# Data-driven and Sparse-to-Dense Concepts in Scene Flow Estimation for Automotive Applications

**Dissertation**

vom Fachbereich Informatik der Technischen Universität Kaiserslautern zur Verleihung des akademischen Grades Doktor der Ingenieurwissenschaften (Dr.-Ing.) genehmigte Dissertation

von

*René Schuster*


| | |
|---|---|
| Datum der Einreichung | 10.05.2021 |
| Datum der wissenschaftlichen Aussprache | 18.03.2022 |
| Dekan | Prof. Dr. Jens Schmitt |
| Gutachter | Prof. Dr. Didier Stricker |
| | Prof. Dr.-Ing. Andrés Bruhn |

**DE-386**

## Abstract

Highly assisted driving and autonomous vehicles require a detailed and accurate perception of the environment. This includes the perception of the 3D geometry of the scene and the 3D motion of other road users. The estimation of both based on images is known as the scene flow problem in computer vision. This thesis deals with a solution to the scene flow problem that is suitable for application in autonomous vehicles. This application imposes strict requirements on accuracy, robustness, and speed. Previous work was lagging behind in at least one of these metrics.

To work towards the fulfillment of those requirements, the sparse-to-dense concept for scene flow estimation is introduced in this thesis. The idea can be summarized as follows: First, scene flow is estimated for some points of the scene for which this can be done comparatively easily and reliably. Then, an interpolation is performed to obtain a dense estimate for the entire scene. Because of the separation into two steps, each part can be optimized individually. In a series of experiments, it is shown that the proposed methods achieve competitive results and are preferable to previous techniques in some aspects.

As a second contribution, individual components in the sparse-to-dense pipeline are replaced by deep learning modules. These are a highly localized and highly accurate feature descriptor to represent pixels for dense matching, and a network for robust and generic sparse-to-dense interpolation. Compared to end-to-end architectures, the advantage of deep modules is that they can be trained more efficiently with data from different domains.

The recombination approach applies a similar concept as the sparse-to-dense approach by solving and combining less difficult, auxiliary sub-problems. 3D geometry and 2D motion are estimated separately, the individual results are combined, and then also interpolated into a dense scene flow.

As a final contribution, the thesis proposes a set of monolithic end-to-end networks for scene flow estimation.

# Kurzfassung

Hochassistiertes Fahren und autonome Fahrzeuge erfordern eine detaillierte und genaue Wahrnehmung der Umgebung. Dazu gehört auch die Wahrnehmung der 3D-Geometrie der Szene und der 3D-Bewegung anderer Verkehrsteilnehmer. Die Schätzung von beidem auf der Basis eines optischen Sensors ist in der Computer Vision als Szenenflussproblem bekannt. Diese Arbeit befasst sich mit einer Lösung des Szenenflussproblems, die für den Einsatz in autonomen Fahrzeugen geeignet ist. Diese Anwendung stellt hohe Anforderungen an die Genauigkeit, Robustheit und Geschwindigkeit. Bisherige Arbeiten hinken in mindestens einem dieser Punkte hinterher.

Um auf die Erfüllung dieser Anforderungen hinzuarbeiten, wird in dieser Arbeit das „Sparse-to-Dense“-Konzept für die Schätzung des Szenenflusses eingeführt. Die Idee lässt sich wie folgt zusammenfassen: Zunächst wird der Szenenfluss für einige Punkte der Szene geschätzt, für die dies vergleichsweise einfach und zuverlässig möglich ist. Dann wird eine Interpolation durchgeführt, um eine dichte Schätzung für die gesamte Szene zu erhalten. Durch die Aufteilung in zwei Schritte kann jeder Teil einzeln optimiert werden. In einer Reihe von Experimenten wird gezeigt, dass die vorgeschlagenen Methoden konkurrenzfähige Ergebnisse erzielen und gegenüber bisherigen Techniken in einigen Aspekten vorzuziehen sind.

Als zweiter Beitrag werden einzelne Komponenten in der „Sparse-to-Dense“-Pipeline durch Deep Learning Module ersetzt. Dabei handelt es sich um einen hoch lokalisierten und hochpräzisen Deskriptor zur Repräsentation von Pixeln, und um ein Netzwerk für eine robuste und generische „Sparse-to-Dense“-Interpolation. Im Vergleich zu End-to-End-Architekturen haben einzelne Lern-Module den Vorteil, dass sie auf eine breitere Menge an Trainingsdaten zurückgreifen können.

Der Rekombinationsansatz wendet ein ähnliches Konzept wie der „Sparse-to-Dense“-Ansatz an, indem weniger schwierige Teilprobleme gelöst und kombiniert werden. 3D-Geometrie und 2D-Bewegung werden separat geschätzt, die einzelnen Ergebnisse werden kombiniert, und dann ebenfalls zu einem dichten Szenenfluss interpoliert.

Als letzter Beitrag wird in dieser Arbeit eine Reihe von monolithischen End-to-End Netzwerken für die Schätzung des Szenenflusses vorgeschlagen.

# Danksagung

Viele Unterstützer haben dazu beigetragen, dass diese Worte an dieser Stelle möglich sind.

Mein Dank gilt meinem Betreuer Prof. Dr. Didier Stricker, der mich in allen Situationen gefördert und gefordert hat. Prof. Dr. Oliver Wasenmüller bin ich für sein Vertrauen und die hervorragende Führung dankbar. Durch die etlichen Gespräche mit ihm habe ich nicht nur meine Forschung strukturieren können, sondern auch einen Freund gefunden. Ich bedanke mich auch bei den zahlreichen Kollegen, die zu einem fruchtbaren Arbeitsklima und durch ihre Zusammenarbeit auch zur Erreichung meiner Ziele beigetragen haben. Meinen Projektpartnern bei BMW, insbesondere Georg Kuschk und Christian Unger, ein großes Dankeschön für ihre aufmunternde Sachlichkeit, das unersättliche Interesse an dem erforschten Thema, und die vielen wegweisenden Anregungen.

Nicht zuletzt gebührt ein Großteil der Anerkennung meiner Familie. Danke an meine Geschwister, die mir aus nah und fern das Studium versüßt haben, und vor allem an meinen Bruder, dem nachzueifern schon immer die größte Motivation war. Meinen Eltern – insbesondere meinem Vater – bin ich zutiefst dankbar für die bedingungslose Unterstützung und alle Freiheiten und Möglichkeiten während meines gesamten Lebens.

Dir, liebe Sara, gilt mein abschließender Dank. Du begleitest mich seit Beginn dieser Reise und hast mich immer auf alle erdenklichen Arten erquickt.

März 2022, René Schuster

# Contents

Chapter 1

# Introduction

*"'Begin at the beginning', the King said, very gravely, 'and go on till you come to the end: then stop.'"*

— Lewis Carroll, *Alice in Wonderland*

*What will define the future of transportation?* While no one is able to answer this question with absolute certainty, a majority of people would think of autonomous driving, intelligent vehicles, or artificial intelligence. It is indeed striking, that the number of Advanced Driver Assistance Systems (ADASs) in series is growing. Simultaneously, the user acceptance for assisting and autonomous systems is reaching a higher level. The advantages of such technologies are manifold: More fluent traffic and more efficient load on traffic infrastructure, accelerated transition from private vehicles to shared or public transportation systems, increased comfort for drivers or passengers to reduce stress and contribute to a healthy mental state, and most importantly safety aspects for passengers and other traffic participants, including the most vulnerable road users – pedestrians and cyclists. Worldwide, over 1.3 million people die due to fatal traffic accidents per year [WHO18]. Experts predict that fully automated vehicles could reduce the total number of accidents and especially those with injury to persons drastically [NHTSA17]. Therefore, the development towards higher assistance and automation levels for more intelligent vehicles becomes a valuable goal for society, governments, and researchers.

Typical high-level tasks in this field comprise navigation, longitudinal and lateral control of the vehicle, and more, which all rely on an accurate perception and understanding of the environment. A system's perception is enabled by a variety of sensors. In computer vision, the most relevant optical sensor in automotive applications are cameras. Next to semantic scene understanding, localization, and others, two core components of the perception are a detailed reconstruction of the 3D geometry of the surroundings as well as a precise estimation of the motion of other traffic participants. Both can be estimated based on image information from cameras, which in computer vision is known as the scene flow problem. Scene flow describes the perceived 3D motion field

for all visible points of the scene. As such it is a dense (with respect to the camera resolution) and detailed representation of the real world.

### Challenges

Chapter 2 describes how scene flow estimation in computer vision narrows down to finding corresponding pixels within multiple images. However, solving such dense correspondence problems is non-trivial for two main reasons:

1. Representation: The appearance of the same world point within an image differs for different viewpoints.

2. Occlusion: Not all points visible in one image, are visible in the other images.

These two points are repeatedly addressed in this thesis.

In the context of automotive applications, both challenges are inherently amplified. Changing weather, seasonal, and daytime conditions increase the variation in image data, environmental influences can induce abrupt changes in lighting, the distances of visible content cover a long range, scenes can be cluttered and very dynamic with possibly many fast moving objects and high ego-velocities. On top, independent of the issues above, an exhaustive comparison of image points is prohibitive even for medium image resolutions.

Furthermore, since the possible applications in intelligent vehicles are potentially safety-critical and require real-time performance, strict requirements on run time, accuracy, reliability, and robustness are induced.

### Motivation for the Sparse-to-Dense Concept

With respect to the aforementioned requirements and challenges, the sparse-to-dense concept promises positive impact. It allows to simplify the correspondence search (matching) because challenging regions do not require special mechanisms in the first place. Instead, these regions can be matched purposely wrong in order to be removed later. In a second step, the non-dense, reliable information is used to interpolate the missing information. The simplification enables more efficient optimization during matching, introducing a potential boost in run time. Within the interpolation, common assumptions, which are used in previous work for expensive regularization, can be modeled in a more principled way. These properties of the sparse-to-dense concept have been validated for the estimation of optical flow. Due to the similarity of the two problems, the concept is a qualified option to be used for the estimation of scene flow as well. The restrictions of previous work, compared to the sparse-to-dense concept, are discussed in Chapter 3.

