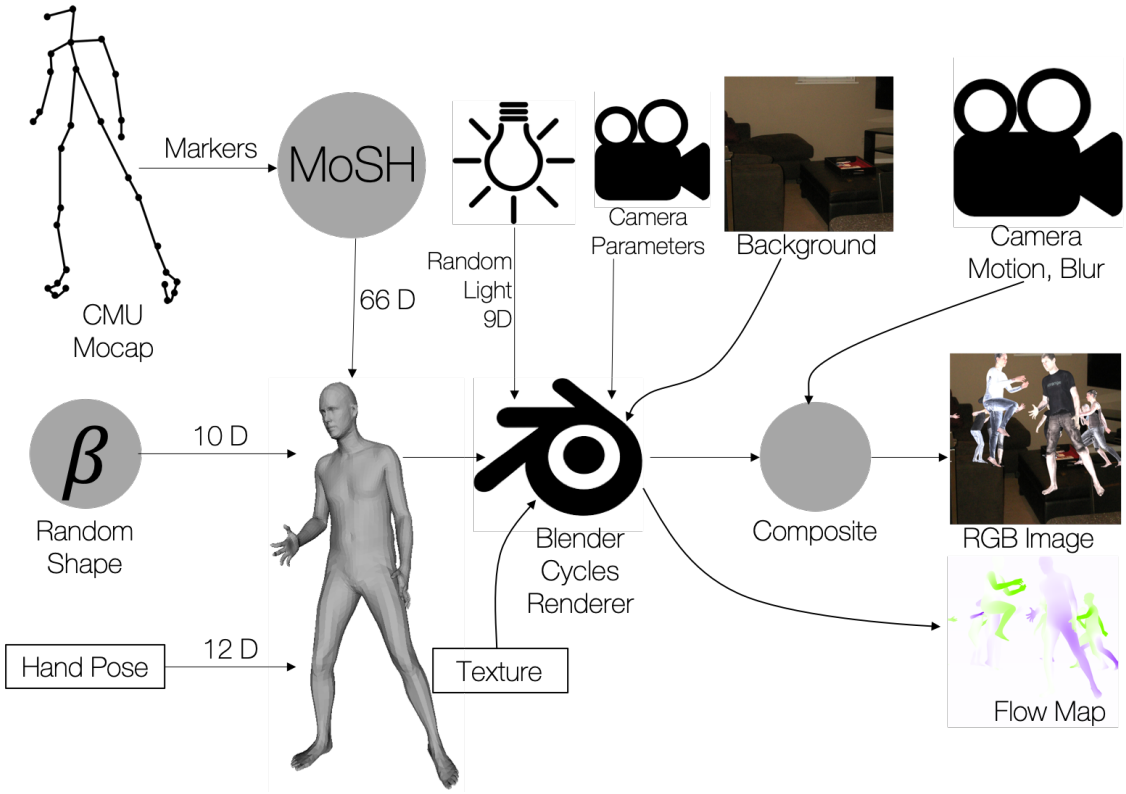


Figure 4.2: Pipeline for generating the RGB frames and ground truth optical flow for the Human Optical Flow dataset. Hand poses are only used for the Multi-Human Optical Flow dataset (MHOF).

Body Poses. In order to obtain a varied set of poses, we use motions from rich MoCap datasets in the literature, by converting 3D marker positions to SMPL shape and pose parameters using MoSh [80]. For the SHOF dataset, we employ the Human3.6M dataset [62]. Human3.6M contains five subjects for training (S1, S5, S6, S7, S8) and two for testing (S9, S11). Each subject performs 15 actions twice, resulting in $1,559,985$ frames for training and $55,727$ for testing. These sequences are subsampled at a rate of 16 fps, resulting in $97,499$ training and $34,420$ testing poses from Human3.6M.

For the MHOF dataset, we choose different MoCap datasets to increase motion variation, namely the CMU Mocap and HumanEva [121] MoCap datasets. The CMU MoCap dataset consists of $2,605$ sequences of 23 high-level action categories. Here we use more than $2,000$ of these sequences. From the HumanEva dataset we use more than 10 sequences performing actions from 6 different action categories. To reduce redundant poses, sequences are subsampled to 12 fps resulting in $321,873$ poses. As a result the final MHOF dataset has $254,211$ poses for training, $32,670$ for validation and $34,992$ for testing.

Hand Poses. Traditional MoCap systems and datasets [121] record the motion of full bodies, and avoid the tedious capture of detailed finger motion. However, in natural settings people jointly use their body, hands and fingers to communicate social cues and to interact with the physical world. In this direction, for the MHOF dataset, we use the SMPL+H model [113] and augment the body-only MoCap datasets, described above, with finger motion. Instead of using random finger poses that would generate unrealistic optical flow, we employ the "embodied hands" dataset [113] and sample continuous finger motion to generate realistic optical flow. We use 43 sequences of hand motion with $37,232$ frames recorded at 60 Hz by [113]. Similarly to body MoCap, we subsample hand MoCap to 12 fps to reduce overlapping poses without sacrificing variability.

Body Shapes. To maximize the variance in the SHOF dataset, for each subsequence of 20 frames we use a random body shape drawn from a gender-specific uniform distribution of SMPL shape parameters bounded by $[-3,3]$ standard deviations for each shape coefficient according to the shape distribution in CAESAR [111]. Using a parametric distribution ensures that each subsequence of frames has a unique body shape. A uniform distribution has more extreme shapes than the Gaussian distribution inferred originally from CAESAR, while avoiding unlikely shapes by strictly bounding the coefficients. For the MHOF dataset, we use similar sampling with only small differences to avoid self-intersecting meshes that would result in generation of inaccurate optical flow. For this, we use shorter subsequences of 10 frames to reduce intersections between different virtual humans. We further bound sampling of shape parameters to the narrower range of $[-2.7,2.7]$ standard deviations for each shape coefficient, since unlikely shapes are more prone to mesh self-intersections after re-targeting motion from subjects with different shape.

Scene Illumination. Optical flow estimation should be robust to different scene illumination. In order to achieve this invariance, we illuminate the bodies with Spherical Harmonics lighting [48]. Spherical Harmonics define basis vectors for light directions that are scaled and linearly combined. This compact parameterization is particularly useful for randomizing the scene light. The linear coefficients are randomly sampled with a slight bias towards natural illumination. The coefficients are uniformly sampled between -0.7 and 0.7, apart from the ambient illumination (which is strictly positive and a minimum of 0.3) and the vertical illumination (which is strictly negative to prevent illumination from below).

Body Texture. To provide a varied set of appearances to the bodies in the scene, we use textures from two different sources. A wide variety of human skin tones is extracted from the CAESAR dataset [111]. Given SMPL registrations to CAESAR scans, the original per-vertex color in the CAESAR dataset is transferred into the SMPL texture map. Since fiducial markers were placed on the bodies of CAESAR subjects, we remove them

from the textures and inpaint them to produce a natural texture. In total we use 166 CAESAR textures that are of good quality. The main drawback of CAESAR scans is their homogeneity in terms of outfit, since all of the subjects wore grey shorts and the women wore sports bras. In order to increase the clothing variety, we also use textures extracted from our 3D scans (referred as non-CAESAR in the following), to which we register SMPL with 4Cap [100]. A total of 772 textures from 7 different subjects with different clothes were captured. All textures were anonymized by replacing the face by the average face in CAESAR, after correcting it to match the skin tone of the texture. Textures are grouped according to the gender, which is randomly selected for each virtual human. For the SHOF dataset, the textures were split in training and testing sets with a $70/30$ ratio, while each texture dataset is sampled with a 50% change. For the MHOF dataset, we introduce more refined splitting with a $80/10/10$ ratio for the train, validation and test sets. Moreover, since we introduce also finger motion, we want to favour sampling non-CAESAR textures, due to the bad quality of CAESAR texture maps for the finger region. Thus each texture is sampled with equal probability.

Hand Texture. As described above, creating texture maps depends on model registration in 3D scans which transfers color from the scan to the texture map of the model. However, certain parts of the body are hard to be scanned due to occlusions or measurement limitations, and are frequently missing in 3D scans. As a result, texture maps are particularly noisy for these regions or might even have holes. Hands and fingers are a very representative case. Since texture is important for optical flow, we augment the body texture maps of the previous section to improve hand regions. For this we follow a divide and conquer approach; we first capture hand-only scans with a 3dMD scanner similar to the one in [113], then we create hand-only textures using the MANO model and the registration method of [113], getting 176 high resolution textures from 20 subjects, and finally we use these hand-only textures to replace the problematic hand regions in the full-body texture maps. For this, we need to find the best matching hand-only texture for every body texture. Therefore, we convert all texture maps in HSV space, and compute the mean HSV value for each texture map from standard sampling regions; for full body textures we sample face regions without facial hair, i.e. the cheekbones and the area between the eyebrows, while for hand-only textures we sample the center of the outer palm. Subsequently, for each body texture map we find the closest hand-only texture map in HSV space, and shift the values of the latter by the HSV difference, so that the the hand skin tone becomes more similar to the facial skin tone. Finally, this improved hand-only texture map is used to replace the pixels in the hand-region of the full body texture map.

Background Texture. The other crucial component of image appearance is the background. Since human motion rarely happens in front of clean and easily segmentable scenes, realistic backgrounds should be included in the synthetic scenes. For the SHOF

dataset, we found that using random indoor images from the LSUN dataset [158] as background provided a good compromise between simplicity and the complex task of generating varied full 3D environments. We use $417,597$ images from the LSUN categories kitchen, living room, bedroom and dining room. The background images are placed as billboards 9 meters from the camera, and are not affected by the spherical harmonics lighting. To increase variability in background appearance, for the MHOF dataset, we additionally employ the Sun397 dataset [154], that contains images for 397 highly variable indoor and outdoor scenes. For quality reasons, we reject all images with resolution smaller than 512×512 px, while we also reject images that contain humans using Mask-RCNN [1, 55], as we do not have ground truth for these humans. As a result, we use $30,222$ images, split in $24,178$ for the training set and $3,022$ for each of the validation and test sets. We further increase the distance between the camera and background billboard to 12 meters, to increase the space in which virtual humans can move without colliding frequently to each other, while still being close enough for visual occlusions.

Segmentation Mask. We use Blender's material pass and store the ground-truth body part segmentation for each frame. Although the body part segmentation for both SMPL and SMPL+H models is similar, SMPL models the palm and fingers as one part, while SMPL+H has a different part segment for each finger bone. Figure 4.3 shows an example body part segmentation for SMPL+H. These segmentation masks allow us to perform a per body-part evaluation of our optical flow estimation.

(Self-)Collision. The MHOF dataset contains multiple virtual humans moving differently, so there are high chances of collisions and penetrations. This is undesirable, not only because penetrations are physically implausible and unrealistic, but also because the generated ground truth optical flow might have artifacts. For these reasons, we employ a collision detection method to avoid intersections and penetrations, by re-sampling the parameters that lead to such. Instead of using simple bounding boxes for rough collision detection, we draw inspiration from [136] and perform accurate and efficient collision detection on the triangle level using bounding volume hierarchies (BVH) [134]. This level of detailed detection allows for challenging occlusions with small distances between virtual humans, that can commonly be observed for realistic interactions between real humans. This method is useful not only for inter-person collision detection, but also for self-intersections. This is especially useful for our scenarios, as re-targeting body and hand motion to people of different shapes might result in unrealistic self-penetrations. The method is applicable out of the box, with the only exception the fact that we exclude checks of neighboring body parts that are always or frequently in contact, e.g. upper and lower arm, or the two thighs.

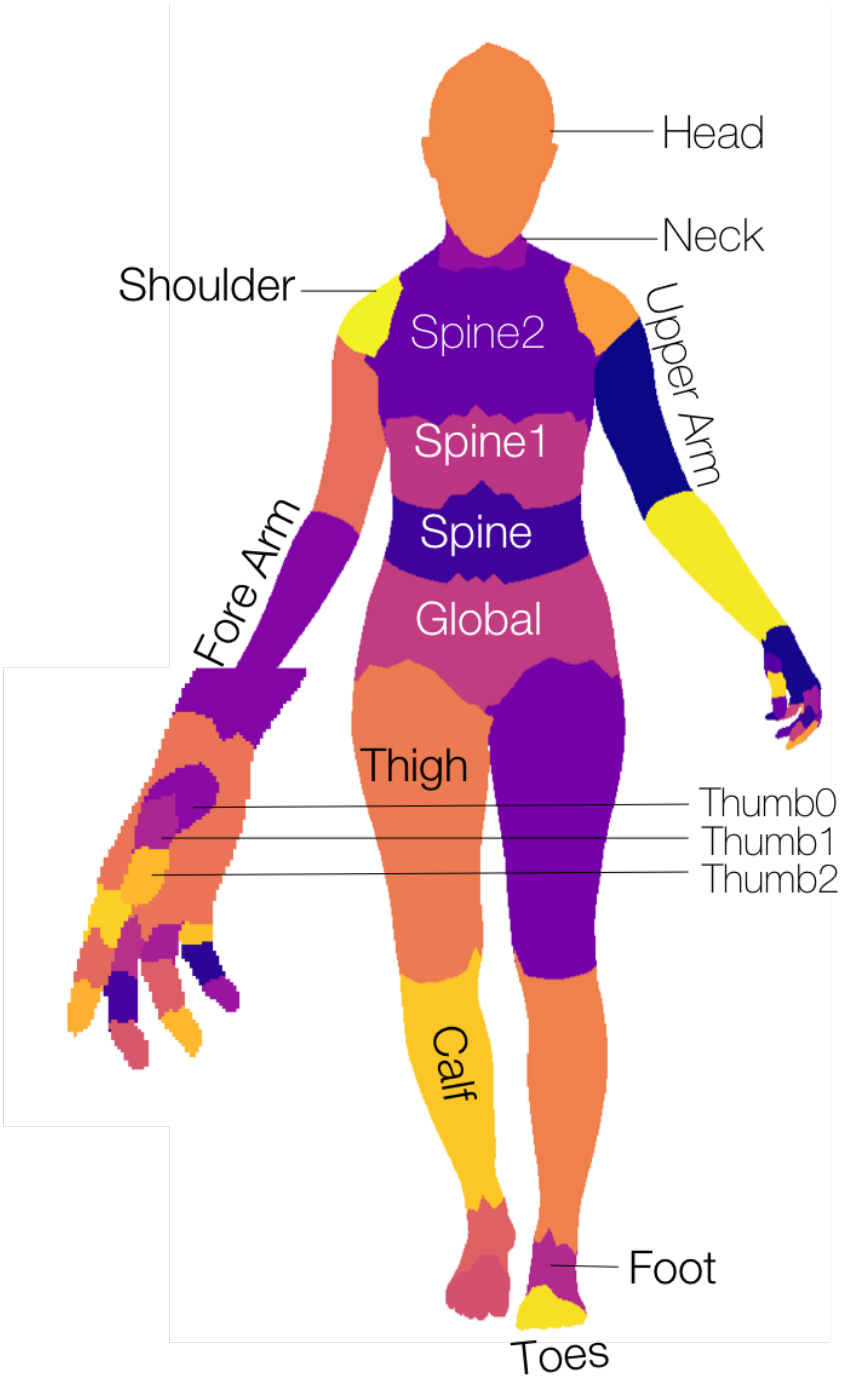


Figure 4.3: Body part segmentation for the SMPL+H model. Symmetrical body parts are labeled only once. Finger joints follow the same naming convention as shown for the thumb. (Best viewed in color)

Increasing Image Realism. One of the main differences between synthetic and real images are the imperfections existing in the latter. Fully sharp images rendered with perfectly static virtual cameras do not represent well the images captured in real situations. In order to increase realism, we introduced three types of image imperfections. First, in 30% of the generated sequences we introduced camera motion between frames. This motion perturbs the location of the camera with Gaussian noise of 1 cm standard deviation between frames and rotation noise of 0.2 degrees standard deviation per dimension in an Euler angle representation. Second, motion blur was added to the scene. The motion blur was implemented with the *Vector Blur Node* in Blender, and integrated over 2 frames sampled with 64 steps between the beginning and end point of the motion. Finally, general image blur was added to 30% of the images, as Gaussian blur with a standard deviation of 1 pixel. The camera motion parameters in the MHOF dataset are kept fixed within a subsequence to allow for more consistent optical flow.

Scene Compositing. For animating virtual humans, each MoCap sequence is selected at least once. To increase variability, each sequence is split into subsequences. For the first frame of each subsequence we sample a different shape, body and background texture, lights, blurring and camera motion parameters, while we also re-position virtual humans on the horizontal plane and introduce a random rotation around the z-axis for variability in the motion direction. For the SHOF dataset, we use subsequences of 20 frames, and at the beginning of each one, the single virtual human is re-positioned in the scene such that the pelvis is projected onto the image center. For the MHOF dataset, we increase the variability with smaller subsequences of 10 frames and introduce more challenging visual occlusions by uniformly sampling the number of virtual humans in the range $[4,8]$. To make sure that each sequence is selected at least once, we deterministically choose it for the first virtual human, while for the rest we sample randomly to make the virtual humans move differently. In this direction, we sample MoCap sequences S_j with $|S_j|$ frames with a probability of $p_j = \frac{|S_j|}{\sum_{j=1}^{|S|}|S_j|}$. Since the chosen sequences might have a different length, we further randomly sample for each one the frame ID for the starting frame, to encourage use of the whole sequences. In contrast to the SHOF dataset, for the MHOF dataset, the virtual humans are not re-positioned at the center, as they would all collide. Instead, they are placed at random locations on the horizontal plane within camera visibility, making sure there are no collisions with other virtual humans or the background plane during the whole subsequence.

Dataset Details. In comparison with other optical flow datasets, our SHOF dataset is larger by an order of magnitude (see Table 4.1). The SHOF dataset contains $135,153$ training frames and $10,867$ test frames with optical flow ground truth. Our MHOF dataset has $86,259$ training, $13,236$ test and $11,817$ validation frames. This dataset contains optical flow ground truth for the much more challenging scenario of multiple moving people moving in the scene. For the SHOF dataset we keep the resolution small

Dataset	# Train Frames	# Test Frames	Resolution
MPI Sintel [22]	1,064	564	1024×436
KITTI 2012 [43]	194	195	1226×370
KITTI 2015 [89]	200	200	1242×375
Virtual Kitti [41]	21,260	–	1242×375
Flying Chairs[33]	22,232	640	512×384
Flying Things [86]	21,818	4,248	960×540
Monkaa [86]	8,591	–	960×540
Driving [86]	4,392	–	960×540
SHOF (ours)	135,153	10,867	256×256
MHOF (ours)	86,259	13,236	640×640

Table 4.1: Comparison of the Human Optical Flow datasets, namely the Single-Human Optical Flow (SHOF) and the Multi-Human Optical Flow (MHOF) dataset, with previous optical flow datasets

at 256×256 px to facilitate easy deployment for training neural networks. This also speeds up the rendering process in Blender for generating large amounts of data. We show the comparisons of processing time of different models on the single-person data in Table 4.2(a). For the MHOF dataset, we increase the resolution to 640×640 px to be able to reason about optical flow even in small body parts like fingers, using SMPL+H. Our data is extensive, containing a wide variety of human shapes, poses, actions and virtual backgrounds to support deep learning systems.

4.2 Learning

Our neural network derives from SPyNet [103] as described in Chapter3, which employs different convnets at different levels of an image pyramid. In SPyNet, these convnets are trained independently and sequentially to estimate optical flow. In SPyNet, fixed downsampling and upsampling operators d, u are used. In contrast, we introduce learnable convolutional layers d, u, for downsampling and upsampling between the pyramid levels. A differential warping operator, w [63] facilitates our model to be fully differentiable and end-to-end trainable. Thus, we perform joint training of all the convnets at different pyramid levels rather than training them sequentially like SPyNet. We briefly describe the spatial pyramid structure and introduce our network and learning process below.

Our architecture consists of 4 pyramid levels. For simplicity, we show 3 of the 4 pyramid levels in Figure 4.4. Each level works on a particular resolution of the image. The top level works on the full resolution and the images are downsampled as we move to the bottom of the pyramid. Each level learns a convolutional layer d, to perform downsampling of images. Similarly, a convolution layer u, is learned for upsampling optical flow. At each level, we also learn a convnet G_k to predict optical flow residuals

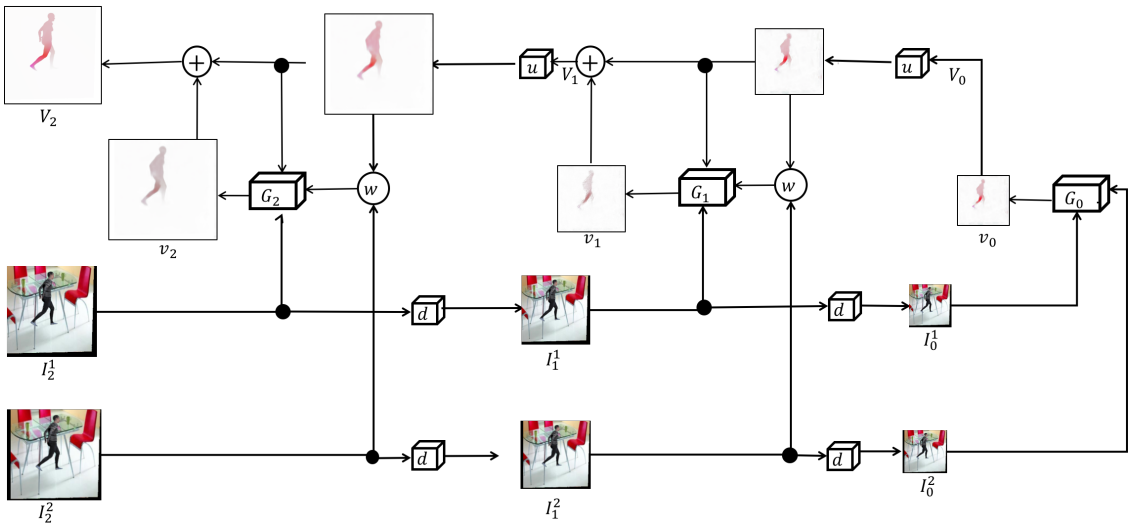


Figure 4.4: A Spatial Pyramid Network [103] for Optical Flow. At each pyramid level, network G_k predicts flow residuals v_k that get added up to produce full flow V_2. w is warping operator. u, d are learned convolutional layers that upsample flows and downsample images. The figure shows 3 pyramid levels for simplicity. Our implementation uses 4 levels.

v_k at that level. These flow residuals get added at each level to produce the full flow, V_K at the finest level of the pyramid.

Each convnet G_k takes a pair of images as inputs along with flow V_{k-1} obtained by upsampling the output of the previous level. The second frame is however warped using V_{k-1} and the triplet $\{I_k^1, w(I_k^2, V_{k-1}), V_{k-1}\}$ is fed as input to the convnet G_k. The structure of the convnets G_k is same as SPyNet. Each of the convnets G_k is a five layer network containing $\{32, 64, 32, 16, 2\}$ feature maps with 7x7 kernels. At each level, the downsampling layers d learn 3x3 convolutional kernels with 6 feature maps to operate on 6 channels of image pairs. Similarly, the upsampling layers u learn 4x4 convolutional kernels with 2 feature maps to operate on 2-channel flows. We use w to refer to a bilinear warping operator which is non-learnable. The general structure of spatial pyramids can be seen in [103]. We import the weights of our convnets G_i from the first four convnets $\{G_0, G_1, G_2, G_3\}$ of SPyNet pre-trained on the Flying Chairs dataset [33]. We use a differentiable warping operator, w [63]. We now construct our fully differentiable spatial pyramid architecture and train it end-to-end minimizing a flow end-point error (EPE).

Hyperparameters. We use Adam [71] to optimize our loss at a constant learning rate of 10^{-6}, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We use a batch size of 8 and run 4000 iterations per epoch. We train our model for 100 epochs on the Single-Human Optical Flow dataset.

Method	Average EPE	Time(s)
Zero	0.6611	-
FlowNet [33]	0.5846	0.080
PCA Flow [151]	0.3652	10.357
SPyNet [103]	0.2066	0.038
Epic Flow [110]	0.1940	1.863
LDOF [19]	0.1881	8.620
Flow Fields [9]	0.1709	4.204
SpyNet+SHOF	**0.1164**	**0.031**

Table 4.2: EPE comparisons and evaluation times of different optical flow methods on the SHOF dataset. Zero refers to the EPE when zero flow is always used for evaluation. Evaluation times are based on the SHOF dataset with 256×256 image resolution.

We use the Torch7[2] framework for our implementation and use four Nvidia K80 GPUs to train in parallel. It takes 1 day for our model to train.

Learning Multi Human Flow For learning on the Multi-Human Optical Flow dataset (MHOF), we train two different architectures, SpyNet [103] and PWC-Net [130]. We refer to the trained networks on the MHOF dataset as SpyNet+MHOF and PWC+MHOF. We follow the same training procedure as followed SpyNet and PWC-Net, and initialize the network weights to the trained SpyNet and PWC-Net before training on the MHOF dataset. We evaluate the results using end-point error on the whole image, as well as different parts of the human body and hands.

4.3 Experiments

We compare the average end-point errors (EPEs) of competing methods in Table 4.2 along with the time for evaluation. Human motion is complex and general optical flow methods fail to capture it. Our trained network SpyNet+SHOF outperforms previous methods, and SPyNet [103] in particular, in terms of average EPE on the SHOF dataset. This indicates that other optical flow networks can also improve their performance on human motion using our dataset. We show visual comparisons in Figure 4.5. Similarly, the networks SPyNet+MHOF, PWC+MHOF trained on Multi-Human Optical Flow datasets outperform previous methods on average and on each of the different body parts as shown in Table 4.3. The visual comparisons are shown in Figure 4.6.

We observe that FlowNetS [33] shows poor generalization on our dataset. Since the results of FlowNetS are very close to the zero flow baseline (Tables 4.2, 4.3), we cross-verify by evaluating FlowNetS on a mixture of Flying Chairs [33] and Human Optical

[2] `http://torch.ch`

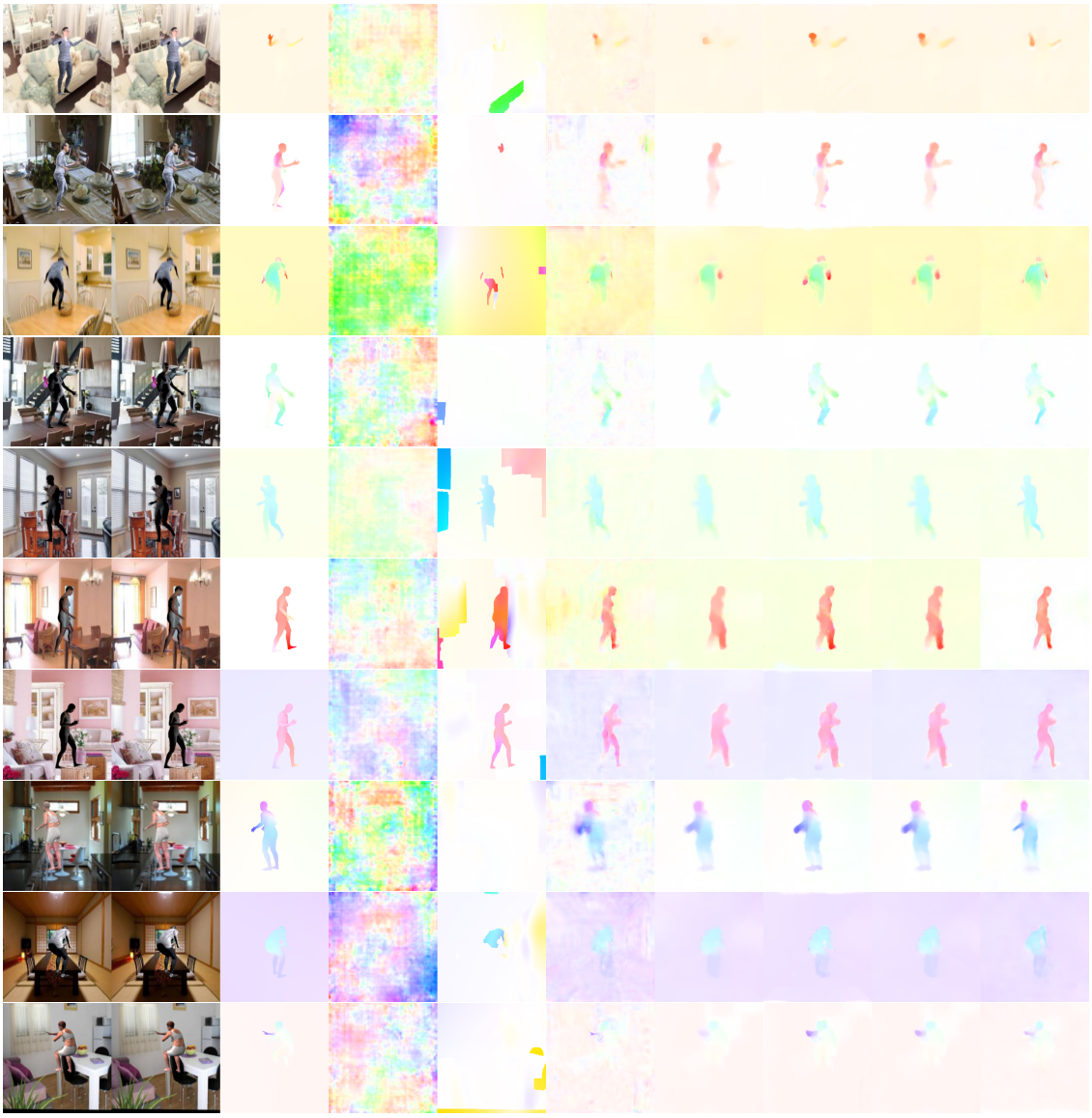


Figure 4.5: Visual comparison of optical flow estimates using different methods on the Single-Human Optical Flow (SHOF) test set. From left to right, we show Frame 1, Frame 2, Ground Truth flow, results on FlowNetS [33], PCA-Layers [151], SPyNet [103], EpicFlow [110], LDOF [19], FlowFields [9] and SpyNet+SHOF (ours).

Parts	SpyNet	PWC	FlowNet	FlowNet2	PWC +MHOF	SpyNet +MHOF
Average	0.448	0.411	0.882	0.346	0.295	0.409
Global	1.772	1.786	2.532	1.688	1.337	1.578
Head	2.128	2.289	3.283	1.942	1.568	1.804
lCalf	2.880	3.056	3.332	2.542	1.779	2.451
lFoot	4.198	4.583	4.334	3.701	2.623	3.750
lForeArm	4.695	4.564	5.011	4.235	3.424	4.147
lHand	6.441	6.424	6.666	5.695	4.651	5.719
lShoulder	1.923	2.073	2.773	1.966	1.599	1.757
lThigh	2.067	2.144	2.729	1.974	1.488	1.814
lToes	4.325	4.666	4.481	4.094	3.228	4.111
lUpperArm	2.879	2.887	3.561	2.816	2.219	2.524
Neck	1.732	1.734	2.621	1.629	1.344	1.478
rCalf	3.006	3.216	3.479	2.648	1.886	2.592
rFoot	4.336	4.753	4.471	3.847	2.759	3.876
rForeArm	4.835	4.636	5.140	4.286	3.473	4.223
rHand	6.479	6.326	6.674	5.581	4.561	5.536
rShoulder	2.013	2.155	2.929	2.066	1.659	1.850
rThigh	2.108	2.230	2.827	2.048	1.551	1.870
rToes	4.515	4.923	4.667	4.261	3.348	4.274
rUpperArm	3.035	3.005	3.729	2.883	2.280	2.643
Spine	1.682	1.690	2.403	1.621	1.313	1.530
Spine1	1.760	1.785	2.473	1.704	1.406	1.618
Spine2	1.754	1.824	2.546	1.752	1.425	1.630

Table 4.3: Comparison using end point error (EPE) on the Multi-Human Optical Flow dataset. We show average EPE and body part specific EPE, where part labels follow from Figure 4.3.

Flow dataset and observe that the flow outputs on human optical flow dataset is quite random (see Figure 4.5). The main reason is that our dataset contains a significant amount of small motions and it is known that FlowNetS does not perform very well on small motions. SPyNet [103] performs quite well and is able to generalize to body motions. The results however look noisy in many cases.

Our dataset employs a layered structure where a human is placed against a background. As such, layered methods like PCA-layers [151] perform very well on a few images (row 6 in Figure 4.5) where they are able to segment a person from the background. However, in most cases, they do not obtain good segmentation into layers.

Previous state-of-the-art methods like LDOF [19], Epic-Flow[110] and FlowFields [9] perform better than others. They get a good overall shape, and smooth backgrounds.

Figure 4.6: Visual comparison of optical flow estimates using different methods on the Multi-Human Optical Flow (MHOF) test set. From left to right, we show Frame 1, Ground Truth flow, results on FlowNet2 [61], LDOF [19], PCA-Layers [151], EpicFlow [110], SPyNet [103], SpyNet+MHOF (ours), PWC-Net [130] and PWC+ MHOF (ours).

However, their estimation is quite blurred. They tend to miss the sharp edges that are typical of human hands and legs. They are also significantly slower.

In contrast, our network outperforms the state-of-the-art optical flow methods by 30% on the Single-Human Optical Flow dataset. Our qualitative results show that our method can capture sharp details like hands and legs of the person. Experiments on the MHOF dataset show an improvement of about 10-20% using the same networks, SPyNet and PWC-Net (see Table 4.3). Since, our test data is comparatively large, we evaluate only the fastest optical methods on our dataset.

Real Scenes. We show a visual comparison of results on real-world scenes of people in motion. We collect these scenes by cropping people from real world videos as shown in Figure 4.7. We use DPM [37] for detecting people and compute bounding box regions in two frames using the ground truth of the MOT16 dataset [90].

Figure 4.7: We use a person detector to crop out people from real-world scenes (left) and use our model to compute optical flow on the cropped section (right).

We visually compare our results with other popular optical flow methods on real world scenes using models trained on the SHOF (Figure 4.8) and the MHOF (Figure 4.9) datasets. The performance of PCA-Layers [151] is highly dependent on its ability to segment. Hence, we see only a few cases where it looks visually correct. SPyNet [103] gets the overall shape but the results look noisy in certain image parts. While LDOF [19], EpicFlow [110] and FlowFields [9] generally perform well, they often find it difficult to resolve the legs, hands and head of the person. The results from models trained on Human Optical Flow datasets look appealing especially while resolving the overall human shape, and various parts like legs, hands and the human head. These models also performs very well under occlusions when the full body is not visible, and under occlusions with multiple people.

Timing Evaluation. Although some of the methods do quite well on our benchmark, they tend to be slow due to their complex nature. As such, they are not likely to be used in real time or embedded applications. We show timing comparisons on a pair of frames in Table 4.2. We show a simple way of learning with human optical flow data using fast optical flow networks. Our model takes 31 ms for inference on NVIDIA TitanX. As such it can run in real time at 32 fps.