### Deep Vision: Computer Vision in the Deep Learning Age

Besides, deep learning is overwhelming every field of technology including computer vision. Neural networks have matured to a point where design and training become such efficient that their predictions outperform those of

many previous, heuristic techniques. In computer vision, deep convolutional neural networks even surpass the performance of humans considering certain basic tasks. All this provided, sufficient and suitable data for training is available. The inherent complexity of the scene flow problem makes it difficult to measure reference labels or to annotate scene flow data with ground truth labels. Therefore only a very limited amount of data exists that can be used to train scene flow estimators in a supervised fashion. Likewise, deep neural networks for scene flow estimation have evolved only recently during the course of this thesis.

## 1.1. Research Statement and Contributions

Hence, there are two central questions this thesis tries to answer:

1. To which extent and how can the established sparse-to-dense concept for optical flow be transferred to the scene flow problem?

2. What are suitable methods to transfer the success of data-driven deep neural networks to the scene flow problem, despite the lack of rich annotated data?

The first question is addressed in Chapters 4 to 6, the second is dealt with in Chapters 5 to 7. The following summary shows how each chapter contributes to the thesis and which publications back the respective content.

### Chapter 4

The sparse-to-dense concept for scene flow estimation is introduced. Best practices from matching in optical flow are transferred to multiple images to obtain sparse correspondences. Different models for the sparse-to-dense interpolation are presented. Multi-frame matching is proposed. At WACV 2018, the sparse-to-dense concept achieved the third best results on KITTI and the best on Sintel among competing dual-frame approaches.

**René Schuster**, Oliver Wasenmüller, Georg Kuschk, Christian Bailer, and Didier Stricker. "SceneFlowFields: Dense Interpolation of Sparse Scene Flow Correspondences." In: *Winter Conference on Applications of Computer Vision (WACV)*. 2018.

**René Schuster**, Oliver Wasenmüller, Christian Unger, Georg Kuschk, and Didier Stricker. "SceneFlowFields++: Multi-frame Matching, Visibility Prediction, and Robust Interpolation for Scene Flow Estimation." In: *International Journal on Computer Vision (IJCV)* (2020).

### Chapter 5

Two components of the sparse-to-dense pipeline are replaced by deep modules. A universal dense feature descriptor for pixel matching is proposed. A dedicated, very robust network is used for sparse-to-dense interpolation. The proposed descriptor improves stereo, optical flow, and scene flow matching for five algorithms on six diverse data sets. The interpolation network is significantly more robust in terms of sparse and noisy input, compared to previous

interpolation techniques for optical flow, scene flow, and depth completion.

**René Schuster**, Oliver Wasenmüller, Christian Unger, and Didier Stricker. "SDC – Stacked Dilated Convolution: A Unified Descriptor Network for Dense Matching Tasks." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019. **Oral**.

**René Schuster**, Oliver Wasenmüller, Christian Unger, and Didier Stricker. "An Empirical Evaluation Study on the Training of SDC Features for Dense Pixel Matching." In: *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019.

**René Schuster**, Oliver Wasenmüller, Christian Unger, and Didier Stricker. "SSGP: Sparse Spatial Guided Propagation for Robust and Generic Interpolation." In: *Winter Conference on Applications of Computer Vision (WACV)*. 2021.

## Chapter 6

By applying the sparse-to-dense concept, a dense scene flow estimation is enabled by combining results of stereo and optical flow estimation. This principle is also transferred and evaluated in a monocular camera setup. On KITTI, the monocular, sparse-to-dense, combination approach obtains the highest accuracy for dynamic objects and the overall best results among methods with sub-second run time.

**René Schuster**, Christian Bailer, Oliver Wasenmüller, and Didier Stricker. "Combining Stereo Disparity and Optical Flow for Basic Scene Flow." In: *Commercial Vehicle Technology Symposium (CVT)*. 2018.

**René Schuster**, Oliver Wasenmüller, and Didier Stricker. "Dense Scene Flow from Stereo Disparity and Optical Flow." Extended Abstract for the Computer Science in Cars Symposium (CSCS). 2018.

**René Schuster**, Christian Unger, and Didier Stricker. "MonoComb: A Sparse-to-Dense Combination Approach for Monocular Scene Flow." In: *Computer Science in Cars Symposium (CSCS)*. 2020.

## Chapter 7

For comparison, one of the first end-to-end trainable deep neural networks for scene flow estimation is presented. A second contribution focuses on improved localization for feature extractors in end-to-end trainable dense matching networks. Finally, (end-to-end) deep learning is applied to frame existing dual-frame methods in a multi-frame setting to further reduce the impact of occlusions. Until this time, there is no faster approach listed on KITTI, that performs better than the proposed end-to-end network. The generic multi-frame extension achieves a close second place among this category of algorithms and is ~400 times faster than the first place.

Rohan Saxena, **René Schuster**, Oliver Wasenmüller, and Didier Stricker. "PWOC-3D: Deep Occlusion-Aware End-to-End Scene Flow Estimation." In: *Intelligent Vehicles Symposium (IV)*. 2019. **Oral**.

Rishav*, **René Schuster***, Ramy Battrawy, Oliver Wasenmüller, and Didier Stricker. "ResFPN: Residual Skip Connections in Multi-Resolution Feature Pyramid Networks for Accurate Dense Pixel Matching." In: *International Conference on Pattern Recognition*

*(ICPR)*. 2021. *Equal contribution. **Oral**.

**René Schuster**, Christian Unger, and Didier Stricker. "A Deep Temporal Fusion Framework for Scene Flow Using a Learnable Motion Model and Occlusions." In: *Winter Conference on Applications of Computer Vision (WACV)*. 2021.

Chapter 2

# Preliminaries

*"Get the fundamentals down and the level of everything you do will rise."*
— Michael Jordan, *I Can't Accept Not Trying*

## Contents



## 2.1. Mathematical Notation

To ease the understanding of mathematical formulations, a common notation is defined within this section. Images are denoted by $I$ with superscript for the viewpoint and the subscript as time step unless unambiguous. A set of images is given by $\mathbf{I} = \{I_0, ..., I_T^v\}$. The reference time is $t = 0$ unless stated otherwise. The image domain is $\Omega$, i.e. the range of discrete pixel positions inside the image. Pixels (positions) are two-dimensional vectors $\mathbf{p} = (x, y)^T \in \Omega$ and represent the projected position of three-dimensional world points $\mathbf{P} = (X, Y, Z)^T$. Small and capitalized letters do not refer to vectors or matrices, but rather relate the image and world domain, however not exclusively. Depth maps versus disparity maps are denoted by $D$ and $d$, for instance. Normal and bold letters refer to scalars or scalar fields and vectors or vector fields, respectively. To

resolve ambiguities, when a value for a scalar or vector field is denoted by the same symbol as the value for single pixels, the multi-dimensional value is given in normal font not in italic, e.g. $\mathrm{d}_1(\mathbf{p}) = d_1$. Here, $\mathrm{d}_1$ is a disparity map and $d_1$ is the disparity value at pixel $\mathbf{p}$. Where necessary, further ambiguity can be resolved by indexing with a certain pixel position, e.g. $I(\mathbf{p}) = (r, g, b)^T$ is the RGB value of a single pixel.

## 2.2. Fundamentals of Computer Vision

A profound understanding of computer vision is assumed in this thesis. However for completeness, two fundamental concepts are outlined in the following. The first is a formulation which models how images (and their representation as an array of pixels) are related to the real world. The second describes how depth can be inferred from two stereo images.

### 2.2.1. The Pinhole Camera Model

Modern real cameras are a complicated composition of sensors, optics, and other components. A pragmatic simplification of this interplay is the pinhole camera model, that describes how the 3D world is projected onto an image plane assuming an ideal pinhole camera. The name derives from a small hole (the aperture) through which the reflected (or emitted) light of objects hits the image plane (cf. Figure 2.1).

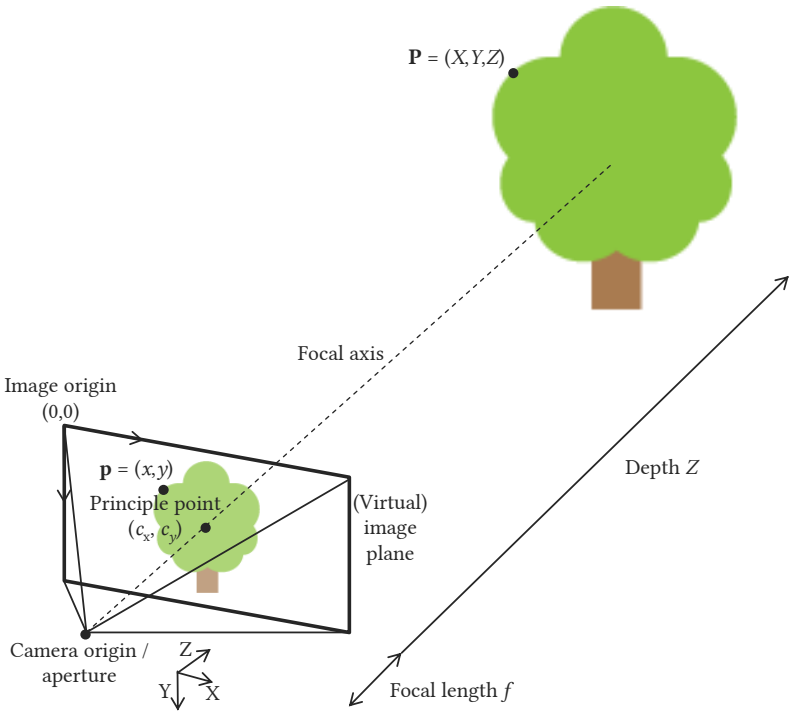



**Figure 2.1.:** *A sketch of the pinhole camera model*

Assume a camera located in the 3D world with an associated coordinate system of which the orientation aligns with a default image coordinate system plus an additional depth dimension in the direction of viewing. The camera position is located at the origin of this camera coordinate system. Light that passes through the aperture is projected onto the image plane. In actual pinhole cameras, the imaged scene is rotated. In practice, one would manually rotate a photograph, or read out a digital image in such a way that it becomes rotated by 180°. In order to practically implement this fact in the mathematical formulation, a virtual image plane in front of the aperture is assumed. This is achieved by inverting the focal distance $f$ from the camera center (aperture) to the image plane, resulting in the desired outcome. Finally, since images in computer science have their origin pixel at the top left corner, the image needs to be shifted by the offset of the principle point $\mathbf{p}_c = (c_x, c_y)^T$, i.e. where the focal axis intersects the image plane.

Given this setting, the intersect theorem leads to the following equations that relate a world point $\mathbf{P} = (X, Y, Z)^T$ to a pixel $\mathbf{p} = (x, y)^T$:

$$x = \frac{f}{Z} X + c_x \tag{2.1}$$

$$y = \frac{f}{Z} Y + c_y \tag{2.2}$$

In matrix notation with homogeneous coordinates:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.3}$$

The process of estimating these internal camera parameters (also known as camera intrinsics) is called camera calibration.

A more generic model considers an arbitrarily positioned and oriented camera and extends Equation 2.3 by a rotation and translation from the world coordinate system to the camera coordinate system. Since for this work this step is not of importance, it is omitted. A camera at a reference time step is always aligned with a hypothetical world coordinate system as depicted in Figure 2.1. However, it is noted that automotive applications typically use a standard car coordinate system which is different from the one above, but can easily be converted by rotation.

Further, an ideal pinhole camera does not model all physical effects, like e.g. lens distortions. However in high quality cameras, these effects can be corrected or become negligibly small, making the pinhole camera model sufficiently accurate for computer vision.

However, given the image alone, the 3D position of an imaged point can not be inferred due to the depth ambiguity. I.e. all points on the ray from camera origin through a pixel are projected to this pixel. Without knowing the associated depth of this point, the pinhole projection can not be inverted. This leads to the next section.

### 2.2.2. Epipolar Geometry for Rectified Stereo Image Pairs

A stereo camera mimics the binocular human visual system. It captures two images simultaneously from two viewpoints with similar cameras, which ideally only differ by a horizontal displacement. This way, the 3D position of two projections of the same world point can be reconstructed by triangulation [HZ03]. The challenge here lies in finding the corresponding points. Thanks to the epipolar geometry of stereo image pairs, the corresponding pixel location of a pixel in one view, lies on the respective epipolar line which always intersects the epipole, i.e. the intersection of the image plane and a line between both camera origins. This restricts the search space for pixel correspondences.

A further simplification of the problem can be achieved by using rectified images. Two rectified images are co-planar, letting the epipoles approach infinity, and rendering the epipolar lines to be all parallel to the line connecting the camera origins. Additionally, the rectification ensures that corresponding points have the same vertical image coordinate.

Consequentially, rectification enforces that the relative position of the two camera origins $O_1$ and $O_2$ are described by a horizontal translation only, i.e. in homogeneous coordinates

$$O_2 = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} O_1 = \begin{pmatrix} 1 & 0 & 0 & B \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} O_1. \tag{2.4}$$

This displacement is called the *baseline* $B$ of the stereo camera.

Undistorted, rectified stereo image pairs from calibrated cameras are a well-founded basis for the reconstruction of depth in computer vision.

Stereo reconstruction is a well-known approach in computer vision to estimate the depth of a scene. Depth estimation wants to recover the shortest distance between the observed scene and the (infinite) image plane for each pixel. The geometric constraints for rectified stereo image pairs as described before provide the following formulation of depth, given a known pixel correspondence between pixel $\mathbf{p}$ in the left view and pixel $\mathbf{p}'$ in the right view (cf. Figure 2.2):

$$Z = \frac{fB}{d} \tag{2.5}$$

Here, $B$ is the baseline between the stereo cameras and $f$ the focal length. $d = x - x'$ is the disparity, i.e. the displacement of the horizontal pixel coordinate between the left and right view of corresponding points as illustrated in Figure 2.2. Note that disparity values are positive by convention, though the pixel in the right view is always located further to the left within the image domain.

A stereo algorithm tries to find the best correspondences for all pixels of one image so that a consistent depth map is created. By doing that, the result of a stereo algorithm is a disparity map which assigns a positive disparity value to every pixel of the image.

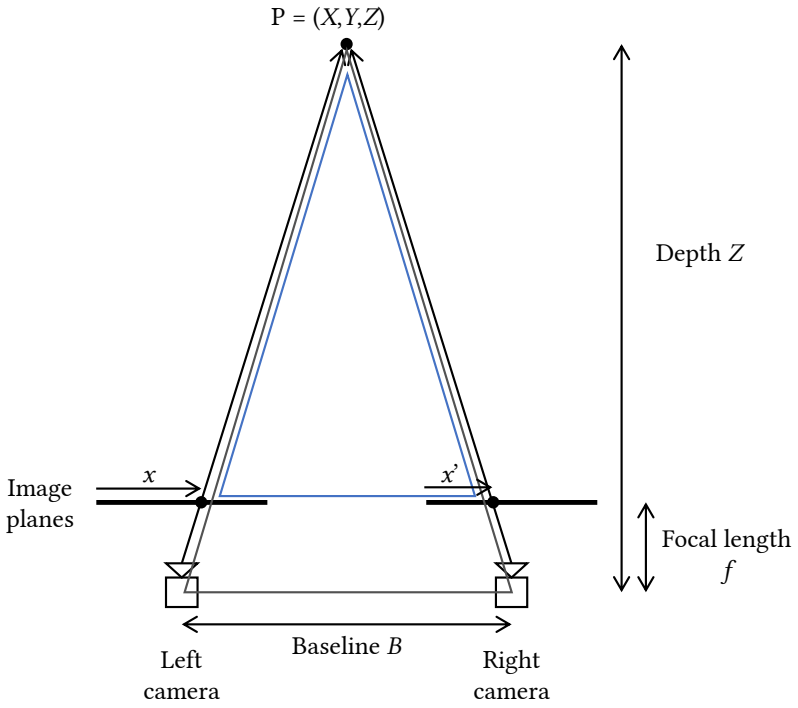



**Figure 2.2.:** *A flat illustration of the 3D reconstruction in rectified stereo images.*

### 2.2.3. Optical Flow

Optical flow is the estimation of a 2D displacement field $\boldsymbol{u}$ between two images. These displacements describe the apparent transition $\boldsymbol{u}(\mathbf{p}) = (u, v)^T$ of one pixel to another position in image space over time. As such, optical flow describes all visible changes, i.e. the superposition of ego-motion of the camera and motion of individual objects. The main purpose of optical flow is to provide a proxy for motion in a dynamic scene, especially in the context of automotive vehicles. Again, the problem is to find matches across two images. But for optical flow the search space is much bigger, as the epipolar geometry does not restrict it considering dynamic environments. Even more than for stereo images, the representation of image points (descriptors) is crucial for successful matching, since corresponding points may appear considerably different in the two views due to the temporal offset. This property, which is especially violated in the automotive scenario, builds the theoretical foundation for optical flow estimation and is called the Brightness Constancy Assumption (BCA) [HS81; LK81].

$$I_t(x, y) = I_{t+\Delta t}(x + u, y + v) \tag{2.6}$$

However in general, this equation holds for $\Delta t \to 0$ only.

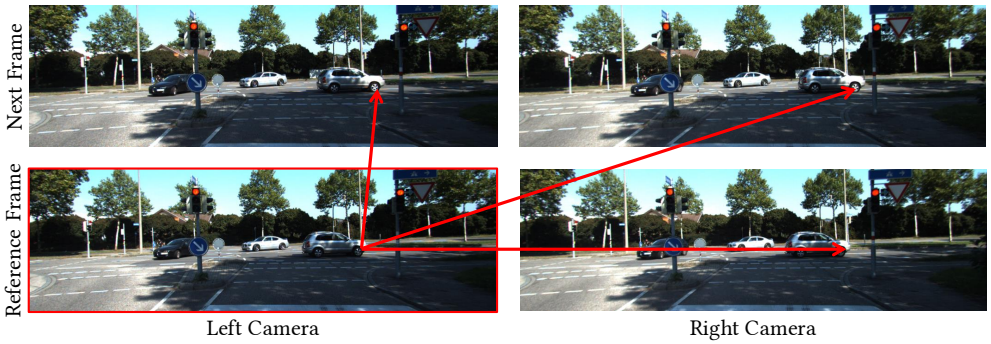



**Figure 2.3.:** *Corresponding points in two calibrated and rectified stereo frame pairs define a 3D scene flow vector.*

## 2.3. Scene Flow

Scene flow describes the apparent motion of points in 3D world space. It is often considered as the extension of optical flow into 3D space using stereo images, or the temporal extension of stereo depth estimation and its change over time. In this respect it is very related to the two previously mentioned problems in computer vision. In fact, in large parts it can be modeled as the joint solution of stereo and optical flow estimation.

For scene flow algorithms in this thesis, it is assumed to have the typical stereo image information provided, i.e. two rectified stereo image pairs ($\mathbf{I} = \{I_0^l, I_1^l, I_0^r, I_1^r\}$) at times $t_0$ and $t_1$ along with the camera intrinsics. It is further assumed that the baseline $B$ is known and constant. Using the fundamentals in Section 2.2, full 6D scene flow in world space (3D position + 3D motion) can be derived if corresponding pixels across all four images can be found.

As a result, the scene flow of a single pixel is represented as a 4D vector $\boldsymbol{s}(\mathbf{p}) = (u, v, d_0, d_1)^T$ consisting of the two optical flow components $u$, $v$ and the disparity values $d_0$, $d_1$ for both time steps. This formulation relates pixels in the four images according to Equation 2.7 as follows:

$$\begin{aligned} & I_0^l(x,y) \\ \approx & I_0^r(x-d_0,y) \\ \approx & I_1^l(x+u,y+v) \\ \approx & I_1^r(x+u-d_1,y+v). \end{aligned} \tag{2.7}$$

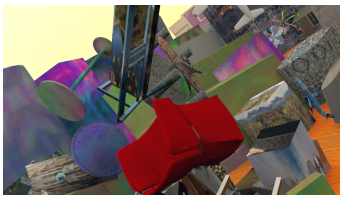
The formulation is comparable to the one in [HD07] and further explained by the illustration in Figure 2.3. Note that $d_1$ does not provide the disparity values between $I_1^l$ and $I_1^r$, since it describes the future disparity for the visible pixels in image $I_0^l$.

### 2.3.1. Data Sets

Data sets serve two main purposes. They allow evaluation and comparison of algorithms in a specific scenario, and they provide the basis for data-

**(a)** *KITTI [GLU12].*

**(b)** *Sintel [BWSB12].*



**(c)** *FT3D [MIHF+16].*

**Figure 2.4.:** *Examples of images from different data sets.*

driven (supervised) learning approaches. In both cases, the data set has to be annotated with ground truth labels of the respective task.