4.4 Conclusion

In summary, we created an extensive dataset containing images of realistic human shapes in motion together with groundtruth optical flow. The realism and extent of this dataset, together with an end-to-end trained system, allows our networks to outperform previous state-of-the-art optical flow methods in our new human-specific dataset. Furthermore,

Figure 4.8: Visual comparison of optical flow estimates using different methods on real scenes. From left to right, we show Frame 1, Frame 2, results on PCA-Layers [151], and SPyNet [103], EpicFlow [110], LDOF [19], FlowFields [9] and SPyNet+SHOF (ours).

Figure 4.9: Multi-Human Optical Flow visuals on real images. From left to right, we show Frame 1, results on FlowNet2 [61], FlowNet [33], LDOF [19], PCA-Layers [151], EpicFlow [110], SPyNet [103], SPyNet+MHOF(ours), PWC-Net [130] and PWC+MHOF (ours).

we show that our method generalizes to human motion in real world scenes for optical flow computation. Our method is compact and runs in real time making it highly suitable for phones and embedded devices.

The dataset and our focus on human optical flow opens up a number of research directions in human motion understanding and optical flow computation. We would like to extend our dataset by modeling more diverse clothing and outdoor scenarios. A direction of potentially high impact for this work is to integrate it in end-to-end systems for action recognition, which typically take pre-computed optical flow as input. The real-time nature of the method could support motion-based interfaces, potentially even on devices like cell phones with limited computing power. The data, model, and training code is available, enabling researchers to apply this to problems involving human motion.

Parts	SpyNet	PWC	FlowNet	FlowNet2	PWC +MHOF	SpyNet +MHOF
lIndex0	6.984	7.125	7.176	6.393	5.309	6.347
lIndex1	7.132	7.315	7.271	6.758	5.741	6.617
lIndex2	7.014	7.223	7.086	6.905	6.028	6.641
lMiddle0	7.021	7.178	7.268	6.342	5.215	6.337
lMiddle1	7.157	7.388	7.356	6.679	5.659	6.570
lMiddle2	6.867	7.123	6.981	6.769	5.865	6.449
lPinky0	6.370	6.559	6.617	5.927	4.957	5.740
lPinky1	6.595	6.828	6.766	6.258	5.379	6.033
lPinky2	6.393	6.635	6.478	6.282	5.486	5.990
lRing0	6.901	7.021	7.152	6.117	5.047	6.159
lRing1	6.941	7.135	7.137	6.374	5.383	6.335
lRing2	6.745	6.987	6.850	6.554	5.705	6.309
lThumb0	6.205	6.167	6.264	5.676	4.821	5.581
lThumb1	6.348	6.418	6.453	6.001	5.099	5.802
lThumb2	6.495	6.613	6.574	6.297	5.460	6.066
rIndex0	6.896	7.002	7.128	6.165	5.186	6.103
rIndex1	7.016	7.220	7.221	6.515	5.615	6.394
rIndex2	6.778	7.029	6.916	6.623	5.818	6.353
rMiddle0	7.105	7.183	7.331	6.249	5.223	6.231
rMiddle1	7.026	7.215	7.224	6.459	5.512	6.350
rMiddle2	6.634	6.871	6.772	6.496	5.633	6.158
rPinky0	6.443	6.469	6.623	6.034	4.969	5.641
rPinky1	6.392	6.491	6.548	6.159	5.162	5.706
rPinky2	6.148	6.265	6.260	6.071	5.143	5.611
rRing0	6.931	6.921	7.161	6.071	5.006	5.996
rRing1	6.716	6.803	6.915	6.062	5.123	5.961
rRing2	6.552	6.702	6.677	6.329	5.438	6.017
rThumb0	5.930	5.959	6.173	5.452	4.609	5.325
rThumb1	6.196	6.241	6.411	5.678	4.877	5.581
rThumb2	6.326	6.418	6.478	5.935	5.210	5.808

Table 4.4: Comparison using End Point Error (EPE) on the Multi-Human Optical Flow dataset. We show average EPE on each of the fingers that follow from Figure 4.3. Note that EPE on the hand parts is larger that other body parts (Table 4.3) which is critical to problems like hand-pose estimation.

Chapter 5

Competitive Collaboration

In the previous chapters, we used supervised learning to train optical flow networks. However, we did not model any explicit constraints that account for the structure of the scene. One can use depth to model the structure of the scene. In a video, depth and camera motion can account for scene structure, and optical flow can be used for estimating independent object motion. Therefore, integrating geometry with motion requires solving multiple problems which requires training multiple networks.

To facilitates the coordinated training of multiple specialized neural networks to solve complex problems, we introduce *Competitive Collaboration (CC)*. This framework facilitates neural networks learn to collaborate and compete in alternating cycles, thereby achieving specific goals. We formalize Competitive Collaboration as a three-player game, consisting of two competitors and a moderator, where the moderator takes the role of a critic and two competitors collaborate to train the moderator. The idea of collaboration can also be seen as neural expectation maximization [49] where one model is trained to distribute data to other models. For unsupervised learning, these ideas have been mainly used to model the data distribution [49] and have not been applied to unsupervised training of regression or classification problems.

5.1 Competitive Collaboration

Competitive Collaboration is formulated as a three-player game consisting of two players competing for a resource that is regulated by a moderator as illustrated in Figure 5.1. Consider an unlabeled training dataset $\mathcal{D} = \{\mathcal{D}_i : i \in \mathbb{N}\}$, which can be partitioned into two disjoint sets. Two players $\{R, F\}$ compete to obtain this data as a resource, and each player tries to partition $\mathcal{D}$ to minimize its loss. The partition is regulated by the moderator's output $m = M(\mathcal{D}_i), m \in [0,1]^{\Omega}$, and Ω is the output domain of the competitors. The competing players minimize their loss function L_R, L_F respectively such that each player optimizes for itself but not for the group. To resolve this problem, our training cycle consists of two phases. In the first phase, we train the competitors by fixing the moderator network M and minimizing

$$E_1 = \sum_i \sum_{\Omega} m \cdot L_R(R(\mathcal{D}_i)) + (1-m) \cdot L_F(F(\mathcal{D}_i)), \tag{5.1}$$

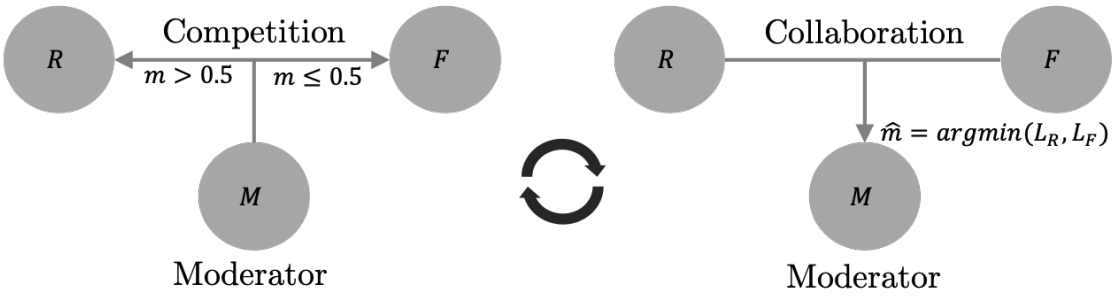


Figure 5.1: Training cycle of Competitive Collaboration: The moderator M drives two competitors $\{R,F\}$ (first phase, left). Later, the competitors collaborate to train the moderator to ensure fair competition in the next iteration (second phase, right).

where $\cdot$ is the elementwise product. However, the moderator M also needs to be trained. This happens in the second phase of the training cycle. The competitors $\{R,F\}$ form a consensus and train the moderator M such that it correctly distributes the data in the next phase of the training cycle. In the collaboration phase, we fix the competitors and train the moderator by minimizing

$$E_2 = E_1 + \sum_i \sum_\Omega L_M(\mathcal{D}_i, R, F), \tag{5.2}$$

where L_M is a loss that denotes a consensus between the competitors $\{R,F\}$. Competitive Collaboration can be applied to more general problems of training multiple task-specific networks. In the rest of the chapter, we will study the convergence properties of the framework and apply it to mixed domain learning on MNIST and SVHN digits.

5.2 Example

Competitive collaboration (CC) can be seen as a general learning framework for training multiple task-specific networks. To showcase this generality, we demonstrate CC on a mixed-domain classification problem.

Mixed Domain Classification

Digit classification is the task of classifying a given image I into one of the 10 digit classes $t \in \{0,1,2,..,9\}$. Two most widely used datasets for digit classification include images of the postal code digits, MNIST [77], and street view house numbers, SVHN [92]. For our setup, we take the samples from both of the datasets, and shuffle them together. This means that, although an image and a target (I_i, t_i) form a pair, there is no information if the digits came from MNIST or from SVHN.

We now train our model under the Competitive Collaboration framework given the mixed-domain dataset MNIST+SVHN, a mixture of MNIST and SVHN. The model consists of two networks R_x and F_x that compete with each other regulated by a moderator M_y which assigns training data to each of the competitors. Here, x denotes the combined weights of the two competitor networks (R,F) and y denotes the weights of the moderator network M. The networks are trained using an alternating optimization procedure consisting of two phases. In the competition phase, we train the competitors by fixing the moderator M and minimizing

$$E_1 = \sum_i m_i \cdot H(R_x(I_i), t_i) + (1 - m_i) \cdot H(F_x(I_i), t_i), \tag{5.3}$$

where $m_i = M_y(I_i) \in [0,1]$ is the output of the moderator and is the probability of assigning a sample to R_x. $H(R_x(I_i), t_i)$ is the cross entropy classification loss on the network R_x and a similar loss is applied on network F_x.

During the collaboration phase, we fix the competitors and train the moderator by minimizing

$$E_2 = E_1 + \lambda \sum_i \cdot \begin{cases} -\log(m_i + \varepsilon) & \text{if } L_{R_i} < L_{F_i}, \\ -\log(1 - m_i + \varepsilon) & \text{if } L_{R_i} \geq L_{F_i}, \end{cases} \tag{5.4}$$

where $L_{R_i} = H(R_x(I_i), t_i)$ is the cross entropy loss from network R_x and similarly $L_{F_i} = H(F_x(I_i), t_i)$. In addition to the above loss function E_2, we use an additional constraint on the moderator output that encourages the variance of m, $\sigma_m^2 = \Sigma_i (m_i - \bar{m})^2$ to be high, where $\bar{m}$ is the mean of m within a batch. This encourages the moderator to assign images to both the models, instead of always assigning them to a single model.

In an ideal case, we expect the moderator to correctly classify MNIST digits from SVHN digits. This would enable each of the competitors to specialize on either MNIST or SVHN, but not both. In such a case, the accuracy of the model under CC would be better than training a single network on the MNIST+SVHN mixture.

Implementation

For simplicity, we use a CNN with 2 convolutional layers followed by 2 fully-connected layers for both the digit classification networks (R,F) as well as the moderator network M. Each of the convolutional layers use a kernel size of 5x5 and 40 feature maps. The fully-connected layers have 40 neurons each.

Evaluation

We now compare the performance of the CC model on MNIST+SVHN mixture with training a single network on the same dataset. We measure our performance on the standard test set of MNIST and SVHN. We see that our performance is better on the

Networks	Training	MNIST	SVHN	MNIST+SVHN
R	Basic	1.34	11.88	8.96
R	CC	1.41	**11.55**	8.74
F	CC	**1.24**	11.75	8.84
R,F,M	CC	**1.24**	**11.55**	**8.70**

Table 5.1: Percentage classification errors. Training using Competitive Collaboration leads to expert models for MNIST (F) and SVHN (R).

	MNIST	SVHN
R	0%	100%
F	100%	0%

Table 5.2: Assignments of moderator to each of the competitors.

mixture dataset as well as individual datasets (see Table 5.1). As shown in Table 5.1, the network R specializes on SVHN digits and network F specializes on MNIST digits. By using the networks (R,F,M), we get the best results as M picks the specialized networks depending on the data sample.

We also examine the classification accuracy of the moderator on MNIST and SVHN digits. We observe that moderator can accurately classify the digits into either MNIST or SVHN without any labels (see Table 5.2). The moderator learns to assign 100% of MNIST digits to F and 100% of SVHN digits to R. In the next chapter, we will use this framework for unsupervised learning of depth, camera motion, optical flow and motion segmentation to support the notion that Competitive Collaboration can be generalized to other problems.

5.3 Convergence

Competitive Collaboration is an alternating optimization procedure. In the competition phase, we minimize E_1 with respect to x; in the collaboration phase we minimize $E_2 = E_1 + \lambda L_M$ with respect to y. One might rightfully worry about the convergence properties of such a procedure, where we optimize different objectives in the alternating steps.

It is important to note that—while E_1 and E_2 are different functions—they are in fact closely related. For example, they have the same minimizer with respect to the moderator output, namely assigning all the mass to the network with lower loss. Ideally, we would want to use E_1 as the objective function in both phases, but resort to using E_2 in the collaboration phase, since it has empirically proven to be more efficient in pushing the moderator towards this optimal choice.

Hence, while we are minimizing different objective functions in the competition and collaboration phases, they are closely related and have the same "goal". In the following, we formalize this mathematically by identifying general assumptions on how "similar" two functions have to be for such an alternating optimization procedure to converge. Roughly speaking, we need the gradients of the two objectives to form an acute angle and to be of similar scales. We will then discuss to what extent these assumptions are satisfied in the case of Competitive Collaboration. Proofs are provided in Appendix B.

General Convergence Theorem Assume we have two functions

$$f, g \colon \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} \tag{5.5}$$

and are performing alternating gradient descent updates of the form

$$x_{t+1} = x_t - \alpha \nabla_x f(x_t, y_t), \tag{5.6}$$
$$y_{t+1} = y_t - \beta \nabla_y g(x_{t+1}, y_t). \tag{5.7}$$

We consider the case of single alternating gradient descent for convenience in the analysis. With minor modifications, the following analysis also extends to the case of multiple gradient descent updates (or even exact minimization) in each of the alternating steps. The following Theorem formulates assumptions on f and g under which such an alternating optimization procedure converges to a first-order stationary point of f.

Theorem 1. *Assume f is lower-bounded and $x \mapsto \nabla_x f(x,y)$ is Lipschitz continuous with constant G_1 for every y and $y \mapsto \nabla_y g(x,y)$ is Lipschitz continuous with constant $G_2(x)$. Assume $\alpha \leq 2L_1^{-1}$. If there is a constant $B > 0$ such that*

$$\beta \langle \nabla_y f(x,y), \nabla_y g(x,y) \rangle \geq \frac{G_2(x)\beta^2}{2} \|\nabla_y g(x,y)\|^2 + B \|\nabla_y f(x,y)\|^2 \tag{5.8}$$

then (x_t, y_t) converges to a first-order stationary point of f.

Eq. (5.8) is a somewhat technical assumption that lower-bounds the inner product of the two gradients in terms of their norms and, thus, encodes that these gradients have to form an acute angle and be of similar scales.

Convergence of Competitive Collaboration We now discuss to what extent the assumptions for Theorem 1 are satisfied in the case of Competitive Collaboration. For the mathematical considerations to follow, we introduce a slightly more abstract notation for the objective functions of Competitive Collaboration. For a *single data point*, E_1 has the form

$$f(x,y) = M(y)L_R(x) + (1 - M(y))L_F(x), \tag{5.9}$$

where $M(y) \in [0,1]$ is a function of y (the weights of the moderator) and $L_R(x), L_F(x) > 0$ are functions of x (the weights of the two competing networks). The loss function E_2 reads

$$g(x,y) = f(x,y) + \lambda \cdot \begin{cases} -\log(M(y)+\varepsilon) & \text{if } L_R(x) < L_F(x), \\ -\log(1-M(y)+\varepsilon) & \text{if } L_R(x) \geq L_F(x). \end{cases} \tag{5.10}$$

The following Proposition shows that f and g satisfy the conditions of Theorem 1 under certain assumptions.

Proposition 1. *Let f and g be defined by Eq. (5.9) and Eq. (5.10), respectively. If $M(y)$, $L_R(x)$ and $L_F(x)$ are Lipschitz smooth, then f and g fulfill the assumptions of Theorem 1.*

The smoothness conditions on $M(y)$, $L_R(x)$, $L_F(x)$ are standard as they are, for example, needed to guarantee convergence of gradient descent for optimizing any of these objective functions individually.

This Proposition shows that the objectives for individual data points satisfy Theorem 1. In practice, however, we are concerned with multiple data points and objectives of the form

$$f(x,y) = \frac{1}{n}\sum_{i=1}^{n} f^{(i)}(x,y), \tag{5.11}$$

where

$$f^{(i)}(x,y) = M^{(i)}(y)L_R^{(i)}(x) + (1-M^{(i)}(y))L_F^{(i)}(x), \tag{5.12}$$

and

$$g(x,y) = \frac{1}{n}\sum_{i=1}^{n} g^{(i)}(x,y), \tag{5.13}$$

where

$$g^{(i)}(x,y) = f^{(i)}(x,y) + \lambda \cdot \begin{cases} -\log(M^{(i)}(y)+\varepsilon) & \text{if } L_R^{(i)}(x) < L_F^{(i)}(x), \\ -\log(1-M^{(i)}(y)+\varepsilon) & \text{if } L_R^{(i)}(x) \geq L_F^{(i)}(x). \end{cases} \tag{5.14}$$

While we have found a suitable lower bound on the inner product of $\nabla_y f^{(i)}$ and $\nabla_y g^{(i)}$, unfortunately, the sum structure of $\nabla_y f$ and $\nabla_y g$ makes it hard to say anything definitive about the value of their inner product. It is *plausible* to assume that $\nabla_y f$ and $\nabla_y g$ will be sufficiently close to guarantee convergence in practical settings. However, the theory developed in Theorem 1 does not directly apply.