Since it is hard to capture or manually label ground truth scene flow, there exist only very few data sets to evaluate scene flow algorithms on. Most of them use virtually rendered scenes to obtain the ground truth data [BWSB12; CMH20; MIHF+16]. To the best of the author's knowledge the only realistic data set providing a benchmark for scene flow is the KITTI Vision Benchmark [GLU12], which combines various tasks for automotive vision. Its introduction has played an important role in the development of stereo and optical flow algorithms, and the extension by [MG15] has also driven progress in scene flow estimation.

The KITTI data set was captured with a stereo camera mounted on a vehicle. The data set consists of highway, rural, and urban traffic scenes. 200 sequences (image quadruples) have been annotated with reference labels for scene flow. This was achieved by using a GPS, IMU and a LiDAR scanner as a reference sensor for precise localization and depth measurement. The motion of dynamic vehicles has been reconstructed by manually fitting CAD car models to the images and point clouds and computing rigid transformations. All other dynamic objects have been excluded from this data set. As a result, the reference labels are non-dense and the static background is furthermore covered sparsely (with the resolution of the LiDAR scanner).

Another densely annotated, yet synthetic, data set for scene flow is the FlyingThings3D (FT3D) subset along with the Monkaa and Driving subsets presented in [MIHF+16]. Especially when it comes to data-driven learning approaches, this data set provides a vital source for supervision. It consists of rendered scenes where random 3D objects fly in front of a random background image.

Lastly, the MPI Sintel data set [BWSB12] is mentioned here. It does not

provide full scene flow labels, but is very popular in the field of optical flow due to its challenging scenes. Beyond that, it offers stereo images and depth labels, which can be used for a partial evaluation of scene flow.

Figure 2.4 presents examples of images for these data sets. An overview of these and more related data sets for other dense correspondence problems is given in Section 5.1.5.

Only recently, the Virtual KITTI 2 data set [CMH20] has been released, which provides full scene flow labels in traffic scenarios. However, the scenes look very similar to the KITTI data set but less realistic, since they are synthetically rendered. It is not used in this thesis and only mentioned for completeness.

### 2.3.2. Evaluation Metrics

Since KITTI is the most relevant and main data set used in this thesis, the evaluation is oriented towards what is used in the KITTI online benchmark. There, the evaluation criteria is the KITTI outlier error rate (KOE). The KOE gives the percentage of pixels that exceed an error of 3 px and deviate more than 5 % from the ground truth. It is computed separately for optical flow and the disparity values. The errors are computed as Euclidean difference in 2D and 1D, respectively. An outlier for scene flow is defined if either one of the estimated disparity maps or the optical flow contains an outlier.

Further in this thesis, the end-point error (EPE) in image space is computed. For scalar disparity maps, the EPE is the pixel-wise average absolute difference between the estimate and the ground truth. For two-dimensional optical flow, the EPE is the average Euclidean distance. The EPE for scene flow is defined as the sum of the EPEs for optical flow and the two disparity maps. In some explicitly mentioned cases, the EPE for scene flow is computed as average Euclidean distance of the four-dimensional scene flow representation in image space (cf. Section 2.3).

A final remark is given with respect to the indexing of time on the KITTI benchmark [GLU12; MG15]. While throughout this thesis, the reference time is $t = 0$ and thus the initial disparity in the scene flow representation is $\mathrm{d}_0$, KITTI starts indexing at 1. Therefore, evaluations with respect to the KITTI online benchmark denote the KOE for the initial and future disparity as *D1* and *D2*, respectively.

### 2.3.3. Visualization

In conformity with the evaluation metrics, the visualization of scene flow is also split into the optical flow field and the initial and future disparity maps. Optical flow is visualized according to the Middlebury optical flow color wheel [BSLR+11], which encodes the 2D direction of the flow by the hue of the color and its magnitude by the saturation (cf. Figure 2.5), i.e. no apparent motion is encoded in white. Disparity – and likewise depth – is visualized using the jet color map from red (close by) to blue (far away). In some cases, disparity (or depth) is used to re-project points as a 3D point cloud which is then observed

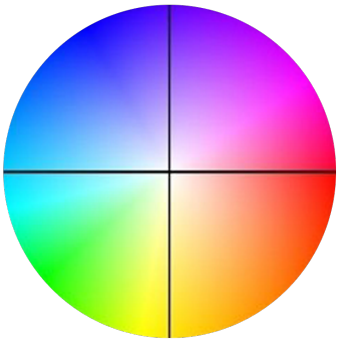


**Figure 2.5.:** *The Middlebury color wheel for the visualization of optical flow.*

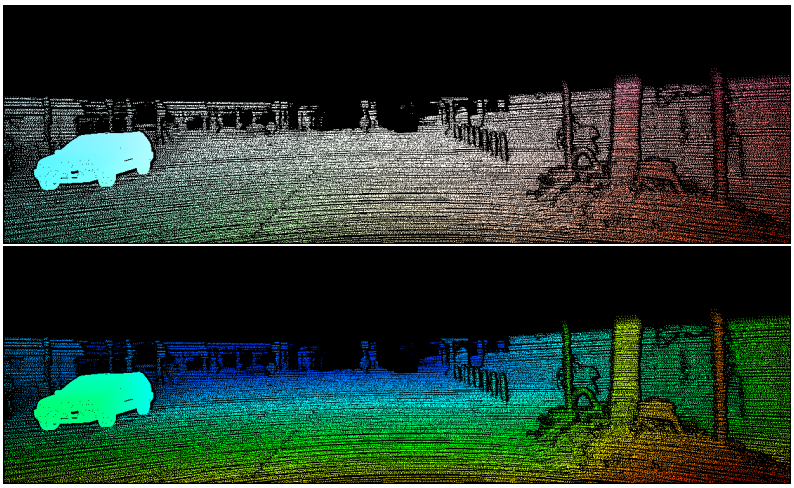


**Figure 2.6.:** *Visualization of the color-coded sparse ground truth for optical flow and disparity of a sample from the KITTI data set.*

and visualized from a different viewpoint (other than the original camera). This way, an easy-to-perceive qualitative impression of the reconstruction of the geometry is conveyed. These two colorization schemes are shown in Figure 2.6 for the ground truth of a sample from the KITTI data set.

For the visualization of errors (i.e. the difference between predicted and actual scene flow), a binary outlier map is used that indicates inliers in green and outliers in blue to magenta, according to the definition in Section 2.3.2.

For the visual comparison of methods that have been submitted to the KITTI online benchmark [GLU12; MG15], the original colorization of the benchmark for results and error maps are used. Error maps on the benchmark show the EPE on a piece-wise quadratic color map (blue to white for inliers, orange to dark red for outliers, cf. e.g. Figure 4.14).

Chapter 3

# An Overview of Methodologies in Scene Flow Estimation

*Nani gigantum umeris insidentes.*

— Bernard of Chartres

## Contents



This chapter discusses previous and related work in the field of scene flow estimation. At first, different sensors which allow for an estimation of scene flow are presented (see Figure 3.1 and Table 3.1). Within this part, the focus of this thesis on cameras – and in particular stereo cameras – is motivated. In a second step, common and popular models for the estimation of scene flow are described, along with algorithmic concepts (cf. Table 3.2). The choice of sensor(s), model, and algorithm is often tightly coupled. Lastly, the most related approaches in the literature are listed in Table 3.3, and categorized according to the previously introduced properties. Most of these methods

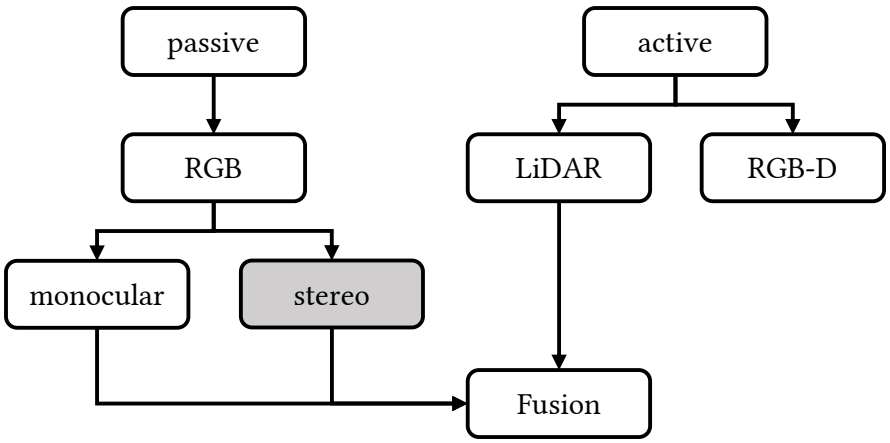



**Figure 3.1.:** *An ontology of visual sensor modalities for scene flow estimation.*

frequently represent state-of-the-art competing to the approaches presented in this thesis.

## 3.1. Sensor Modalities

### 3.1.1. Cameras

The method that coined the term scene flow was presented in [VBRC+99]. This method posed scene flow estimation in a multi-view setting captured with a monocular camera. To introduce a more principled way for 3D reconstruction, several scene flow algorithms followed, which used stereo cameras as sensor device. Until this point, stereo cameras are the most widespread sensor for scene flow estimation and can be considered the default. Only recently, a trend towards using less sensors has emerged again, introducing another series of monocular approaches [BAM19; HR20; YR20]. Here, depth is recovered from a single image using deep learning to estimate monocular scene flow. However, even though deep neural networks are able to solve this task to some extend, the overall performance lags behind stereo methods. On the upside, a single camera is cheaper and does not require calibration or rectification. Note that a camera does not necessarily has to capture RGB data. A monochrome camera can be used as well.

### 3.1.2. Depth Cameras

Direct measurement of depth simplifies the problem of scene flow estimation greatly, because the geometry of the scene does not need to be reconstructed from image information. On the other hand, active sensing of depth introduces other limitations. Depending on the sensor type (e.g. structured light), ambient effects may deteriorate the quality of the depth measurements. In all cases, the range of depth cameras is limited, and the resolution is often lower compared to a plain camera. These properties make depth cameras unsuitable for

**Table 3.1.:** *Comparison of visual sensors for scene flow estimation.*

| | Sensor | Advantages | Disadvantages |
|---|---|---|---|
| passive | Monocular Camera | Minimalistic setup, no calibration, cheap | Weak depth cues |
| | Stereo Camera | Principled | Depth resolution, calibration, rectification |
| active | RGB-D Camera | Direct depth measurement | Low range, sensitive to environment |
| | LiDAR | High precision | Low density, mostly geometric sensing, expensive |

automotive applications, which require depth sensing up to 100 meters in outdoor environments with strong and diverse light sources. However, for indoor environments this sensor setup is used. A list of notable methods includes [HFR14; HRF13; JSGC15; QBDC14]. The terms depth camera and RGB-D camera are used interchangeably in this thesis.

### 3.1.3. LiDAR Scanners

Similarly to depth cameras, laser scanners (or Light Detection and Ranging (LiDAR) scanners) provide a way to measure distances directly. In comparison, LiDAR scanners are less sensitive to environmental lighting and even more precise, covering also a much longer range. Yet, LiDAR devices still suffer from limitations, like e.g. (semi) transparent surfaces. The most important drawback of a laser scanner is its density along with the price. Especially high-resolution scanners are too expensive for series production in vehicles and they still capture only a fraction of the density of cameras. In the context of scene flow estimation, LiDAR sensors are used in two ways. Firstly they are used for the fusion with a camera, either to support the stereo setup [BSWR+19], or to replace it and instead use a monocular camera and obtain reliable 3D reconstruction from the (densified) LiDAR [RBSW+20]. Secondly, a use as the only sensor to estimate scene flow from consecutively captured point clouds is possible. This approach has again emerged recently and is strongly powered by deep neural networks [BPDG19; GWWL+19; LQG19; MOH20; WWLL+20].

## 3.2. Models and Algorithmic Categories

### 3.2.1. Variational Optimization

Vedula et al. [VBRC+99] were among the first to compute 3D scene flow. Afterwards, many variational approaches for scene flow estimation followed. A similar variational approach using multiple images from a stereo camera was presented in [PKF05; PKF07]. These approaches were inspired by similar

concepts for optical flow estimation [HS81]. First, pure color images [BMK13; FRRR+14; HD07; POJT+12; VBZW+10; WRVB+08] and later RGB-D images [HFR14; HRF13; JSGC15; QBDC14; ZCYZ12] were used as input. While a variational formulation is typically complex, Jaimez et al. [JSGC15] achieved real-time performance with a primal-dual framework. Yet, all these approaches are sensitive to initialization and can not cope with large displacements, which is why they use a coarse-to-fine scheme. This in turn tends to miss finer details. Nowadays, variational methods are outperformed in terms of speed and accuracy by other approaches and are only used as a refinement step.

### 3.2.2. Rigid Planes

Due to the advent of a piece-wise rigid plane model [VSR13], scene flow has recently achieved a boost in performance. The majority of top performing methods at the KITTI benchmark employ this model to enforce strong regularization [BJMA+17; LBAL+16; LML21; MG15; MWHX+19; NŠ17; VSR15]. Vogel et al. [VRS14; VSR15] encode this model by an alternating assignment of each pixel to a plane segment and each segment to a rigid motion, based on a discrete set of planes and motions in view-consistent manner over multiple frames. The complexity of the model is further lowered by the assumption that a scene consists of very few independently moving, rigid objects in [BJMA+17; MG15; NŠ17]. Thus, each plane segment only needs to be assigned to one object. All segments assigned to the same object share the same motion. By propagation of objects over multiple frames, temporal consistency for the model of [MG15] is achieved by [NŠ17]. In [BJMA+17], deep learning is used to obtain semantic object information a-priori. The pixel-to-plane assignment and the plane-to-motion assignment is solved in a continuous domain in [LBAL+16]. Auxiliary Deep Neural Networks (DNNs) are employed in [BJMA+17; MWHX+19] to obtain an accurate initialization for the rigid, planar model before it is further optimized. Despite the remarkable accuracy on KITTI, many of these methods are not applicable to domains with different characteristics. The rigid motion assumption is strongly violated by articulated gestures and other non-rigid motions that often occur in the Sintel data set [BWSB12]. The assumption made by [BJMA+17; MG15; NŠ17], that there are only a few independent dynamic objects in a scene, is inappropriate for highly dynamic scenarios. Further, methods falling into this category typically have very long run times of several minutes up to almost one hour per frame.

### 3.2.3. Divide and Conquer

**Static-Dynamic Decomposition.** Yet another promising strategy builds on the decomposition of a scene into static and moving parts [TSS17]. While the motion of dynamic objects is estimated by solving a discrete labeling problem (as in [CK16]) using the Semi-Global Matching (SGM) algorithm [Hir08], the perceived motion of all static parts is directly obtained from the 3D geometry of the scene and the ego-motion of the camera. This approach is especially convenient for scenes, where only a small proportion consists of moving objects,

like it is usually the case in traffic scenarios. However, any a-priori assumption limits the versatility of a method. A rigid plane model performs poorly when applied to deformable objects, and ego-motion estimation for highly dynamic scenes is difficult.

**Combination Approach.** Because the scene flow problem is highly related to the auxiliary tasks of optical flow and stereo disparity estimation, it is possible to estimate scene flow by combining separate results for optical flow and stereo disparity [MIHF+16]. Though the separation brings advantages for the complexity of the problem and thus the run time, it is believed that a single formulation of the problem yields more consistent scene flow results. Also due to occlusions, such a separation is known to yield non-dense result. A solution to this issue is the sparse-to-dense concept. Therefore, this combination approach is investigated in Chapter 6 of this thesis.

**Sensor Conversion.** The transformation of the input into a different, virtual one, is another special case of the separation of scene flow estimation into smaller, related sub-problems. Stereo images e.g. can be used to estimate dense depth in a first step, followed by applying a RGB-D method. This was done on the KITTI data set by combining SGM [Hir08] with SphereFlow [HFR14]. A stereo result can also be used to create a high density point cloud to run scene flow estimators that usually work with input from a LiDAR scanner (if these methods were capable of processing the resolution of a camera). However, it is likely that the virtual input is more noisy compared to an actual measurement.

### 3.2.4. Deep Learning

**Deep Neural Networks.** Shortly after the success of the first end-to-end networks for optical flow estimation [DFIH+15; IMSK+17], deep learning approaches also started to take over scene flow estimation. This development happened in large part in parallel to the research of this thesis. Therefore, one of the very first DNNs for scene flow estimation is presented within the thesis in Chapter 7. In the meantime, other approaches followed [APTM20; ISKB18; MIHF+16; MWHX+19]. The massive parallelization on Graphics Processing Units (GPUs) introduces a remarkable speed-up in scene flow estimation.

**Semantic Segmentation.** Other scene flow algorithms use deep learning to incorporate semantic information into the motion estimation problem [RSKS17]. Yet, in terms of robustness, deep learning approaches typically lag behind because they generalize insufficiently to unseen data and even less to data from different domains [WD18]. This is especially true for semantic segmentation, where the domain gap is amplified by the mismatch of semantic classes between domains [LLYL+18]. Semantic segmentation (and instance segmentation) in the field of scene flow estimation is used in different ways: As strong segmentation for a static-dynamic decomposition [RSKS17], together with the

piece-wise rigid planes model [BJMA+17; LML21; MWHX+19], or in a joint multi-task formulation of the problem [JSJL+19].

### 3.2.5. Multi-frame Stereo Approaches

Finally, one has to differentiate between dual-frame [LBAL+16; MG15; VSR15] and multi-frame [NŠ17; TSS17; VSR15] approaches. Especially in the context of automotive applications, several characteristics make matching between two frame pairs much more challenging than in a multi-frame setting. These characteristics are: 1.) Considerably large stereo and flow displacements. 2.) Difficult lighting conditions and many reflective and (semi-)transparent surfaces of cars. 3.) Fast ego-motions sometimes combined with low to medium frame rates, which causes large regions to move out of the field of view. Therefore, pairs of dual-frame and multi-frame methods exist. The transitions from OSF [MG15] to OSF+TC [NŠ17] and from PRSF [VSR13] to PRSM [VRS14; VSR15] have brought essential improvements by using the additional information from multiple stereo frames. However, this additional information comes at a cost. The relationship between multiple temporal steps needs to be modeled to make use of the additional images. A typical model is to assume smooth, constant motion between neighboring time steps [NŠ17; VSR15].

### 3.2.6. Sparse-to-Dense

The sparse-to-dense approach for scene flow estimation is introduced in this thesis (see Chapter 4). Consequentially, previous work in the field of scene flow estimation does not exist. Instead, this section discusses the concept for optical flow estimation to highlight the properties, challenges, and advantages of this approach. For optical flow, the sparse-to-dense concept was successfully realized by EPICFlow [RWHS15] with competitive accuracy. Recall that the main idea is to separate the estimation of dense flow into two steps: 1.) Obtaining sparse correspondences across images and 2.) Interpolating the sparse result into a dense one. The biggest advantage during the first step is that matching can focus on easy image areas to obtain most reliable and accurate results in an efficient manner. One challenge in this step is, that it is unclear in the first place, which areas or pixels are easy to match and important during interpolation. Another advantage of the second step is, that the interpolation can tackle typical challenges in flow estimation with effective models (e.g. occlusions, homogeneous areas, etc.). To obtain a similar effect in previous approaches, complex regularization by assuming local smoothness is necessary. Since this assumption is modeled globally (e.g. in variational optimization), such methods tend to blur discontinuities and raise the run time. Many works have improved the concept since its first successes. The matching used in EPICFlow was later refined by FlowFields [BTS15] and its many derivatives [BTS19; BVS17; WKR17] or by competitors like CPM [HSL16]. The interpolation itself was also developed further by RICFlow [HLS17], SemFlow [WZLY+19], or InterpoNet [ZW17], in case of the latter two with the aid of deep neural networks. Some

**Table 3.2.:** *Overview of different algorithmic categories of scene flow algorithms.*

| Category | Advantages | Disadvantages |
|---|---|---|
| Variational | | Slow, inaccurate, only indoors |
| Static-Dynamic Decomposition | Potentially fast | Ego-motion dependency, inconsistent |
| Pice-wise Rigid Planes Model | Strong regularization | Slow |
| Deep Learning | Fast | Poor generalization |
| Sparse-to-Dense | Comparatively fast, good generalization | Sensitive to distribution of matches |

work even improved the post-processing steps in a sparse to dense pipeline [MSB17].