Chapter 6

Joint Unsupervised Learning of Depth, Camera Motion, Optical Flow and Motion Segmentation

In the previous chapter, we introduced Competitive Collaboration to facilitate coordinated training of multiple neural networks. We now use this framework to train multiple neural networks that address geometry and motion in a scene, thereby providing explicit constraints on scene geometry for motion estimation.

We need to solve four different problems to jointly reason about scene structure and motion – depth prediction, camera motion estimation, optical flow, and motion segmentation. Depth prediction and camera motion estimation account for static scene structure whereas optical flow accounts for independent object motion. The segmentation of a video into static scene structure and independently moving regions is jointly estimated in our framework using motion segmentation. Therefore, we formulate our problem as joint unsupervised learning of single view depth prediction, camera motion estimation, optical flow, and segmentation of a video into the static scene and moving regions (see Figure 6.1).

6.1 Introduction

Deep learning methods have achieved state-of-the-art results on computer vision problems with supervision using large amounts of data [55, 72, 79]. However, for many vision problems requiring dense, continuous-valued outputs, it is either impractical or expensive to gather ground truth data [43]. We consider four such problems – single view depth prediction, camera motion estimation, optical flow, and motion segmentation. Previous work has approached these problems with supervision using real [34] and synthetic data [33]. However there is always a realism gap between synthetic and real data, and real data is limited or inaccurate. For example, depth ground truth obtained using LIDAR [43] is sparse. Furthermore, there are no sensors that provide ground truth optical flow, so all existing datasets with real imagery are limited or approximate [10, 43, 65].

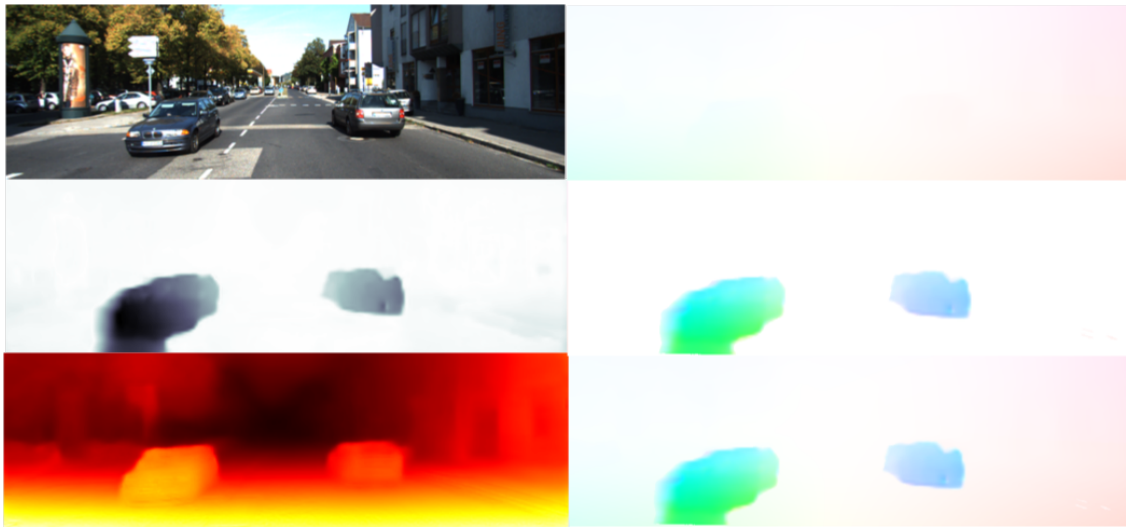


Figure 6.1: Unsupervised Learning of Depth, Camera Motion, Optical Flow and Motion Segmentation. Left, top to bottom: sample image, soft masks representing motion segmentation, estimated depth map. Right, top to bottom: static scene optical flow, segmented flow in the moving regions and combined optical flow.

Motion segmentation ground-truth currently requires manual labeling of all pixels in an image [101].

Problem

Recent work has tried to address the problem of limited training data using unsupervised learning [66, 87]. To learn a mapping from pixels to flow, depth, and camera motion without ground truth is challenging because each of these problems is highly ambiguous. To address this, additional constraints are needed and the geometric relations between static scenes, camera motion, and optical flow can be exploited. For example, unsupervised learning of depth and camera motion has been coupled in [84, 159]. They use an explainability mask to exclude evidence that cannot be explained by the static scene assumption. Yin et al. [156] extend this to estimate optical flow as well and use forward-backward consistency to reason about unexplained pixels. These methods perform poorly on depth [159] and optical flow [156] benchmarks. A key reason is that the constraints applied here do not segment objects that move independently like people and cars. More generally, not all the data in the unlabeled training set will conform to the model assumptions, and some of it might corrupt the network training. For instance, the training data for depth and camera motion should not contain independently moving objects. Similarly, for optical flow, the data should not contain occlusions, which disrupt the commonly used photometric loss.

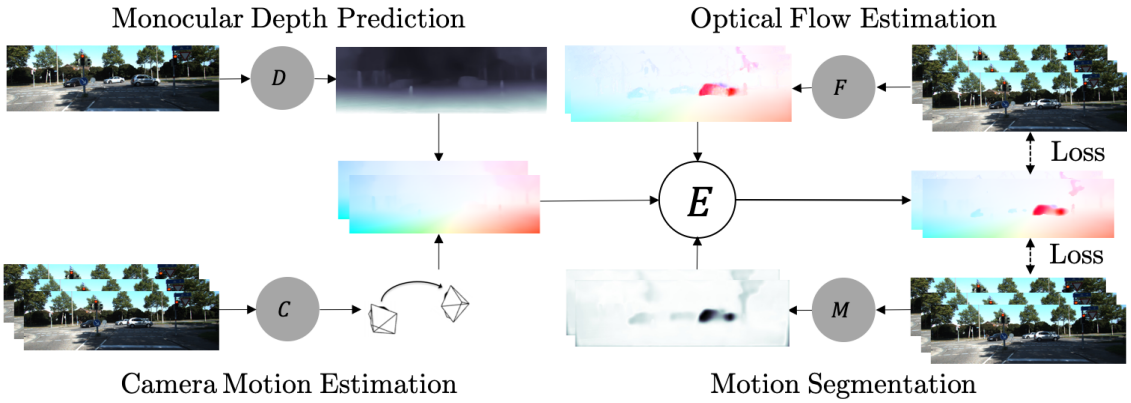


Figure 6.2: The network $R = (D, C)$ reasons about the scene by estimating optical flow over static regions using depth, D, and camera motion, C. The optical flow network F estimates flow over the whole image. The motion segmentation network, M, masks out static scene pixels from F to produce composite optical flow over the full image. A loss, E, using the composite flow is applied over neighboring frames to train all these models jointly.

Idea

A typical real-world scene consists of static regions, which do not move in the physical world, and moving objects [153]. Given depth and camera-motion, we can reason about the static scene in a video sequence. Optical flow, in contrast, reasons about all parts of the scene. Motion segmentation classifies a scene into static and moving regions. Our key insight is that these problems are coupled by the geometry and motion of the scene; therefore solving them jointly is synergistic. We show that by learning jointly from unlabeled data, our coupled networks can partition the dataset and use only the relevant data, resulting in more accurate results than learning without this synergy.

Approach

Competitive Collaboration is a three-player game consisting of two players competing for a resource that is regulated by a moderator. As shown in Figure 6.2, we introduce two players in our framework, the static scene reconstructor, $R = (D, C)$, that reasons about the static scene pixels using depth, D, and camera motion, C; and a moving region reconstructor, F, that reasons about pixels in the independently moving regions. These two players compete for training data by reasoning about static-scene and moving-region pixels in an image sequence. The competition is moderated by a motion segmentation network, M, that segments the static scene and moving regions, and distributes training data to the players. However, the moderator also needs training to ensure a fair competi-

tion. Therefore, the players, R and F, collaborate to train the moderator, M, such that it classifies static and moving regions correctly in alternating phases of the training cycle.

Contributions

We show that jointly training networks with this framework has a synergistic effect on the performance of all the networks in this framework. To our knowledge, our method is the first to use low level information like depth, camera motion and optical flow to solve a segmentation task without any supervision. We achieve state-of-the-art performance on single view depth prediction and camera motion estimation among unsupervised methods. We achieve state of art performance on optical flow among unsupervised methods methods that reason about the geometry of the scene, and introduce the first baseline for fully unsupervised motion segmentation. We even outperform competing methods that use much larger networks [156] and multiple refinement steps such as network cascading [87]. All our models and code are available at `https://github.com/anuragranj/cc`.

6.2 Approach

We use Competitive Collaboration as discussed in the previous chapter, and set up 3 players for jointly learning depth, camera motion, optical flow and motion segmentation. The first player $R = (D,C)$ consists of the depth and camera motion networks that reason about the static regions in the scene. The second player F is the optical flow network that reasons about the moving regions. For training the competitors, the motion segmentation network M selects networks (D,C) on pixels that are static and selects F on pixels that belong to moving regions. The competition ensures that (D,C) reasons only about the static parts and prevents moving pixels from corrupting its training. Similarly, it prevents any static pixels from appearing in the training loss of F, thereby improving its performance in the moving regions. In the second phase of the training cycle, the competitors (D,C) and F now collaborate to reason about static scene and moving regions by forming a consensus that is used as a loss for training the moderator, M. We now formulate the joint unsupervised estimation of depth, camera motion, optical flow and motion segmentation within the CC framework.

Notation. We use $\{D_\theta, C_\phi, F_\psi, M_\chi\}$ to denote the networks that estimate depth, camera motion, optical flow and motion segmentation respectively. The subscripts $\{\theta, \phi, \psi, \chi\}$ are the network parameters. We will omit the subscripts in several places for brevity. Consider an image sequence I_-, I, I_+ with target frame I and temporally neighboring reference frames I_-, I_+. In general, we can have many neighboring frames. In our implementation, we use 5-frame sequences for C_ϕ and M_χ but for simplicity use 3 frames

to describe our approach. We estimate the depth of the target frame as

$$d = D_\theta(I). \tag{6.1}$$

We estimate the camera motion, e, of each of the reference frames I_-, I_+ w.r.t. the target frame I as

$$e_-, e_+ = C_\phi(I_-, I, I_+). \tag{6.2}$$

Similarly, we estimate the segmentation of the target image into the static scene and moving regions. The optical flow of the static scene is defined only by the camera motion and depth. This generally refers to the structure of the scene. The moving regions have independent motion w.r.t. the scene. The segmentation masks corresponding to each pair of target and reference image are given by

$$m_-, m_+ = M_\chi(I_-, I, I_+), \tag{6.3}$$

where $m_-, m_+ \in [0,1]^\Omega$ represent the probabilities of regions being static in spatial pixel domain, Ω. Finally, the network F_ψ estimates the optical flow. F_ψ works with 2 images at a time, and its weights are shared while estimating u_-, u_+, the backward and forward optical flow[1] respectively.

$$u_- = F_\psi(I, I_-), \qquad u_+ = F_\psi(I, I_+). \tag{6.4}$$

Loss. We learn the parameters of the networks $\{D_\theta, C_\phi, F_\psi, M_\chi\}$ by jointly minimizing the energy

$$E = \lambda_R E_R + \lambda_F E_F + \lambda_M E_M + \lambda_C E_C + \lambda_S E_S, \tag{6.5}$$

where $\{\lambda_R, \lambda_F, \lambda_M, \lambda_C, \lambda_S\}$ are the weights on the respective energy terms. The terms E_R and E_F are the objectives that are minimized by the two competitors reconstructing static and moving regions respectively. The competition for data is driven by E_M. A larger weight λ_M will drive more pixels towards the static scene reconstructor. The term E_C drives the collaboration, and E_S is a smoothness regularizer. The static scene term, E_R minimizes the photometric loss on the static scene pixels given by

$$E_R = \sum_{s \in \{+,-\}} \sum_{\Omega} \rho\Big(I, w_c(I_s, e_s, d)\Big) \cdot m_s, \tag{6.6}$$

where Ω is the spatial pixel domain, ρ is a robust error function, and w_c warps the reference frames towards the target frame according to depth d and camera motion e.

[1] Note that this is different from the forward and backward optical flow in the context of two-frame estimation.

Similarly, E_F minimizes photometric loss on moving regions

$$E_F = \sum_{s\in\{+,-\}} \sum_{\Omega} \rho\Big(I, w_f(I_s, u_s)\Big) \cdot (1 - m_s), \tag{6.7}$$

where w_f warps the reference image using flow u. We show the formulations for w_c, w_f in the Appendix A. We compute the robust error $\rho(x,y)$ as

$$\rho(x,y) = \lambda_\rho \sqrt{(x-y)^2 + \varepsilon^2} + (1-\lambda_\rho)\left[1 - \frac{(2\mu_x\mu_y + c_1)(2\mu_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x + \sigma_y + c_2)}\right], \tag{6.8}$$

where λ_ρ is a fixed-constant and $\varepsilon = 0.01$. The second term is known as the structure similarity loss (SSIM) [145] that has been used in previous work [84, 156], μ_x, σ_x are the local mean and variance over the pixel neighborhood with $c_1 = 0.01^2$ and $c_2 = 0.03^2$.

The loss E_M minimizes the cross entropy, H, between the masks and a unit tensor regulated by λ_M

$$E_M = \sum_{s\in\{+,-\}} \sum_{\Omega} H(\mathbf{1}, m_s). \tag{6.9}$$

A larger λ_M gives preference to the static scene reconstructor R, biasing the scene towards being static.

Let $\nu(e,d)$ represent the optical flow induced by camera motion e and depth d, as described in the Appendix A. The consensus loss E_C drives the collaboration and constrains the masks to segment moving objects by taking a consensus between flow of the static scene given by $\nu(e,d)$ and optical flow estimates from F_ψ. It is given by

$$E_C = \sum_{s\in\{+,-\}} \sum_{\Omega} H\big(\mathbb{I}_{\rho_R < \rho_F} \vee \mathbb{I}_{||\nu(e_s,d) - u_s|| < \lambda_c}, m_s\big), \tag{6.10}$$

where $\mathbb{I} \in \{0,1\}$ is an indicator function that equals 1 if the condition in the subscript is true. The first indicator function favors mask assignments to the competitor that achieves lower photometric error on a pixel by comparing $\rho_R = \rho(I, w_c(I_s, e_s, d))$ and $\rho_F = \rho(I, w_f(I_s, u_s))$. In the second indicator function, the threshold λ_c forces $\mathbb{I} = 1$ if the static scene flow $\nu(e,d)$ is close to the optical flow u, indicating a static scene. The symbol $\vee$ denotes logical OR between indicator functions. The consensus loss E_C encourages a pixel to be labeled as static if R has a lower photometric error than F or if the induced flow of R is similar to that of F. Finally, the smoothness term E_S acts as a regularizer on depth, segmentations and flow,

$$E_S = \sum_{\Omega} ||\lambda_e \nabla d||^2 + ||\lambda_e \nabla u_-||^2 + ||\lambda_e \nabla u_+||^2 + ||\lambda_e \nabla m_-||^2 + ||\lambda_e \nabla m_+||^2, \tag{6.11}$$

where $\lambda_e = e^{-||\nabla I||}$ (elementwise) and ∇ is the first derivative along spatial directions.

The term λ_e ensures that smoothness is guided by edges of the images.

Inference. The depth d and camera motion e are directly inferred from network outputs. The motion segmentation m^* is obtained by the output of mask network M_χ and the consensus between the static flow and optical flow estimates from F_χ. It is given by

$$m^* = \mathbb{I}_{m_+ \cdot m_- > 0.5} \vee \mathbb{I}_{||\nu(e_+,d) - u_+|| < \lambda_c}. \tag{6.12}$$

The first term takes the intersection of mask probabilities inferred by M_χ using forward and backward reference frames. The second term takes a consensus between flow estimated from $R = (D_\theta, C_\phi)$ and F_ψ to reason about the masks. The final masks are obtained by taking the union of both terms. Finally, the full optical flow, u^*, between (I, I_+) is a composite of optical flows from the static scene and the independently moving regions given by

$$u^* = \mathbb{I}_{m^* > 0.5} \cdot \nu(e_+, d) + \mathbb{I}_{m^* \leq 0.5} \cdot u_+. \tag{6.13}$$

The loss in Eq. (6.5) is formulated to minimize the reconstruction error of the neighboring frames. Two competitors, the static scene reconstructor $R = (D_\theta, C_\phi)$ and moving region reconstructor F_ψ minimize this loss. The reconstructor R reasons about the static scene using Eq. (6.6) and the reconstructor F_ψ reasons about the moving regions using Eq. (6.7). The moderation is achieved by the mask network, M_χ using Eq. (6.9). Furthermore, the collaboration between R, F is driven using Eq. (6.10) to train the network M_χ.

If the scenes are completely static, and only the camera moves, the mask forces (D_θ, C_ϕ) to reconstruct the whole scene. However, (D_θ, C_ϕ) are wrong in the independently moving regions of the scene, and these regions are reconstructed using F_ψ. The moderator M_χ is trained to segment static and moving regions correctly by taking a consensus from (D_θ, C_ϕ) and F_ψ to reason about static and moving parts on the scene, as seen in Eq. (6.10). Therefore, our training cycle has two phases. In the first phase, the moderator M_χ drives competition between two models (D_θ, C_ϕ) and F_ψ using Eqs. ((6.6), (6.7)). In the second phase, the competitors (D_θ, C_ϕ) and F_ψ collaborate together to train the moderator M_χ using Eqs. ((6.9), (6.10)).