The lasting interest of researchers in this category of algorithms indicates its importance. However, there is a sensitive interplay between the two steps. If matching yields too few correspondences, gaps for interpolation grow inappropriately big. If too many matches are returned, run time scales unfavorably, or even worse, the accuracy of the sparse results deteriorate. Consequently, sparse-to-dense approaches require a careful tuning to find a sweet spot where both parts perform adequately.

## 3.3. Chronicle of Most Relevant Scene Flow Methods

Table 3.3 lists relevant methods in chronological order. Methods presented in this thesis are not included. In general, but especially in the field of automotive driving, a majority of approaches utilizes stereo cameras as a sensor, with a very recent rise of pure LiDAR approaches. Since 2019, there has been a clear trend towards the utilization of DNNs, and away from heuristic methods, in particular away from variational techniques since 2015. Apart from the early beginning, multi-frame methods have only been actively investigated between 2015 and 2017. To reactivate research in this direction, two novel multi-frame approaches are presented in this thesis (cf. Sections 4.2 and 7.3). A similar pattern can be observed for monocular approaches. After the method of Vedula et al. [VBRC+99], monocular methods have only been proposed since the rise of DNNs, as well as in Section 6.2 of this work.

**Table 3.3.:** *A non-exhaustive, chronological list of scene flow methods. The list does not cover the offspring of this thesis.*

| Year | Reference | Sensor | Frames | Category |
|---|---|---|---|---|
| 1999 | [VBRC+99] | Mono | Multi | Variational |
| 2005 | [PKF05] | Stereo | Multi | Variational |
| 2007 | [HD07] | Stereo | Dual | Variational |
| 2008 | [WRVB+08] | RGB-D | Dual | Variational |
| 2013 | [HRF13] | RGB-D | Dual | Variational |
| 2013 | [VSR13] | Stereo | Dual | Rigid Planes |
| 2014 | [QBDC14] | RGB-D | Dual | Variational |
| 2014 | [HFR14] | RGB-D | Dual | Variational |
| 2015 | [VSR15] | Stereo | Multi | Rigid Planes |
| 2015 | [JSGC15] | RGB-D | Dual | Variational |
| 2015 | [MG15] | Stereo | Dual | Rigid Objects |
| 2016 | [LBAL+16] | Stereo | Dual | Rigid Planes |
| 2017 | [TSS17] | Stereo | Multi | Decomposition |
| 2017 | [RSKS17] | Stereo | Dual | Rigid Objects |
| 2017 | [BJMA+17] | Stereo | Dual | Rigid Instances |
| 2019 | [JSJL+19] | Stereo | Dual | DNN |
| 2019 | [LQG19] | LiDAR | Dual | DNN |
| 2019 | [MWHX+19] | Stereo | Dual | DNN+PRSM |
| 2019 | [BPDG19] | LiDAR | Dual | DNN |
| 2019 | [BAM19] | Mono | Dual | DNN+PRSM+CRF |
| 2020 | [APTM20] | Stereo | Dual | DNN |
| 2020 | [HR20] | Mono | Dual | DNN |
| 2020 | [YR20] | Stereo | Dual | DNN |
| 2020 | [YR20] | Mono | Dual | DNN |
| 2020 | [WWLL+20] | LiDAR | Dual | DNN |
| 2021 | [LML21] | Stereo | Dual | DNN+CRF |

Chapter 4

# Sparse-to-Dense Scene Flow Estimation

*The whole is other than the sum of its parts.*
— Aristotle

## Contents



This chapter is backed by the following publications:

**René Schuster**, Oliver Wasenmüller, Georg Kuschk, Christian Bailer, and Didier Stricker. "SceneFlowFields: Dense Interpolation of Sparse Scene Flow Correspondences." In: *Winter Conference on Applications of Computer Vision (WACV)*. 2018.

**René Schuster**, Oliver Wasenmüller, Christian Unger, Georg Kuschk, and Didier Stricker. "SceneFlowFields++: Multi-frame Matching, Visibility Prediction, and Robust Interpolation

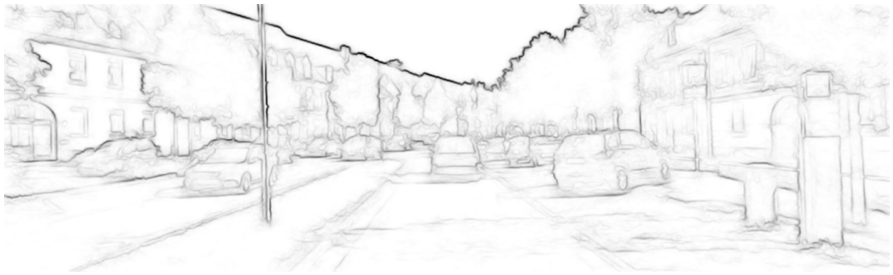
for Scene Flow Estimation." In: *International Journal on Computer Vision (IJCV)* (2020).

The various advantages of sparse-to-dense estimation for dense displacement fields, as described in Section 3.2, motivate to transfer the concept to the estimation of scene flow. This chapter outlines the general idea and presents a realization of the same based on well-established best practices in optical flow estimation.

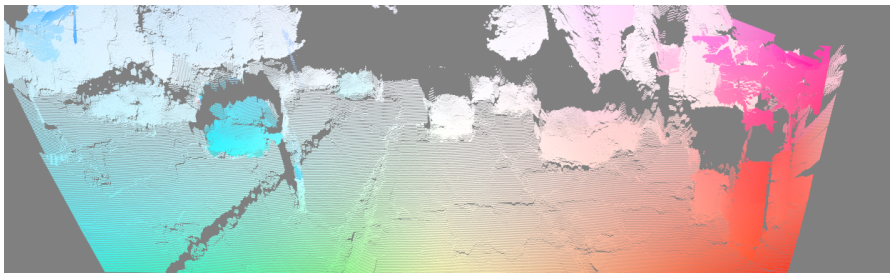
There are different techniques to handle the non-matchable parts of a scene, i.e. cases where it is difficult or even impossible to find correspondences in the relevant images. Typically, some kind of regularization is applied, like in form of a smoothness assumption which encodes that neighboring pixels should represent a similar motion so that local visual evidence can support the motion estimation in the difficult areas. In methods that employ the piece-wise rigid plane model [BJMA+17; LBAL+16; MG15; NŠ17; VSR13; VSR15] this kind of regularization is for two reasons considerably strong. Firstly, each local patch describing a slanted plane undergoes the same transformation by design. Secondly, inter-plane smoothness is further enforced by dedicated terms in the energy formulation. However, regularization terms increase the computational effort significantly (cf. Table 4.5), and would prohibit the use of the efficient optimization strategy of Section 4.1.1. An alternate concept to handle non-matchable regions is sparse-to-dense interpolation. This idea is rather young and was first successfully realized by EPICFlow [RWHS15] for the optical flow problem. The idea is to remove regions of low confidence (i.e. regions where regularization would be required to match them accurately) and to use interpolation to fill the gaps based on reliable matches. SceneFlowFields (SFF) is to the best of the author's knowledge the first method of sparse-to-dense interpolation for scene flow correspondences and the extension in Section 4.2 is transferring this concept to a multi-frame setup for the first time.

**Discrimination from State-of-the-Art.** Related work can be clustered into certain categories (cf. Table 3.2). Because the proposed scene flow methods follow the newly introduced sparse-to-dense approach, it differs from any of the related approaches. First, sparse scene flow matches are found which are then interpolated to a dense scene flow field, recovering the geometry of the scene and the 3D motion. These methods have to be distinguished from purely variational approaches. Although variational optimization is used, it can be considered as a post-processing step for refinement. Similar, the cameras ego-motion is only used to refine the results for static image regions. During interpolation, the geometry of a scene is modeled by very small planar segments, but there is no initial coarse segmentation presumed. In fact, the very small size of the plane segments leads to smoothly curved shapes and sharp boundaries. The same holds for the piece-wise motion model that is used to interpolate the 3D motion. Methods which are guided by semantic segmentation from deep neural networks generalize badly to other domains, unless they are fine-tuned for the new task. Same is assumed for upcoming purely learning based approaches

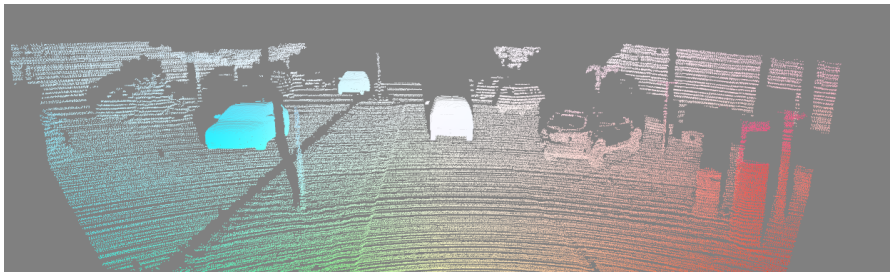
**(a)** *Reference image.*



**(b)** *Scene flow boundaries.*



**(c)** *Estimated scene flow.*



**(d)** *Sparse ground truth.*

**Figure 4.1.:** *Based on stereo image pairs and corresponding boundaries, SFF estimates a dense 3D motion field. The color of the point clouds encodes the optical flow.*

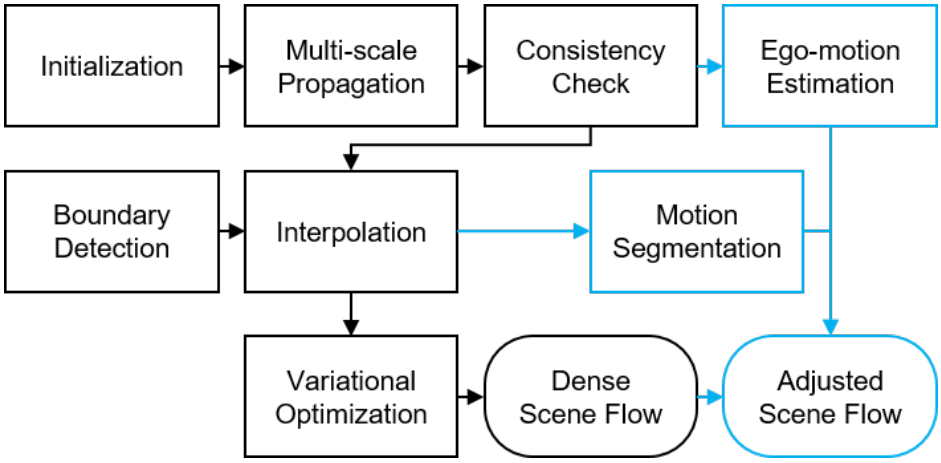



**Figure 4.2.:** *Overview of the pipeline of SFF. Blue color indicates the optional ego-motion extension.*

[APTM20; JSJL+19; MWHX+19] which are potentially even faster than the sparse-to-dense approach. SceneFlowFields++ (SFF++) focuses especially on robustness across domains and applications.

The different algorithmic categories are contrasted and compared in Table 3.2. The piece-wise rigid planes model is particularly accurate due to its strong regularization (as long as the assumptions are not violated), but is also complex and computationally expensive. The decomposition (separation) approach that splits the scene flow problem into less difficult sub-problems is especially fast and benefits from advances in the auxiliary tasks. Yet, separate computation leads to overall less consistent scene flow. Deep learning is potentially fast due to the inherent parallelization on GPUs, but sensitive to the distribution of available training data and not interpretable in case of failure. The novel scene flow concept of sparse-to-dense interpolation allows to separate matching from regularization. With the use of appropriate interpolation models and interpolation regions, the negative impact of violated assumptions can be diminished. However, the separation of matching and regularization makes the sparse-to-sense approach sensitive to the quality of the sparse matching results. To overcome this issue, a multi-frame extension is proposed within this thesis.

## 4.1. Dense Interpolation of Sparse Scene Flow Correspondences

In the concrete implementation sparse matches are obtained by performing dense coarse-to-fine matching and filtering with a forward-backward consistency check. Given the formulation of Section 2.3, dense (noisy) scene flow is estimated as follows: For $k$ subscales the coarsest scale is initialized by finding the best correspondences from kD-trees built with feature vectors using the Walsh-Hadamard-Transform (WHT) [HH05]. For all $k+1$ scales (the $k$ subscales

plus full resolution) scene flow vectors are iteratively propagated and adjusted by random search. Afterwards, the dense scene flow map on full resolution is filtered using an inverse scene flow field and a region filter. The filtered scene flow map is further thinned out by only taking the best match in each non-overlapping $3 \times 3$ block. Scene flow boundaries are detected using a structured random forest. Geometry and 3D motion are separately interpolated based on a boundary-aware neighborhood. Finally, the 3D motion is refined by variational optimization and optionally by estimating the ego-motion of the vehicle. An overview of SFF is outlined in Figure 4.2.

### 4.1.1. Sparse Correspondences

**Matching Cost.** An important aspect in sparse-to-dense matching is that the matching cost depends on a data term solely. No additional smoothness assumptions are made like e.g. in [HD07; HRF13; LBAL+16; MG15; VSR13; VSR15]. In this case given a scene flow vector, its matching cost is defined by the sum of Euclidean distances between SIFTFlow features [LYT11] over small image patches [BTS15]. The matching error for two corresponding pixels $\mathbf{p}$ and $\mathbf{p}'$ in images $I$ and $I'$ is defined by the following cost

$$C\left(I, \mathbf{p}, I', \mathbf{p}'\right) = \sum_{\tilde{\mathbf{p}} \in W(\mathbf{p})} \left\| \phi\left(I, \tilde{\mathbf{p}}\right) - \phi\left(I', \mathbf{p}' + \tilde{\mathbf{p}} - \mathbf{p}\right) \right\|, \tag{4.1}$$

whereas $W(\mathbf{p})$ is a $7 \times 7$ patch window centered at pixel $\mathbf{p}$ and $\phi(I, \mathbf{p})$ is a function that returns the first three principal components of a SIFT feature vector (SIFTFlow) for pixel $\mathbf{p}$ in image $I$. The principal axes are computed for the combined SIFT features [Low99] of all four images. At image boundaries, the boundary pixel is replicated to pad the images. The cost for three image correspondences are evaluated. These correspondences are the stereo image pair at time $t$, the temporal image pair for the left viewpoint (standard optical flow correspondence) and a cross correspondence between the reference frame and the right frame at the next time step (cf. Figure 4.2). This leads to the following overall cost $C$ for a scene flow vector $\boldsymbol{s} = (u, v, d_0, d_1)^T$ at pixel $\mathbf{p}$:

$$\begin{aligned} C(\mathbf{p}, \boldsymbol{s}) = {} & C\left(I_0^l, \mathbf{p}, I_1^l, \mathbf{p} + (u, v)^T\right) \\ & + C\left(I_0^l, \mathbf{p}, I_0^r, \mathbf{p} + (-d_0, 0)^T\right) \\ & + C\left(I_0^l, \mathbf{p}, I_1^r, \mathbf{p} + (u - d_1, v)^T\right). \end{aligned} \tag{4.2}$$

Thereby, a dense scene flow field $\mathbf{s}$ can be obtained by optimizing the following energy minimization problem

$$E(\mathbf{I}, \mathbf{s}) = \sum_{\mathbf{p} \in \Omega} C(\mathbf{p}, \mathbf{s}(\mathbf{p})), \tag{4.3}$$

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}} \; E(\mathbf{I}, \mathbf{s}). \tag{4.4}$$

Though this optimization includes a lot of variables, the fact that $\boldsymbol{s} = (u, v, d_0, d_1)^T$ can be optimized for each pixel individually is exploitable. This

is possible since the formulation includes no inter-pixel dependencies, e.g. no explicit regularization. Therefore, an efficient, greedy, stochastic optimization approach of propagation and random search can be used. While additional terms for correspondences between the remaining images are possible (e.g. optical flow for the right camera), this would lead to two implications. Firstly, more correspondences are less computationally efficient. Secondly, the additional cost would enforce higher precision where all views are non-occluded, but would introduce higher errors if the 3D point was occluded in a single viewpoint. This in turn would lead to less dense matches after the consistency check. The matching cost in Equation 4.2, using three image correspondences, realizes a tradeoff between precision and density.

**Initialization.** Initialization is based on kD-trees similar to [HS12], but with three trees, using WHT features as in [BTS15; WKR17]. For each frame other than the reference frame, one feature vector per pixel is computed and stored in a tree. To initialize a pixel of the reference image, the feature vector of that pixel is compared to the pre-computed kD-trees. Scene flow matches are then obtained by comparing all combinations of the leafs for each queried node according to the matching data term introduced before (Equation 4.1). Since the stereo image pairs are rectified, kD-trees which regard the epipolar constraint are created for the images observed from the right camera view, i.e. queries for such a tree only return elements which lie on the same image row as the query pixel. This way, the number of leaves per node for the epipolar trees are lowered efficiently, which speeds up the initialization process without loss of accuracy. For further acceleration, this initialization is used on the coarsest resolution only, and the gaps when evolving to the next higher scale are filled up by the propagation.

**Multi-Scale Propagation.** The initial matches are spread by propagation and steadily refined by random search. This is done over multiple scales which helps to distribute rare correct initial matches over the whole image. For each scale, several iterations of propagation are performed in one out of the four image quadrants so that each direction is used equally. During propagation, a scene flow vector is replaced if the propagated vector has a smaller matching cost. If this is not the case, the propagation along this path continues with the existing scene flow vector. After each iteration a random search is done. That means that for all pixels a uniformly distributed random offset in the interval $]-1, 1[$ in pixel units of the current scale is added to each of the four scene flow components to check whether the matching cost decreases. Both propagation and random search help to obtain a smoothly varying vector field and to find correct matches even if the initialization is slightly flawed. For the different scale spaces, the scaling is simulated by smoothing the images and taking only every $n$-th pixel for a sub-sampling factor of $n = 2^k$ so that the patches consist of the same number of pixels for all scales. This way, (up-)sampling errors are prevented because all operations are performed on exact pixel locations on the full image resolution. Smoothing is done by area-based down-sampling

followed by up-sampling using Lanczos interpolation. Note that this matching method has already been used in [BTS15; BTS19], but while it was for optical flow in this work, here it is applied to twice as many dimensions in the search space.

**Consistency Check.** The matching procedure yields a dense map of scene flow correspondences across all images. However, many of the correspondences are wrong because of occlusions, out-of-bounds motion or simply because of mismatching due to challenging image conditions. To remove these outliers, a two-step consistency check is performed. Firstly, an inverse scene flow field is computed for which $I^r_{t+1}$ is the reference image. Temporal order as well as points of view are swapped. Everything else remains as explained above. During the consistency check, optical flow and both disparity maps for each pixel are compared to the corresponding values of the inverse scene flow field. If either difference exceeds a consistency threshold $\tau_c$ in image space, the scene flow vector is removed. Secondly, small regions of the remaining pixels are formed as in [BTS15], where a pixel is added to a region if it has approximately the same scene flow vector. Afterwards, it is checked if one of the already removed outliers in the neighborhood could be added according to the same rule. If this is possible and the region is smaller than $s_c$ pixels, the whole region is removed. This way, the filtered final scene flow correspondences of high accuracy and very few outliers are obtained (cf. Table 4.3). Because the joint filtering of the matches removes more disparity values than necessary, gaps in the disparity maps are filled up with additional values. These values are the result of a separate consistency check for the disparity matches only. For the separate check a second disparity map with Semi-Global Matching (SGM) [Hir08] is computed and compared with the same threshold $\tau_c$ as before. The additional disparity values that are retrieved this way are as accurate as the one from the standard consistency check but much denser, which is shown in Figure 4.3 and Table 4.3.

### 4.1.2. Dense Interpolation

**Sparsification.** Before interpolating the filtered scene flow field in order to recover full density, an additional sparsification step is performed. This helps to extend the spatial support of the neighborhoods during the interpolation and speeds up the whole process [BTS15]. For non-overlapping $3 \times 3$ blocks, only the match with the lowest consistency error during filtering is selected. The remaining matches are called seeds with respect to the interpolation.

**Interpolation Boundaries.** A crucial part of the interpolation is the estimation of scene flow boundaries. While [BTS15; RWHS15] approximate motion boundaries for optical flow with a texture-agnostic edge detector [DZ13], the edge detector used here is trained on semantic boundaries. It is argued that this models geometric boundaries as well as motion boundaries much better than image edges and is much more robust to lighting, shadows, and coarse textures.