6.3 Experiments

Network Architecture

For the depth network, we experiment with DispNetS [159] and DispResNet where we replace convolutional blocks with residual blocks [56]. The network D_θ takes a single RGB image as input and outputs depth. For the flow network, F_ψ, we experiment with both FlowNetC [33] and PWC-Net [130]. The PWC-Net uses the multi-frame unsupervised learning framework from Janai et al. [64]. The network F_ψ computes optical flow between a pair of frames. The networks C_ϕ, M_χ take a 5 frame sequence

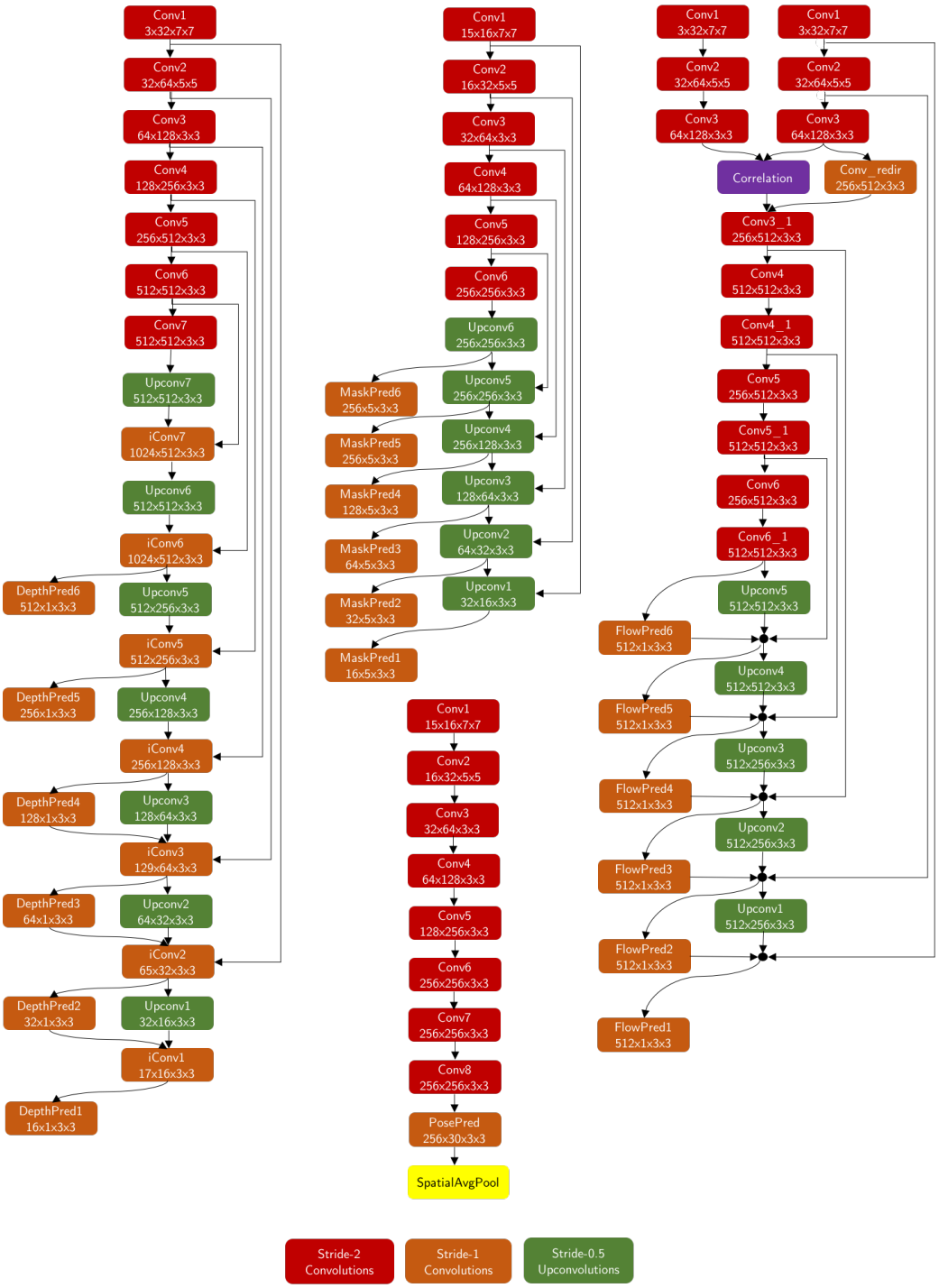


Figure 6.3: Architecture of the DispNet (left), MaskNet (center-top), FlowNetC (right) and Camera Motion Network (center-bottom). Convolutional layers are red (stride 2) and orange (stride 1) and upconvolution layers are green (stride 2). Other colors refer to special layers. Each layer is followed by ReLU, except prediction layers. In each block, the numbers indicate the number of channels of the input feature map, the number of channels of the output feature map, and the filter size.

$(I_{--}, I_-, I, I_+, I_{++})$ as input. The mask network M_χ has an encoder-decoder architecture. The encoder consists of stacked residual convolutional layers. The decoder has stacked upconvolutional layers to produce masks $(m_{--}, m_-, m_+, m_{++})$ of the reference frames. The camera motion network C_ϕ consists of stacked convolutions followed by adaptive average pooling of feature maps to get the camera motions $(e_{--}, e_-, e_+, e_{++})$. The networks D_θ, F_ψ, M_χ output their results at 6 different spatial scales that are used for training. The predictions at the finest scale are used. The highest scale is of the same resolution as the image, and each lower scale reduces the resolution by a factor of 2. We now describe architecture details of each of the networks as shown in Figure 6.3.

Depth Network D. Our depth network is similar to DispNetS [86] and outputs depths at 6 different scales. Each convolution and upconvolution is followed by a ReLU except the prediction layers. The prediction layer at each scale has a non-linearity given by $1/(\alpha$ sigmoid$(x)+\beta)$. The architecture of DispResNet is obtained by replacing convolutional blocks in DispNet by residual blocks [56].

Camera Motion Network C. The camera motion network consists of 8 convolutional layers, each of stride 2 followed by a ReLU activation. This is followed by a convolutional layer of stride 1, whose feature maps are averaged together to get the camera motion.

Flow Network F. We use the FlowNetC architecture [33] with 6 output scales of flow as shown in Figure 6.3. All convolutional and upconvolutional layers are followed by a ReLU except prediction layers. The prediction layers have no activations. For PWC Net, we use the network architecture from Janai et al. [64].

Mask Network M. The mask network has a U-Net [33] architecture. The encoder is similar to the camera motion with 6 convolutional layers. The decoder has 6 upconvolutional layers. Each of these layers have ReLU activations. The prediction layers use a sigmoid.

Network Training

We use raw KITTI sequences [43] for training using Eigen et al.'s split [34] that is consistent across related works [34, 78, 84, 156, 159, 161]. We train the networks with a batch size of 4 and learning rate of 10^{-4} using ADAM [71] optimization. The images are scaled to 256×832 for training. The data is augmented with random scaling, cropping and horizontal flips. We use Algorithm 1 for training. Initially, we train (D_θ, C_ϕ) with only photometric loss over static pixels E_R and smoothness loss E_S while other loss terms are set to zero. Similarly, we train F_ψ independently with photometric loss over all pixels and

smoothness losses. The models $(D_\theta, C_\phi), F_\psi$ at this stage are referred to as 'basic' models in our experiments. We then learn M_χ using the joint loss. We use $\lambda_R = 1.0, \lambda_F = 0.5$ for joint training because the static scene reconstructor R uses 4 reference frames in its loss, whereas the optical flow network F uses 2 frames. Hence, these weights normalize the loss per neighboring frame. We iteratively train $(D_\theta, C_\phi), F_\psi, M_\chi$ using the joint loss while keeping the other network weights fixed. The consensus weight $\lambda_C = 0.3$ is used only while training the mask network. Other constants are fixed with $\lambda_S = 0.005$, and threshold in Eq. (6.12), $\lambda_c = 0.001$. The constant $\lambda_\rho = 0.003$ regulates the SSIM loss and is chosen empirically. We iteratively train the competitors $(D_\theta, C_\phi), F_\psi$ and moderator M_χ for about 100,000 iterations at each step until validation error saturates.

Qualitative results of the predictions are shown in Figure 6.4. We would like to point out that our method is able to segment the moving car, and not the parked cars on the roads. In addition, it segments other moving objects, such as the bicyclist.

Result: Trained Network Parameters, $(\theta, \phi, \psi, \chi)$
Define $\lambda = (\lambda_R, \lambda_F, \lambda_M, \lambda_C)$
Randomly initialize $(\theta, \phi, \psi, \chi)$
Update (θ, ϕ) by jointly training (D_θ, C_ϕ) with $\lambda = (1.0, 0.0, 0.0, 0.0)$
Update ψ by training F_ψ with $\lambda = (0.0, 1.0, 0.0, 0.0)$
Update χ by jointly training $(D_\theta, C_\phi, F_\psi, M_\chi)$ with $\lambda = (1.0, 0.5, 0.0, 0.0)$
Loop
  Competition Step
    Update θ, ϕ by jointly training $(D_\theta, C_\phi, F_\psi, M_\chi)$ with $\lambda = (1.0, 0.5, 0.05, 0)$
    Update ψ by jointly training $(D_\theta, C_\phi, F_\psi, M_\chi)$ with $\lambda = (0.0, 1.0, 0.005, 0)$
  Collaboration Step
    Update χ by jointly training $(D_\theta, C_\phi, F_\psi, M_\chi)$ with $\lambda = (1.0, 0.5, 0.005, 0.3)$
EndLoop

Algorithm 1: Competitive Collaboration Training Algorithm

Monocular Depth and Camera Motion Estimation

KITTI Dataset. We obtain state-of-the-art results on single view depth prediction and camera motion estimation as shown in Tables 6.1 and 6.3. The depth is evaluated on the Eigen et al. [34] split of the raw KITTI dataset [43] and camera motion is evaluated on the KITTI Odometry dataset [43]. These evaluation frameworks are consistent with previous work [34, 78, 84, 156]. All depth maps are capped at 80 meters. As shown in Table 6.1, by training our method only on KITTI [43], we get similar or better performance than competing methods like [156, 161] that use a much bigger Resnet-50 architecture [56] and are trained on the larger Cityscapes dataset [27]. Using Cityscapes in our training further improves our performance on depth estimation benchmarks (Table 6.1).

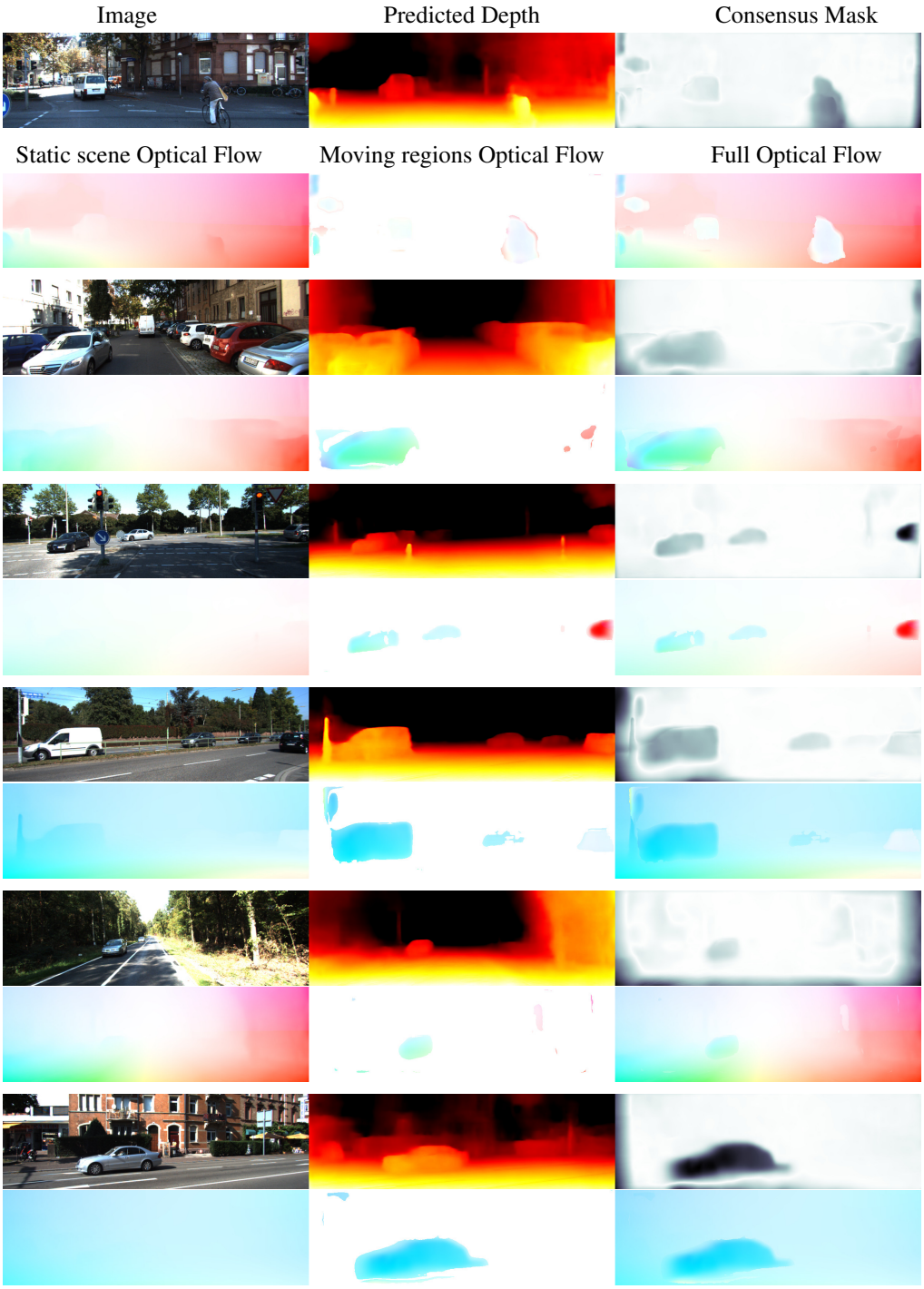


Figure 6.4: Network Predictions. In each group of images, top row: we show image, predicted depth, consensus masks, bottom row: we show static scene optical flow, segmented flow in the moving regions and full optical flow.

Method	Error				Accuracy, δ		
	AbsRel	SqRel	RMS	RMSlog	<1.25	$<1.25^2$	$<1.25^3$
Supervised Methods trained on KITTI							
Eigen et al. [34] coarse	0.214	1.605	6.563	0.292	0.673	0.884	0.957
Eigen et al. [34] fine	0.203	1.548	6.307	0.282	0.702	0.890	0.958
Liu et al. [78]	0.202	1.614	6.523	0.275	0.678	0.895	0.965
Unsupervised Methods trained on Cityscapes+KITTI							
Zhou et al. [159]	0.198	1.836	6.565	0.275	0.718	0.901	0.960
Mahjourian et al. [84]	0.159	1.231	5.912	0.243	0.784	0.923	0.970
Geonet-Resnet [156]	0.153	1.328	5.737	0.232	0.802	0.934	0.972
DF-Net [161]	0.146	1.182	5.215	**0.213**	0.818	**0.943**	**0.978**
CC (ours)	**0.139**	**1.032**	**5.199**	**0.213**	**0.827**	**0.943**	0.977
Unsupervised Methods trained on KITTI							
Zhou et al.* [159]	0.183	1.595	6.709	0.270	0.734	0.902	0.959
Mahjourian et al. [84]	0.163	1.240	6.220	0.250	0.762	0.916	0.968
Geonet-VGG [156]	0.164	1.303	6.090	0.247	0.765	0.919	0.968
Geonet-Resnet [156]	0.155	1.296	5.857	0.233	0.793	0.931	0.973
Godard et el. [47]	0.154	1.218	5.699	0.231	0.798	0.932	0.973
DF-Net [161]	0.150	1.124	5.507	0.223	0.806	0.933	0.973
CC (ours)	**0.140**	**1.070**	**5.326**	**0.217**	**0.826**	**0.941**	**0.975**

Table 6.1: Results on Depth Estimation. * refers to improved results authors' github page.

Ablation studies on depth estimation are shown in Table 6.2. In the basic mode, our network architecture DispNet for depth and camera motion estimation is most similar to Zhou et al. [159] and this is reflected in the performance of our basic model. We get some performance improvements by adding the SSIM loss [145]. However, we observe that using the Competitive Collaboration (CC) framework with a joint loss results in larger performance gains in both tasks. Further improvements are obtained by using a better network architecture, DispResNet. Greater improvements in depth estimation are obtained when we use a better network for flow, which shows that improving on one task improves the performance of the other in the CC framework (row 4 vs 5 in Table 6.2).

The camera motion estimation also shows similar performance trends as shown in Table 6.3. Using a basic model, we achieve similar performance as the baseline [159], which improves with the addition of the SSIM loss. Using the CC framework leads to further improvements in performance.

In summary, we show that joint training using CC boosts performance of single view depth prediction and camera motion estimation. We compare qualitative results for single image depth prediction in Figure 6.5. We also contrast our results with basic models that were trained independently without a joint loss in Figure 6.6. We observe that our model produces better results, capturing moving objects such as cars and bikes, as well as surface edges of trees, pavements and buildings.

Method	Data	Net D	Net F	Error			
				AbsRel	SqRel	RMS	RMSlog
Basic	K	DispNet	-	0.184	1.476	6.325	0.259
Basic + ssim	K	DispNet	-	0.168	1.396	6.176	0.244
CC + ssim	K	DispNet	FlowNetC	0.148	1.149	5.464	0.226
CC + ssim	K	DispResNet	FlowNetC	0.144	1.284	5.716	0.226
CC + ssim	K	DispResNet	PWC Net	0.140	1.070	5.326	0.217
CC + ssim	CS+K	DispResNet	PWC Net	**0.139**	**1.032**	**5.199**	**0.213**

Method	Data	Net D	Net F	Accuracy, δ		
				<1.25	$<1.25^2$	$<1.25^3$
Basic	K	DispNet	-	0.732	0.910	0.967
Basic + ssim	K	DispNet	-	0.767	0.922	0.971
CC + ssim	K	DispNet	FlowNetC	0.815	0.935	0.973
CC + ssim	K	DispResNet	FlowNetC	0.822	0.938	0.973
CC + ssim	K	DispResNet	PWC Net	0.826	0.941	0.975
CC + ssim	CS+K	DispResNet	PWC Net	**0.827**	**0.943**	**0.977**

Table 6.2: Ablation studies on Depth Estimation. Joint training using Competitive Collaboration and better architectures improve the results. The benefits of CC can be seen when depth improves by using a better network for flow (row 4 vs 5). K refers to the KITTI dataset and CS refers to the Cityscapes dataset.

Method	Sequence 09	Sequence 10
ORB-SLAM (full)	0.014 ± 0.008	0.012 ± 0.011
ORB-SLAM (short)	0.064 ± 0.141	0.064 ± 0.130
Mean Odometry	0.032 ± 0.026	0.028 ± 0.023
Zhou et al. [159]	0.016 ± 0.009	0.013 ± 0.009
Mahjourian et al. [84]	0.013 ± 0.010	0.012 ± 0.011
Geonet [156]	$\mathbf{0.012 \pm 0.007}$	0.012 ± 0.009
DF-Net [161]	0.017 ± 0.007	0.015 ± 0.009
Basic (ours)	0.022 ± 0.010	0.018 ± 0.011
Basic + ssim (ours)	0.017 ± 0.009	0.015 ± 0.009
CC + ssim (ours)	$\mathbf{0.012 \pm 0.007}$	$\mathbf{0.012 \pm 0.008}$

Table 6.3: Average Trajectory Errors on Camera Pose Estimation.

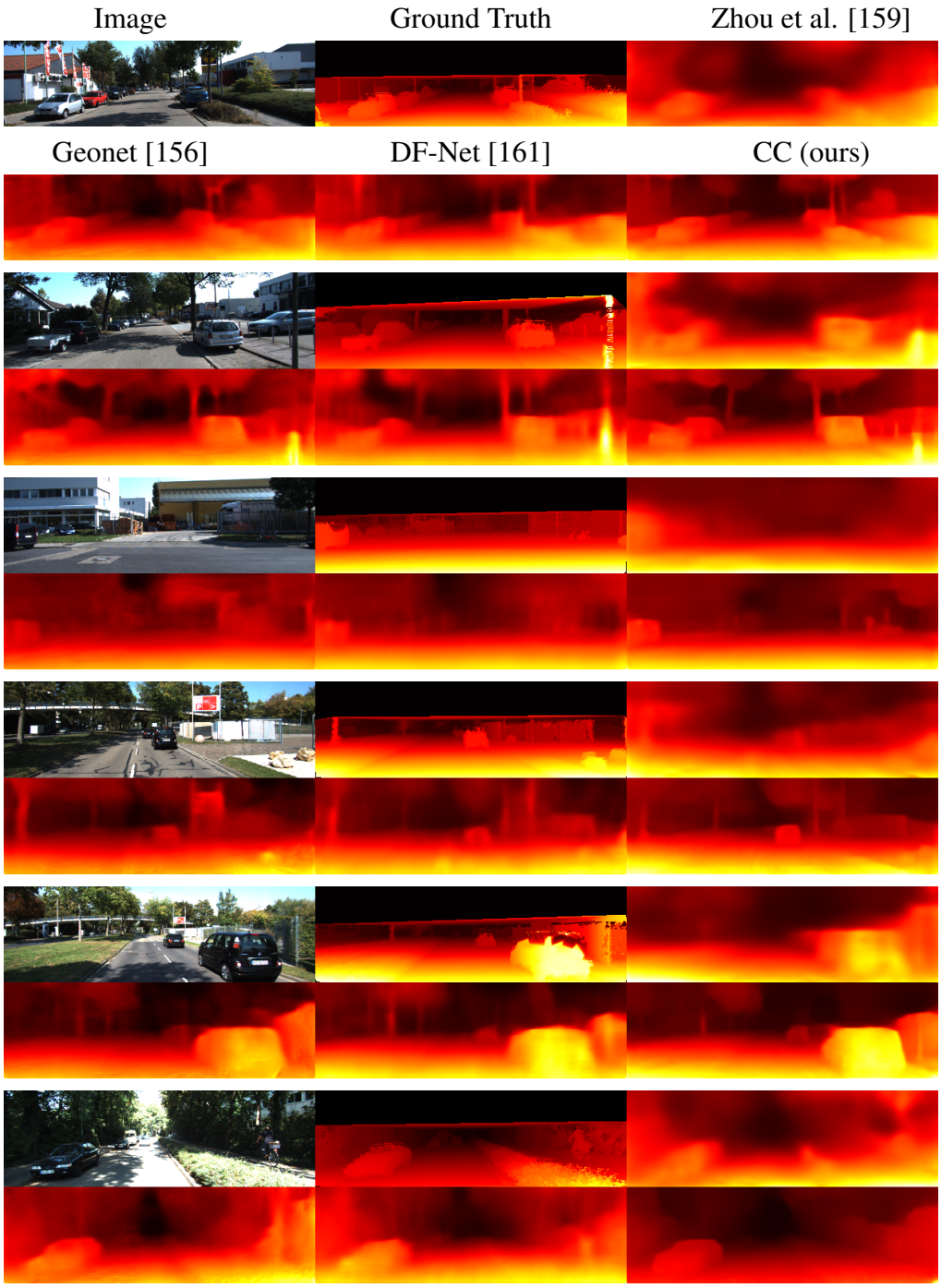


Figure 6.5: Qualitative results on single view depth prediction. In each group of images, top row: we show image, interpolated ground truth depths, Zhou et al. [159] results, bottom row: we show results from Geonet [156], DF-Net [161] and CC (ours) results.

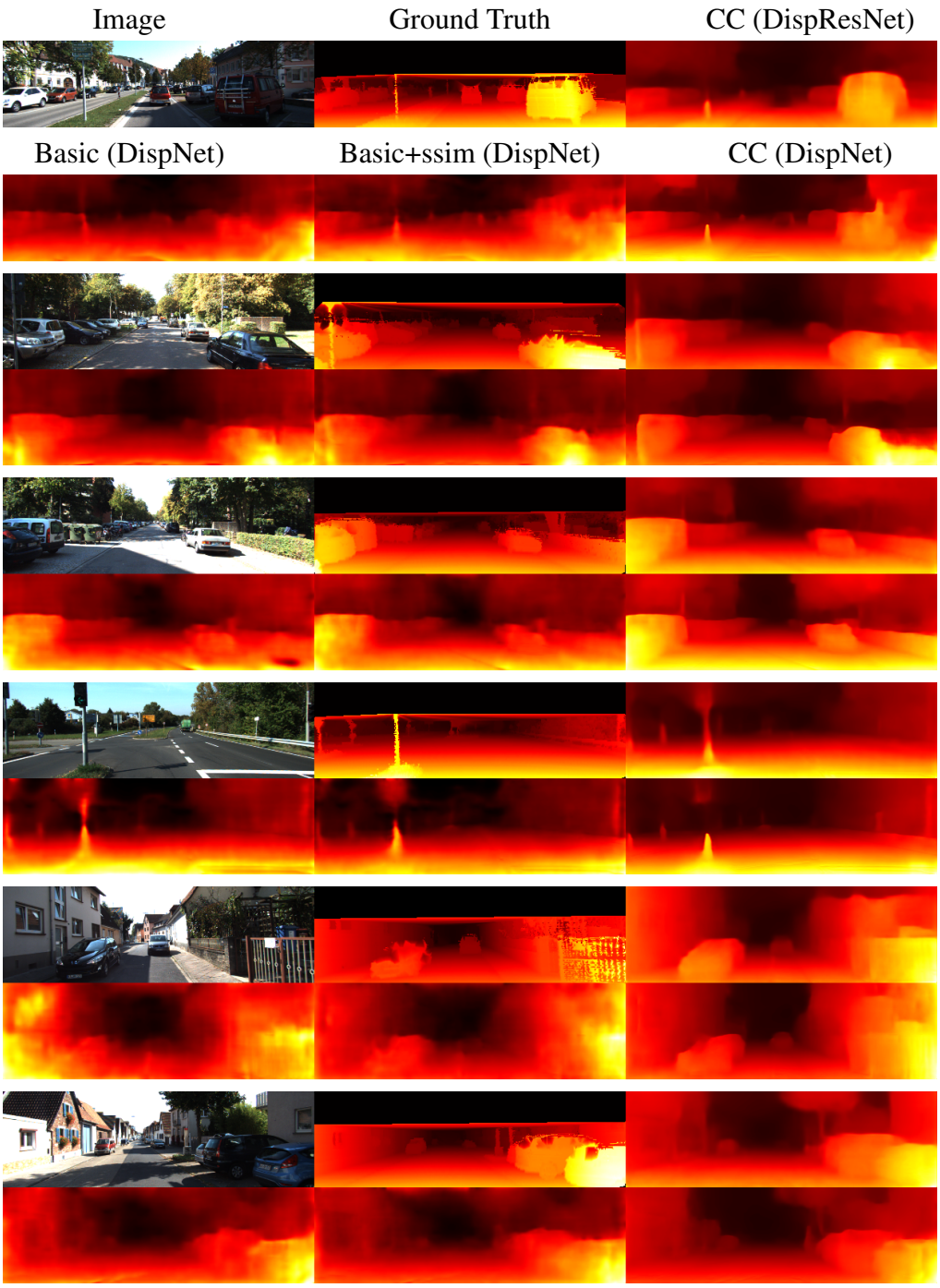


Figure 6.6: Ablation studies on single view depth prediction. In each group of images, top row: we show image, interpolated ground truth depths, CC using DispResNet architecture, bottom row: we show results using Basic, Basic+ssim and CC models using DispNet architecture.

Zhou [159]	DF-Net [161]	Godard [47]	CC (ours)
0.383	0.331	0.361	**0.320**

Table 6.4: Absolute Relative errors on Make3D test set.

Make3D Dataset. We also test on the Make3D dataset [117] without training on it. We use our model that is trained only on Cityscapes and KITTI. Our method outperforms previous work [47, 159, 161] as shown in Table 6.4. We show qualitative results in Figure 6.7.

Optical Flow Estimation

We compare the performance of our approach with competing methods using the KITTI 2015 training set [43] to be consistent with previous work [87, 156]. We obtain state-of-the-art performance among joint methods as shown in Table 6.5. Unsupervised fine tuning (CC-uft) by setting $\lambda_M = 0.02$ gives more improvements than CC as masks now choose the best flow between R and F without being overconstrained to choose R. In contrast, UnFlow-CSS [87] uses 3 cascaded networks to refine optical flow at each stage. Geonet [156] and DF-Net [161] are more similar to our architecture but use a larger ResNet-50 architecture. Back2Future [64] performs better than our method in terms of outlier error, but not in terms of average end point error.

In Table 6.6, we observe that training the static scene reconstructor R or moving region reconstructor F independently leads to worse performance. This happens because R can not reason about dynamic moving objects in the scene. Similarly F is not as good as R for reasoning about static parts of the scene, especially in occluded regions. Using them together, and compositing the optical flow from both as shown in Eq. (6.13) leads to a large improvement in performance. Moreover, using better network architectures further improves the performance under the CC framework.

We compare qualitative results for optical flow estimation in Figure 6.8. We show that our method performs better than UnFlow [87], Geonet [156] and DF-Net [161]. Our flow estimations are better at the boundaries of the cars and pavements. In contrast, competing methods produce blurry flow fields.

Motion Segmentation

We evaluate the estimated motion segmentations using the KITTI 2015 training set [43] which provides ground-truth segmentation for moving cars. Since our approach does not distinguish between different semantic classes while estimating segmentation, we evaluate segmentations only on car pixels. Specifically, we consider car pixels and compute Intersection over Union (IoU) scores for moving and static car pixels. In Table 6.7, we show the IoU scores of the segmentation masks obtained using our technique under

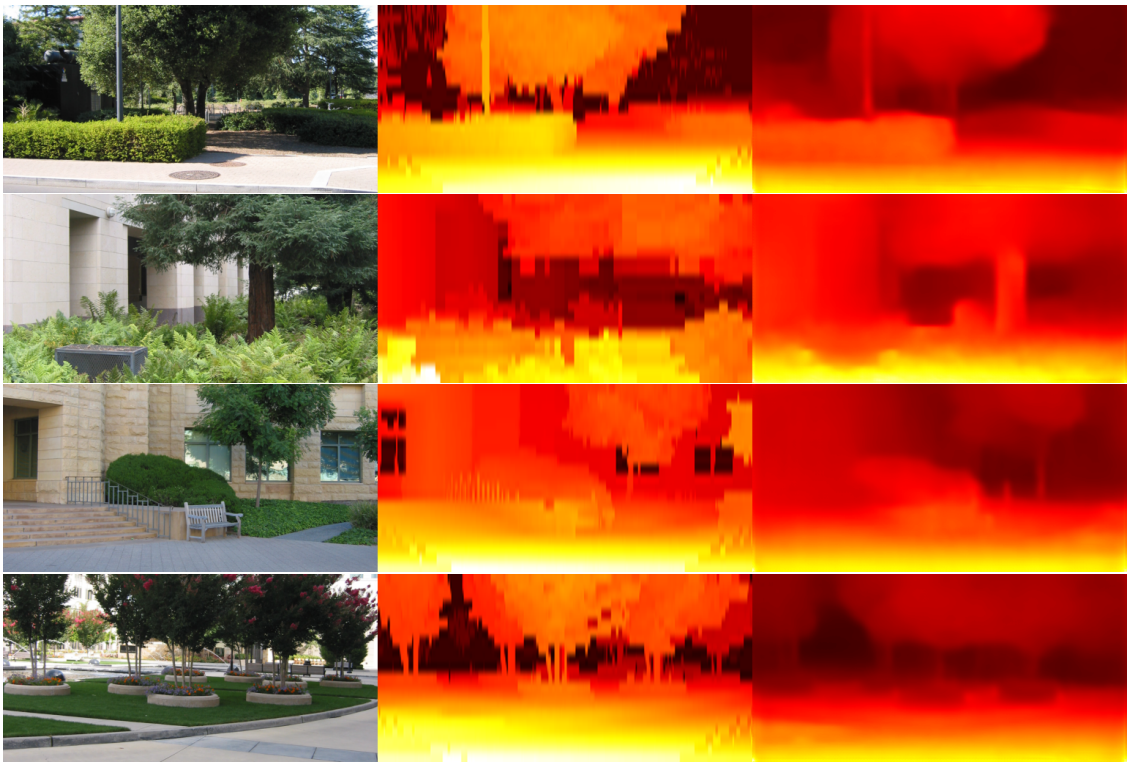


Image Ground Truth Prediction

Figure 6.7: Qualitative results on the Make3D test set.

	Train		Test
Method	EPE	Fl	Fl
FlowNet2 [61]	10.06	30.37 %	-
SPyNet [103]	20.56	44.78%	-
UnFlow-C [87]	8.80	28.94%	29.46%
UnFlow-CSS [87]	8.10	**23.27%**	-
Back2Future [64]	**6.59**	-	**22.94%**
Back2Future* [64]	7.04	24.21%	-
Geonet [156]	10.81	-	-
DF-Net [161]	8.98	26.01%	25.70%
CC (ours)	6.21	26.41%	-
CC-uft (ours)	**5.66**	**20.93%**	**25.27%**

Table 6.5: Results on Optical Flow. We compare with supervised methods (top 2 rows) that are trained on synthetic data only; unsupervised methods specialized for optical flow (middle 4 rows) and joint methods that solve more than one task (bottom 4 rows). * refers to our Pytorch implementation used in our framework which gives slightly lower accuracy.

			Average EPE		
Method	**Net D**	**Net F**	Static Pixels	Moving Pixels	Total
R	DispNet	-	7.51	32.75	13.54
F	-	FlowNetC	15.32	6.20	14.68
CC	DispNet	FlowNetC	6.35	6.16	7.76
CC	DispResNet	PWC Net	5.67	5.04	6.21

Table 6.6: Ablation studies on Flow estimation. EPE is computed over the KITTI 2015 training set. R, F are trained independently without CC.

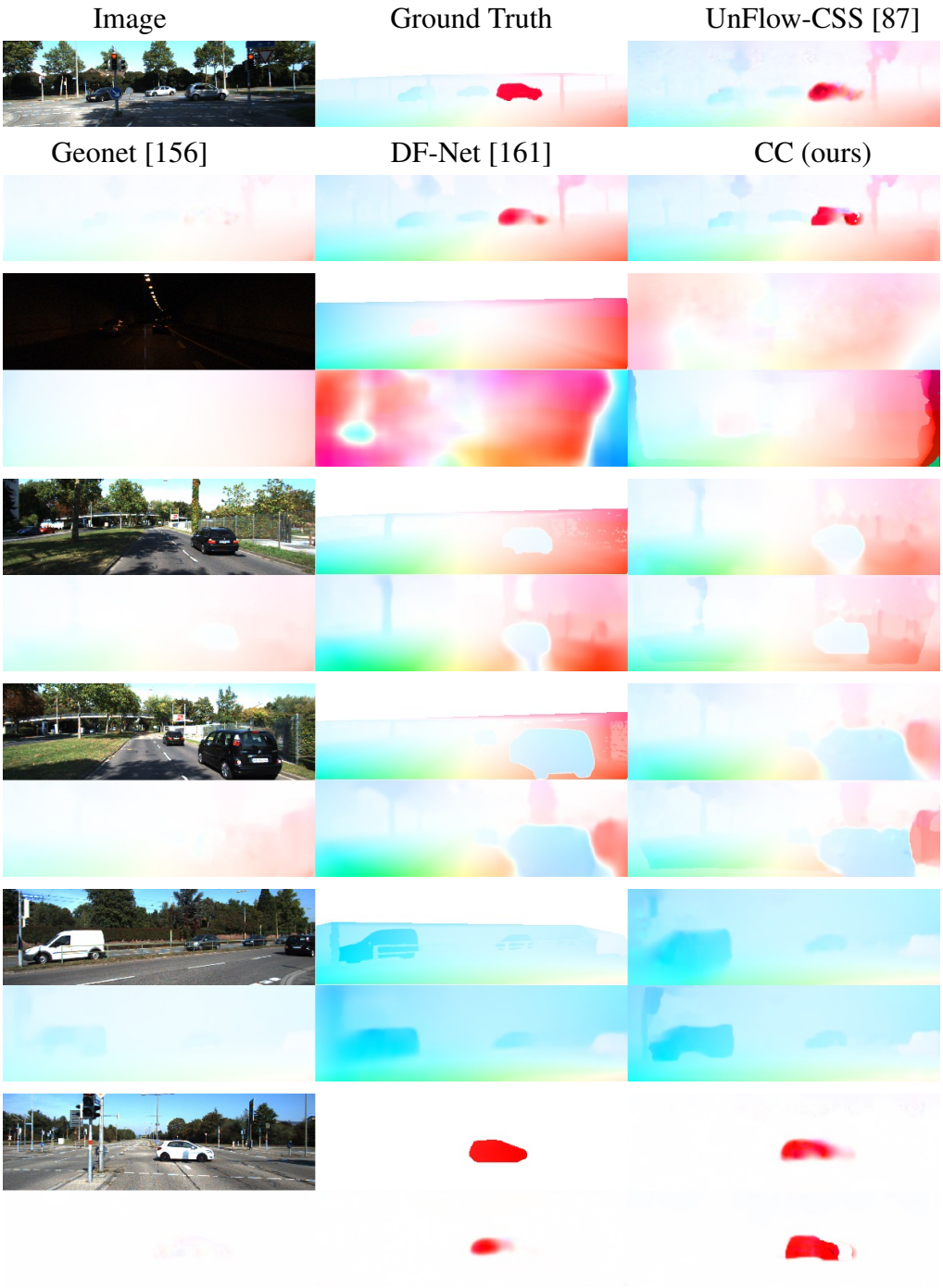


Figure 6.8: Qualitative results on Optical Flow estimation. In each group of images, top row: we show image 1, ground truth flow, and predictions from UnFlow-CSS[87], bottom row: we show predictions from Geonet [156], DF-Net [161] and CC (ours) model.

	Overall	Static Car	Moving Car
MaskNet	41.64	30.56	52.71
Consensus	51.52	47.30	55.74
Joint	**56.94**	**55.77**	**58.11**

Table 6.7: Motion Segmentation Results. Intersection Over Union (IoU) scores on KITTI2015 training dataset images computed over car pixels.

Method	Depth	Pose	Flow	Mask
Geonet [156]	15*ms*	4*ms*	45*ms*	-
CC (ours)	13*ms*	2*ms*	34*ms*	3*ms*

Table 6.8: Average runtime on TitanX GPU with images of size 128×418.

different conditions. We refer to the masks obtained with the motion segmentation network $\left(\mathbf{I}_{m_- m_+ > 0.5}\right)$ as 'MaskNet' and refer to the masks obtained with flow consensus $\left(\mathbf{I}_{||v(e_+,d)-u_+||<\lambda_c}\right)$ as 'Consensus'. The final motion segmentation masks m^* obtained with the intersection of the above two estimates are referred to as 'Joint' Eq. (6.12). IoU results indicate substantial IoU improvements with 'Joint' masks compared to both 'MaskNet' and 'Consensus' masks, illustrating the complementary nature of different masks. Qualitative results are shown in Figure 6.4.

Timing Analysis

We analyze inference time of our network and compare it with Geonet [156] in Table 6.8. We observe that our networks have a faster run time using the same sized 128×416 images on a single TitanX GPU. This is because our networks are simpler and smaller than ones used by Geonet.

For training, we measure the time taken for each iteration using a batch size of 4. Training depth and camera motion networks (D, C) takes $0.96s$ per iteration. Training the mask network, M takes $0.48s$ per iteration, and the flow network F takes $1.32s$ per iteration. All iterations have a batch size of 4. In total, it takes about 7 days for all the networks to train starting with random initialization on a single 16GB Tesla V100.

6.4 Conclusion and Discussion

Typically, learning to infer depth from a single image requires training images with ground-truth depth scans, and learning to compute optical flow relies on synthetic data, which may not generalize to real image sequences. For static scenes, observed by a

moving camera, these two problems are related by camera motion; depth and camera motion completely determine the 2D optical flow. This holds true over several frames if the scene is static and only the camera moves. Thus by combining depth, camera, and flow estimation, we can learn single-image depth by using information from several frames during training. This is particularly critical for unsupervised training since both depth and optical flow are highly ill-posed. Combining evidence from multiple tasks and multiple frames helps to synergistically constrain the problem. This alone is not enough, however, as real scenes contain multiple moving objects that do not conform to static scene geometry. Consequently, we also learn to segment the scene into static and moving regions without supervision. In the independently moving regions, a generic flow network learns to estimate the optical flow.

To facilitate this process, we introduce Competitive Collaboration in which networks, both compete and cooperate. We demonstrate that this results in top performance among unsupervised methods for all subproblems. Additionally, the moderator learns to segment the scene into static and moving regions without any direct supervision.

Future Work

We can add small amounts of supervised training, with which we expect to significantly boost performance on benchmarks, cf. [87]. We could use, for example, sparse depth and flow from KITTI and segmentation from Cityscapes to selectively provide ground truth to different networks. A richer segmentation network together with semantic segmentation should improve non-rigid segmentation. For automotive applications, the depth map formulation should be extended to a world coordinate system, which would support the integration of depth information over long image sequences. Finally, as shown in [153], the key ideas of using layers and geometry apply to general scenes beyond the automotive case and we should be able to train this method to work with generic scenes and camera motions.

Chapter 7

Conclusion and Future Work

The motion of the world is constrained by the structure and geometry in the scene. Throughout this thesis, we have seen an intricate relationship between the structure of the world and its motion. Jointly modeling structure of the world and the motion has shown state-of-the-art performance for several computer vision problems.

Spatial Structure and Motion. In Chapter 3, we modelled the image structure at different resolution levels using a Spatial Pyramid Network. This enabled us to estimate optical flow with real-time performance using a very small neural network. Recent works have applied spatial pyramid networks to estimate optical flow in scenes containing human motion [108]. Integrating cost-volume structures in spatial pyramid networks for estimating optical flow has led to state-of-the-art performance [130]. In the future, we expect small and fast networks for estimating optical flow that can run on embedded devices. Real-time and accurate estimation of optical flow would support several applications in computer vision and robotics.

Furthermore, small networks for optical flow estimation can lead to a better insight into how we estimate and understand motion. Having a small network implies that motion estimation can be parametrized by a small set of numbers [102]. Analyzing these parameters, or learned filters has a promise of reconciling the functionality of MT (the part of the cortex that responds to motion) with computational tools like deep networks. The long-term goal is to come up with a small set of numbers that can parametrize functions to accurately compute optical flow. These functions can also represent the responses in the human visual cortex that corresponds to motion in scenes.

Human Structure and Motion. The structure of the world certainly influences the estimation of motion. In Chapter 4, we modelled structure of humans and their motion by generating synthetic data. The distribution of human motion was captured using synthetic MoCap data. Training a deep neural network on this data showed improved performance of optical flow networks on scenes containing human motion. These deep networks also showed generalization to real scenes containing human motion. This reinforces the idea of incorporating the structure of the world, in this case humans, to improve optical flow estimation. Human optical flow is important for several applications. Sev-

eral action recognition methods [123] rely on optical flow in scenes containing human action. A good optical flow estimation for humans would improve several computer vision applications that rely on accurate estimation of human motion such as human pose and shape estimation, action recognition, modeling interaction of human and scenes, and human body segmentation.

Learning Structure and Motion. So far, we have imposed constraints on the structure of the world such as spatial structure in image pyramids [103] and structure of humans [108]. This motivated us to develop a framework for jointly learning structure and motion of the world. In Chapter 5, we introduced the Competitive Collaboration framework to incorporate geometric constraints about the structure of the world in deep networks using a game-theoretic formulation. Competitive Collaboration formulates a three-player game with two competitors and one moderator that regulates the competition. This framework is useful for problems such as representation learning of multi-modal distributions, training a mixture of experts. More importantly, Competitive Collaboration enabled us to disentangle structure and motion components from a scene, where each component is handled by a competing player.

Competitive Collaboration allows independent networks for scene structure and the motion to compete and collaborate together to enable learning. In Chapter 6, we modeled the geometry of the world together with the motion to enable the joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. We showed that jointly learning these problems is synergistic and gives improved performance on all tasks.

In the future, we expect to see unification of all computer vision problems in a single framework. This would lead to tasks benefiting from each other when the training data is limited for certain tasks. Deep learning has helped us to reach a point, where we can train a model to be an expert at a particular task given enough data. However, a lot of tasks related to understanding vision have very limited or no training data. To solve them, we would need to integrate them together with tasks which are well solved.

7.1 Long-Term Goals

The long-term goal of this work is full scene understanding (Figure 7.1). This means that, given image frames of a scene, we understand the physical properties of the scene. This includes the underlying structure that can be represented by depth maps, point clouds, meshes or another representation. We should be able to estimate the motion of each and every object in the scene along with the camera motion. We should be aware of the true textures on every object and the lighting of the scene. If we truly understand a scene, we should be able to render it from different view points, replace objects in the scene, change texture of objects and lighting, and change the motion of scene objects.

Figure 7.1: Given an image, understand the scene. Reproduced from Perceiving Systems webpage, MPI `https://ps.is.tuebingen.mpg.de/research_fieldsinverse-graphics`.

Computer vision has traditionally seen geometry or structure separately from the problem of estimating motion. We have shown that approaching these problems jointly has a synergistic effect on the solutions. Modeling geometry of the scene together with its motion would enable us to approach the solution of full scene understanding.

Appendix A

Geometry and Optical Flow

A.1 Solution for Optical Flow using Depth and Camera Motion

The estimate for camera motion consists of camera rotations $sin\alpha, sin\beta$, $sin\gamma$, and translations t_x, t_y, t_z. Let $e = (sin\alpha, sin\beta, sin\gamma, t_x, t_y, t_z)$ denote a vector representing camera motion. Given camera motion and depth d, we transform the image coordinates (x, y) into world coordinates (X, Y, Z).

$$X = \frac{d}{f}(x - c_x) \tag{A.1}$$

$$Y = \frac{d}{f}(y - c_y) \tag{A.2}$$

$$Z = d \tag{A.3}$$

where (c_x, c_y, f) constitute the camera intrinsics. We now transform the world coordinates given the camera rotation and translation

$$\mathbf{X}' = R_x R_y R_z \mathbf{X} + t,$$

where $(R_x R_y R_z, t) \in SE3$ denote 3D rotation and translation, and $\mathbf{X} = [X, Y, Z]^T$. Hence, in image coordinates

$$x' = \frac{f}{Z} + c_x, \tag{A.4}$$

$$y' = \frac{f}{Z} + c_y. \tag{A.5}$$

We can now apply the warping as

$$w_c\big(I(x,y), e, d\big) = I(x', y'). \tag{A.6}$$

The static flow v can be then obtained using

$$v(o, d) = (x' - x, y' - y). \tag{A.7}$$

A.2 Warping an Image using Optical Flow

The flow warping function w_f warps the image using optical flow. It is given by

$$w_f\big(I(x,y),u_x,u_y\big) = I(x+u_x,y+u_y), \tag{A.8}$$

where (u_x,u_y) is the optical flow, (x,y) is the spatial coordinate system, and I is the image.

Appendix B

Proofs

B.1 Proof of Theorem 1

Proof. The update of x is a straight-forward gradient descent step on f. Using the Lipschitz bound on f, we get

$$\begin{aligned} f(x_{t+1},y_t) &\leq f(x_t,y_t) - \alpha\langle\nabla_x f(x_t,y_t),\nabla_x f(x_t,y_t)\rangle + \frac{G_1\alpha^2}{2}\|\nabla_x f(x_t,y_t)\|^2 \\ &= f(x_t,y_t) - \left(\alpha - \frac{G_1\alpha^2}{2}\right)\|\nabla_x f(x_t,y_t)\|^2 \\ &\leq f(x_t,y_t) - A\|\nabla_x f(x_t,y_t)\|^2 \end{aligned} \tag{B.1}$$

with $A > 0$ due to our assumption on α. For the update of y, we have

$$\begin{aligned} f(x_{t+1},y_{t+1}) \leq f(x_{t+1},y_t) &- \beta\langle\nabla_y f(x_{t+1},y_t),\nabla_y g(x_{t+1},y_t)\rangle \\ &+ \frac{\beta^2 G_2(x)}{2}\|\nabla_y g(x_{t+1},y_t)\|^2. \end{aligned} \tag{B.2}$$

Using the assumption on the inner product, this yields

$$f(x_{t+1},y_{t+1}) \leq f(x_{t+1},y_t) - B\|\nabla_y f(x_{t+1},y_t)\|^2. \tag{B.3}$$

Combining the two equations, we get

$$\begin{aligned} f(x_{t+1},y_{t+1}) &\leq f(x_t,y_t) - A\|\nabla_x f(x_t,y_t)\|^2 - B\|\nabla_y f(x_{t+1},y_t)\|^2 \\ &\leq f(x_t,y_t) - C\left(\|\nabla_x f(x_t,y_t)\|^2 + \|\nabla_y f(x_{t+1},y_t)\|^2\right). \end{aligned} \tag{B.4}$$

with $C = \max(A,B)$. We define $G_t = \|\nabla_x f(x_t,y_t)\|^2 + \|\nabla_y f(x_{t+1},y_t)\|^2$ and rewrite this as

$$G_t \leq \frac{f(x_t,y_t) - f(x_{t+1},y_{t+1})}{C} \tag{B.5}$$

Summing this equation for $t = 0, \ldots, T$, we get

$$\sum_{t=0}^{T} G_t \leq \frac{f(x_0, y_0) - f(x_{T+1}, y_{T+1})}{C}. \tag{B.6}$$

Since f is lower-bounded, this implies $G_t \to 0$, which in turn implies convergence to a first-order stationary point of f. □

B.2 Proof of Proposition 1

Proof. The gradient of f with respect to x is

$$\nabla_x f(x,y) = M(y)\nabla L_R(x) + (1 - M(y))\nabla L_F(x) \tag{B.7}$$

Since $M(y)$ is bounded, $\nabla_x f$ is Lipschitz continuous in x given that L_R and L_F are Lipschitz smooth.

For the assumptions on the y-gradients, we fix x and treat the two cases in the definition of g separately. We only consider the case $L_R(x) < L_F(x)$ here, the reverse case is completely analogous. Define $L(x) = L_F(x) - L_R(x) > 0$. The gradient of f with respect to y is

$$\nabla_y f(x,y) = -L(x)\nabla M(y) \tag{B.8}$$

and is Lipschitz continuous with constant $G_2(x) = |-L(x)|G = L(x)G$, where G is the Lipschitz constant of $M(y)$. We have

$$\nabla_y g(x,y) = -\left(L(x) + \frac{\lambda}{M(y) + \varepsilon}\right)\nabla M(y). \tag{B.9}$$

The inner product of the two gradients reads

$$\langle \nabla_y f(x,y), \nabla_y g(x,y) \rangle = L(x)\left(L(x) + \frac{\lambda}{M(y) + \varepsilon}\right)\|\nabla M(y)\|^2, \tag{B.10}$$

and for the gradient norms we get

$$\|\nabla_y f(x,y)\|^2 = L(x)^2 \|\nabla M(y)\|^2, \tag{B.11}$$

as well as

$$\|\nabla_y g(x,y)\|^2 = \left(L(x) + \frac{\lambda}{M(y) + \varepsilon}\right)^2 \|\nabla M(y)\|^2. \tag{B.12}$$

Plugging everything into the inner product assumption of Theorem 1 and simplifying

yields

$$\beta\left(L(x)+\frac{\lambda}{M(y)+\varepsilon}\right)\geq\frac{G\beta^2}{2}\left(L(x)+\frac{\lambda}{M(y)+\varepsilon}\right)^2+BL(x) \tag{B.13}$$

Since M, L_R and L_F are bounded, one easily finds a choice for β and B that satisfies this condition. □

Bibliography

[1] W. Abdulla. Mask R-CNN for object detection and instance segmentation on keras and tensorflow. `https://github.com/matterport/Mask_RCNN`, 2017.

[2] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.

[3] E. H. Adelson and J. R. Bergen. Spatiotemporal energy models for the perception of motion. *Journal of the Optical Society of America A*, 2(2), 1985.

[4] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, (4):384–401, 1985.

[5] A. Ahmadi and I. Patras. Unsupervised convolutional neural networks for motion estimation. In *International Conference on Image Processing (ICIP)*, pages 1629–1633, 2016.

[6] A. Amiranashvili, A. Dosovitskiy, V. Koltun, and T. Brox. Motion perception in reinforcement learning with dynamic objects. *arXiv preprint arXiv:1901.03162*, 2019.

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

[8] S. Ayer and H. S. Sawhney. Layered representation of motion video using robust maximum-likelihood estimation of mixture models and MDL encoding. In *International Conference on Computer Vision (ICCV)*, pages 777–784, 1995.

[9] C. Bailer, B. Taetz, and D. Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *International Conference on Computer Vision (ICCV)*, pages 4015–4023, 2015.

[10] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision (IJCV)*, 92(1):1–31, 2011.

[11] L. Bao, Q. Yang, and H. Jin. Fast edge-preserving PatchMatch for large displacement optical flow. *IEEE Transactions on Image Processing (ICIP)*, 23(12):4996–5006, 2014.

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

[13] M. J. Black and P. Anandan. A framework for the robust estimation of optical flow. In *International Conference on Computer Vision (ICCV)*, pages 231–236, 1993.

[14] M. J. Black and A. D. Jepson. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 18(10):972–986, 1996.

[15] M. J. Black, Y. Yacoob, A. D. Jepson, and D. J. Fleet. Learning parameterized models of image motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 561–567, 1997.

[16] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *International Conference on Computer Vision (ICCV)*, volume 2, pages 1395–1402, 2005.

[17] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *European Conference on Computer Vision (ECCV)*, 2016.

[18] R. T. Born and D. C. Bradley. Structure and function of visual area MT. *Annual Reviews of Neuroscience*, 28:157–189, 2005.

[19] T. Brox, C. Bregler, and J. Malik. Large displacement optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 41–48, 2009.

[20] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision (ECCV)*, pages 25–36, 2004.

[21] P. Burt and E. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on communications*, 31(4):532–540, 1983.

[22] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision (ECCV)*, pages 611–625, 2012.

[23] C. Cadieu and B. A. Olshausen. Learning transformational invariants from natural movies. In *Advances in Neural Information Processing System (NeurIPS)*, pages 209–216, 2008.

[24] J. Charles, T. Pfister, D. R. Magee, D. C. Hogg, and A. Zisserman. Personalizing human video pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3063–3072, 2016.

[25] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 40(4):834–848, 2017.

[26] M. Chessa, N. K. Medathati, G. S. Masson, F. Solari, and P. Kornprobst. Decoding MT motion response for optical flow estimation: An experimental evaluation. In *Signal Processing Conference (EUSIPCO), 2015 23rd European*, pages 2241–2245, 2015.

[27] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes dataset for semantic urban

scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[28] D. Cremers and S. Soatto. Motion competition: A variational approach to piecewise parametric motion segmentation. *International Journal of Computer Vision (IJCV)*, 62(3):249–265, 2005.

[29] D. Cudeiro, T. Bolkart, C. Laidlaw, A. Ranjan, and M. J. Black. Capture, learning, and synthesis of 3D speaking styles. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10101–10111, 2019.

[30] J. G. Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *Journal of Optical Society of America A*, 2(7):1160–1169, 1985.

[31] J. W. Davis. Hierarchical motion history images for recognizing human motion. In *Proceedings IEEE Workshop on Detection and Recognition of Events in Video*, pages 39–46, 2001.

[32] E. L. Denton, S. Chintala, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing System (NeurIPS)*, pages 1486–1494, 2015.

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

[34] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing System (NeurIPS)*, pages 2366–2374, 2014.

[35] R. Fablet and M. J. Black. Automatic detection and tracking of human motion with a view-based representation. In *European Conference on Computer Vision (ECCV)*, pages 476–491, 2002.

[36] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, 2016.

[37] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(9):1627–1645, 2009.

[38] K. Fragkiadaki, H. Hu, and J. Shi. Pose from flow and flow from pose. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2059–2066, 2013.

[39] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *International Journal of Computer Vision (IJCV)*, 40(1):25–47, 2000.

[40] A. Gaidon, Z. Harchaoui, and C. Schmid. Activity representation with motion hierarchies. *International Journal of Computer Vision (IJCV)*, 107(3):219–238, 2014.

[41] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4340–4349, 2016.

[42] D. M. Gavrila and L. S. Davis. 3-d model-based tracking of humans in action: a multi-view approach. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 73–80, 1996.

[43] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[44] D. Geman and S. Geman. Opinion: Science in the age of selfies. *Proceedings of the National Academy of Sciences*, 113(34):9384–9387, 2016.

[45] J. J. Gibson. *The perception of the visual world.* Houghton Mifflin, 1950.

[46] F. C. Glazer. *Hierarchical motion detection*. University of Massachusetts, 1987.

[47] C. Godard, O. Mac Aodha, and G. Brostow. Digging into self-supervised monocular depth estimation. *arXiv preprint arXiv:1806.01260*, 2018.

[48] R. Green. Spherical harmonic lighting: The gritty details. *Archives of the Game Developers Conference*, 2003.

[49] K. Greff, S. van Steenkiste, and J. Schmidhuber. Neural expectation maximization. In *Advances in Neural Information Processing System (NeurIPS)*, pages 6694–6704, 2017.

[50] F. Guney and A. Geiger. Displets: Resolving stereo ambiguities using object knowledge. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4165–4175, 2015.

[51] F. Güney and A. Geiger. Deep discrete flow. In *Asian Conference on Computer Vision (ACCV)*, 2016.

[52] Z. Han, Z. Xu, and S.-C. Zhu. Video primal sketch: A generic middle-level representation of video. In *International Conference on Computer Vision (ICCV)*, pages 1283–1290, 2011.

[53] R. Hartley. Multiview reconstruction of static and dynamic scenes. In *Digital Image Computing: Techniques and Applications: Proceedings of the VIIth Biennial Australian Pattern Recognition Society Conference, DICTA*, 2003.

[54] S. Hauberg, A. Feragen, R. Enficiaud, and M. J. Black. Scalable robust principal component analysis using Grassmann averages. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 38(11):2298–2311, 2015.

[55] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[56] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[57] D. J. Heeger. Model for the extraction of image flow. *Journal of the Optical Society of America A*, 4(8):1455–1471, 1987.

[58] B. Horn, B. Klaus, and P. Horn. *Robot vision*. MIT press, 1986.

[59] B. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, 1981.

[60] J. Hur and S. Roth. Iterative residual refinement for joint optical flow and occlusion estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5754–5763, 2019.

[61] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

[62] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3D human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 36(7):1325–1339, 2014.

[63] M. Jaderberg, K. Simonyan, and A. Zisserman. Spatial transformer networks. In *Advances in Neural Information Processing System (NeurIPS)*, pages 2017–2025, 2015.

[64] J. Janai, F. Güney, A. Ranjan, M. Black, and A. Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *European Conference on Computer Vision (ECCV)*, pages 690–706, 2018.

[65] J. Janai, F. Güney, J. Wulff, M. Black, and A. Geiger. Slow flow: Exploiting high-speed cameras for accurate and diverse optical flow reference data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[66] J. Y. Jason, A. W. Harley, and K. G. Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *European Conference on Computer Vision (ECCV)*, pages 3–10, 2016.

[67] A. Jepson and M. J. Black. Mixture models for optical flow computation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 760–761, 1993.

[68] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *International Conference on Computer Vision (ICCV)*, pages 3192–3199, Sydney, Australia, 2013.

[69] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9000–9008, 2018.

[70] G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception & Psychophysics*, 14(2):201–211, 1973.

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

[72] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing System (NeurIPS)*, pages 1097–1105, 2012.

[73] T. Kroeger, R. Timofte, D. Dai, and L. V. Gool. Fast optical flow using dense inverse search. In *European Conference on Computer Vision (ECCV)*, 2016.

[74] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *International Conference on Computer Vision (ICCV)*, pages 2556–2563, 2011.

[75] M. F. Land and D.-E. Nilsson. *Animal eyes*. Oxford University Press, 2012.

[76] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[77] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[78] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 38(10):2024–2039, 2016.

[79] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[80] M. Loper, N. Mahmood, and M. J. Black. Mosh: Motion and shape capture from sparse markers. *ACM Transactions on Graphics (TOG)*, 33(6):220, 2014.

[81] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, 2015.

[82] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 674–679, 1981.

[83] Q. Ma, S. Tang, S. Pujades, G. Pons-Moll, A. Ranjan, and M. J. Black. Dressing 3D humans using a conditional Mesh-VAE-GAN. *arXiv preprint arXiv:1907.13615*, 2019.

[84] R. Mahjourian, M. Wicke, and A. Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3D geometric constraints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5667–5675, 2018.

[85] E. J. Marey. Analyse cinématique de la marche [chronophotograph]. *CR Séances Acad. Sci*, 1884.

[86] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, 2016.

[87] S. Meister, J. Hur, and S. Roth. Unflow: Unsupervised learning of optical flow with a bidirectional census loss. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[88] R. Memisevic and G. E. Hinton. Learning to represent spatial transformations with factored higher-order Boltzmann machines. *Neural Computation*, 22(6):1473–1492, 2010.

[89] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3061–3070, 2015.

[90] A. Milan, L. Leal-Taixé, I. D. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016.

[91] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt. GANerated hands for real-time 3D hand tracking from monocular RGB. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[92] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, volume 2011, page 5, 2011.

[93] R. Nevatia and T. O. Binford. Structured descriptions of complex objects. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 641–647, 1973.

[94] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *International Conference on Computer Vision (ICCV)*, pages 2320–2327, Washington, DC, USA, 2011.

[95] A. Newell, K. Yang, and J. Deng. *Stacked Hourglass Networks for Human Pose Estimation*. 2016.

[96] D.-E. Nilsson. The evolution of eyes and visually guided behaviour. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 364(1531):2833–2847, 2009.

[97] B. A. Olshausen. Learning sparse, overcomplete representations of time-varying natural images. In *International Conference on Image Processing (ICIP)*, volume 1, pages I–41, 2003.

[98] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 3, 2017.

[99] T. Pfister, J. Charles, and A. Zisserman. Flowing convnets for human pose estimation in videos. In *International Conference on Computer Vision (ICCV)*, pages 1913–1921, 2015.

[100] G. Pons-Moll, J. Romero, N. Mahmood, and M. J. Black. Dyna: A model of dynamic human shape in motion. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 34(4):120:1–120:14, 2015.

[101] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool. The 2017 DAVIS challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

[102] A. Ranjan. How many numbers does it take to compute optical flow?, 2017. [https://medium.com/@anuragranj/how-many-numbers-does-it-take-to-compute-optical-flow-aa9545337b91; posted 1-June-2017].

[103] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

[104] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black. Generating 3D faces using convolutional mesh autoencoders. In *European Conference on Computer Vision (ECCV)*, pages 704–720, 2018.

[105] A. Ranjan, D. T. Hoffmann, D. Tzionas, S. Tang, J. Romero, and M. J. Black. Learning multi-human optical flow. *arXiv preprint arXiv:1910.11667*, 2019.

[106] A. Ranjan, V. Jampani, L. Balles, K. Kim, D. Sun, J. Wulff, and M. J. Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[107] A. Ranjan, J. Janai, A. Geiger, and M. J. Black. Attacking optical flow. In *International Conference on Computer Vision (ICCV)*, 2019.

[108] A. Ranjan, J. Romero, and M. J. Black. Learning human optical flow. In *British Machine Vision Conference (BMVC)*, 2018.

[109] B. Ray and A. Ranjan. Unsupervised video segmentation, 2019. US Patent App. 15/849,341.

[110] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[111] K. M. Robinette, S. Blackwell, H. Daanen, M. Boehmer, and S. Fleming. Civilian American and European surface anthropometry resource (CAESAR), final report. volume 1. summary. Technical report, DTIC Document, 2002.

[112] J. Romero, M. Loper, and M. J. Black. FlowCap: 2D human pose from optical flow. In *German Conference on Pattern Recognition (GCPR)*, volume LNCS 9358, pages 412–423, 2015.

[113] J. Romero, D. Tzionas, and M. J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), 2017.

[114] D. Rosenbaum, D. Zoran, and Y. Weiss. Learning the local statistics of optical flow. In *Advances in Neural Information Processing System (NeurIPS)*, pages 2373–2381, 2013.

[115] S. Roth and M. J. Black. Fields of experts. *International Journal of Computer Vision (IJCV)*, 82(2):205–229, 2009.

[116] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[117] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *Advances in Neural Information Processing System (NeurIPS)*, pages 1161–1168, 2006.

[118] L. Sevilla-Lara, Y. Liao, F. Güney, V. Jampani, A. Geiger, and M. J. Black. On the integration of optical flow and action recognition. In *German Conference on Pattern Recognition (GCPR)*, pages 281–297, 2018.

[119] L. Sevilla-Lara, D. Sun, V. Jampani, and M. J. Black. Optical flow with semantic segmentation and localized layers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[120] L. Sevilla-Lara, D. Sun, E. G. Learned-Miller, and M. J. Black. Optical flow estimation with channel constancy. In *European Conference on Computer Vision (ECCV)*, volume 8689, pages 423–438, 2014.

[121] L. Sigal, A. O. Balan, and M. J. Black. HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision (IJCV)*, 87(1-2):4, 2010.

[122] E. P. Simoncelli and D. J. Heeger. A model of neuronal responses in visual area MT. *Vision Research*, 38(5):743–761, 1998.

[123] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing System (NeurIPS)*, pages 568–576, 2014.

[124] F. Solari, M. Chessa, N. K. Medathati, and P. Kornprobst. What can we expect from a V1-MT feedforward architecture for optical flow estimation? *Signal Processing: Image Communication*, 39:342–354, 2015.

[125] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[126] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2432–2439, 2010.

[127] D. Sun, S. Roth, and M. J. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision (IJCV)*, 106(2):115–137, 2014.

[128] D. Sun, S. Roth, J. Lewis, and M. J. Black. Learning optical flow. In *European Conference on Computer Vision (ECCV)*, pages 83–97, 2008.

[129] D. Sun, E. B. Sudderth, and M. J. Black. Layered image motion with explicit occlusions, temporal consistency, and depth ordering. In *Advances in Neural Information Processing System (NeurIPS)*, pages 2226–2234, 2010.

[130] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[131] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by GPU-accelerated large displacement optical flow. In *European Conference on Computer Vision (ECCV)*, pages 438–451, 2010.

[132] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[133] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *European Conference on Computer Vision (ECCV)*, pages 140–153, 2010.

[134] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino. Collision detection for deformable objects. In *The Eurographics Association*, pages 119–139, 2004.

[135] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Deep End2End Voxel2Voxel prediction. In *3rd Workshop on Deep Learning in Computer Vision*, 2016.

[136] D. Tzionas, L. Ballan, A. Srikantha, P. Aponte, M. Pollefeys, and J. Gall. Capturing hands in action using discriminative salient points and physics simulation. *International Journal of Computer Vision (IJCV)*, 118(2):172–193, 2016.

[137] S. Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426, 1979.

[138] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 5, 2017.

[139] J. H. van Hateren and D. L. Ruderman. Independent component analysis of natural image sequences yields spatio-temporal filters similar to simple cells in primary visual cortex. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 265(1412):2315–2320, 1998.

[140] G. Varol, J. Romero, X. Martin, N. Mahmood, M. Black, I. Laptev, and C. Schmid. Learning from synthetic humans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[141] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. In *International Conference on Computer Vision (ICCV)*, volume 2, pages 722–729, 1999.

[142] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. SfM-Net: learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017.

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

[144] Y. Wang and G. Mori. Hidden part models for human action recognition: Probabilistic versus max margin. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(7):1310–1323, 2010.

[145] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing (TIP)*, 13(4):600–612, 2004.

[146] A. M. Waxman and J. H. Duncan. Binocular image flows: Steps toward stereo-motion fusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, (6):715–729, 1986.

[147] A. Wedel, T. Pock, C. Zach, H. Bischof, and D. Cremers. An improved algorithm for TV-L1 optical flow. In *Statistical and Geometrical Approaches to Visual Motion Analysis*, pages 23–45. 2009.

[148] D. Weinland, R. Ronfard, and E. Boyer. Motion history volumes for free viewpoint action recognition. In *Workshop on Modeling People and Human Interaction (PHI)*, 2005.

[149] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *International Conference on Computer Vision (ICCV)*, pages 1385–1392, 2013.

[150] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 optical flow. In *British Machine Vision Conference (BMVC)*, London, UK, September 2009.

[151] J. Wulff and M. J. Black. Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 120–130, 2015.

[152] J. Wulff and M. J. Black. Temporal interpolation as an unsupervised pretraining task for optical flow estimation. In *German Conference on Pattern Recognition (GCPR)*, 2018.

[153] J. Wulff, L. Sevilla-Lara, and M. J. Black. Optical flow in mostly rigid scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[154] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3485–3492, 2010.

[155] A. Yilmaz and M. Shah. Actions sketch: A novel action representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 984–989, 2005.

[156] Z. Yin and J. Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. *arXiv preprint arXiv:1803.02276*, 2018.

[157] G.-S. Young and R. Chellappa. 3-d motion estimation using a sequence of noisy stereo images: Models, estimation, and uniqueness results. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 12(8):735–759, 1990.

[158] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[159] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 7, 2017.

[160] C. Zimmermann and T. Brox. Learning to estimate 3D hand pose from single RGB images. In *International Conference on Computer Vision (ICCV)*, 2017.

[161] Y. Zou, Z. Luo, and J.-B. Huang. DF-Net: Unsupervised joint learning of depth and flow using cross-task consistency. In *European Conference on Computer Vision (ECCV)*, pages 38–55, 2018.

[162] S. Zuffi, J. Romero, C. Schmid, and M. J. Black. Estimating human pose with flowing puppets. In *International Conference on Computer Vision (ICCV)*, pages 3312–3319, 2013.