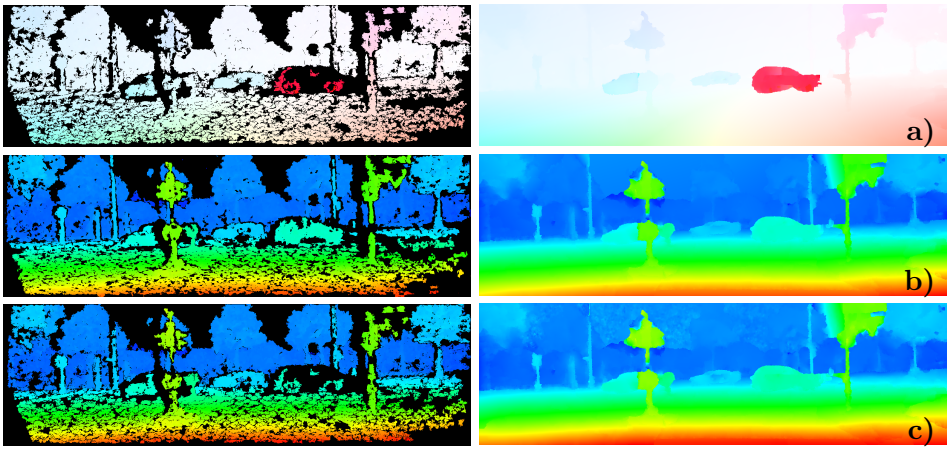



**Figure 4.3.:** *Sparse correspondences (left) and dense interpolation (right). Optical flow* ***(a)*** *and disparities at* $t_0$ ***(b)*** *and* $t_1$ ***(c)***.

To do so, about 400 images of the KITTI data set from [HF16; RRGB+15; XDBZ+16] have been gathered which have been labeled with semantic class information. Within these images, semantic classes that in general neither align with geometric nor motion discontinuities have been merged, e.g. lane markings and road, or pole and panel. The boundaries between the remaining semantic labels are used as binary edge maps to train the edge detector. Similar to the approach in [RWHS15], the framework of Structured Edge Detection (SED) [DZ13] is used to train a random forest with the same parameters as in their paper, except for the number of training patches. For SFF, twice as many positive and negative patches are sampled during the training because the data set is bigger and contains images of higher resolution. The impact of the novel boundary detector is evaluated in Section 4.3.

**Interpolation Models.** For the interpolation of geometry and motion, two different models are tested. Both parts are interpolated separately which leads to a more accurate reconstruction of the scene. This is due to the fact that the separate consistency check for disparity leaves more geometric matches where motion would leave image boundaries. Suppose a local, boundary-aware neighborhood of seeds is given for each unknown scene flow vector $\hat{\boldsymbol{s}}$ at pixel $\hat{\mathbf{p}}$ for geometric and motion seeds respectively, $\mathcal{N}_{geo}$ and $\mathcal{N}_{motion}$. The depth of pixel $\hat{\mathbf{p}}$ is reconstructed by fitting a plane $E(\hat{\mathbf{p}}) : a_1 x + a_2 y + a_3 = d_0$ through all seeds of the neighborhood $\mathcal{N}_{geo}$. This is done by solving a linear system of equations for all neighboring seed points $\mathbf{p_g}$ for which the disparity values are known, using weighted least squares. The weights for each seed are obtained from a Gaussian kernel $g(D) = \exp(-\alpha D)$ on the distance $D(\hat{\mathbf{p}}, \mathbf{p_g})$ between target pixel and seed. The missing disparity value of $\hat{\mathbf{p}}$ is obtained by plugging the coordinates of $\hat{\mathbf{p}}$ into the estimated plane equation. In a similar fashion, but using a neighborhood of motion seeds $\mathcal{N}_{motion}$, the missing 3D motion is obtained by fitting an affine 3D transformation $\mathbf{P_1} = \mathbf{A}\mathbf{P_0} + \mathbf{t}$ using weighted

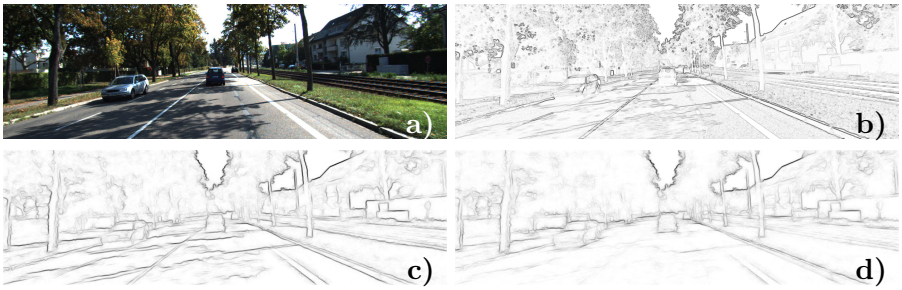



**Figure 4.4.:** *Whereas SED [DZ13] **(c)** detects all image boundaries similar to the image gradient **(b)**, the improved boundary detector **(d)** suppresses lane markings and shadows.*

least squares on all motion seeds $\mathbf{p_m}$. Where $\mathbf{P_t} = (X_t, Y_t, Z_t)^T$ are the 3D world coordinates of motion seed $\mathbf{p_m}$ at time $t_0$ and $t_1$, and $[\mathbf{A}|\mathbf{t}] \in \mathbb{R}^{3\times4}$ is the affine 3D transformation of twelve unknowns. The weights are computed by the same Gaussian kernel as for geometric interpolation, but using the distances $D(\hat{\mathbf{p}}, \mathbf{p_m})$ between the target pixel and the motion seeds. To summarize, for the full reconstruction of scene flow $\hat{\boldsymbol{s}} = (u, v, d_0, d_1)^T$ at pixel $\hat{\mathbf{p}}$, $d_0$ is computed by using the plane model $E(\hat{\mathbf{p}})$, re-projecting the point into 3D world space, transforming it according to its associated affine transformation $[\mathbf{A}|\mathbf{t}]$, and projecting it back to image space to obtain $u$, $v$ and $d_1$.

**Edge-Aware Neighborhood.** To find the local neighborhoods, the idea of Revaud et al. [RWHS15] is applied using both their approximations. That is firstly, the $n$ closest seeds to a pixel $\hat{\mathbf{p}}$ are the $n-1$ closest seeds to the closest seed of $\hat{\mathbf{p}}$, thus all pixels with the same closest seed share the same local neighborhood. And secondly, the distance between $\hat{\mathbf{p}}$ and its closest seed is a constant offset for all neighboring seeds, which can be neglected. It is therefore sufficient to find a labeling that assigns each pixel to its closest seed and to find the local neighborhood for each seed. The graph-based method of [RWHS15] is used for this, where the distances between seeds are the geodesic distances that are directly based on the edge maps from the boundary detector (cf. Figure 4.4).

Since this first interpolation mechanism for scene flow is greatly inspired by the algorithm in EPICFlow [RWHS15], the adapted higher dimensional version is called *EPIC3D*.

### 4.1.3. Variational Optimization

To further refine the 3D motion after interpolation, variational energy minimization is used to optimize the objective

$$E(u, v, d') = E_{data}^{flow} + E_{data}^{cross} + \varphi \cdot E_{smooth}. \tag{4.5}$$

Motion is represented in image space by optical flow and the change in disparity $d' = d_{t+1} - d_t$. The energy consists of three parts. Two data terms, one temporal

correspondence and one cross correspondence, and an adaptively weighted smoothness term for regularization. The data terms use the gradient constancy assumption. Experiments have shown, that an additional term for the color constancy assumption can be neglected.

$$E^{*}_{data}(I, I', \mathbf{p}, \mathbf{w}) = \int_{\Omega} \beta(\mathbf{p}, \mathbf{w}) \cdot \Psi\left(\gamma \cdot \left|\nabla I'(\mathbf{p}+\mathbf{w}) - \nabla I(\mathbf{p})\right|^2\right) \mathrm{d}\mathbf{p} \tag{4.6}$$

$$E^{flow}_{data} = E^{*}_{data}\left(I^{l}_{0}, I^{l}_{1}, \mathbf{p}, (u, v)^T\right) \tag{4.7}$$

$$E^{cross}_{data} = E^{*}_{data}\left(I^{l}_{0}, I^{r}_{1}, \mathbf{p}, (u - d_0 - d', v)^T\right) \tag{4.8}$$

The data terms do not contribute to the energy if the function

$$\beta(\mathbf{p}, \mathbf{w}) = \begin{cases} 1, & \text{if } (\mathbf{p}+\mathbf{w})^T \in \Omega \\ 0, & \text{otherwise} \end{cases} \tag{4.9}$$

indicates that the scene flow is leaving the image domain. The smoothness term

$$E_{smooth} = \int_{\Omega} \Psi\left(|\nabla u|^2 + |\nabla v|^2 + \lambda \cdot \left|\nabla d'\right|^2\right) \mathrm{d}\mathbf{p} \tag{4.10}$$

penalizes changes in the motion field and is weighted by

$$\varphi(\mathbf{p}) = e^{-\kappa B(\mathbf{p})} \tag{4.11}$$

where $B(\mathbf{p})$ is the edge value of the boundary detector at pixel $\mathbf{p}$. All parts use the Charbonnier penalty $\Psi(x) = \sqrt{x^2 + \varepsilon^2}$ to achieve robustness. Since the smoothness term rather enforces constancy than smoothness if $\beta$ for both data terms is zero, the scene flow is not optimized at pixels where the interpolated scene flow field leaves $\Omega$. This energy formulation is inspired by [BBPW04; HD07; WRVB+08]. Linear approximations of the Euler-Lagrange equations for the objective are used within the framework of Brox et al. [BBPW04] without the coarse-to-fine steps to find a solution by Successive Over-Relaxation (SOR).

### 4.1.4. Ego-Motion Model

In Section 4.3 it is shown that the approach as described so far achieves results comparable to state-of-the-art. For the special challenges of the KITTI data set, an additional, optional assumption is made to further improve the performance of SFF. Following [TSS17], it is argued that most parts of a scene are static and thus the 3D motion for these areas is fully determined by the ego-motion of the observer. Given the ego-motion and a motion segmentation into static and dynamic areas, the inverse ego-motion can be applied to all static points in the scene to obtain the apparent scene flow. Using the matching and interpolation scheme above, both required components can be estimated easily with almost no additional effort.

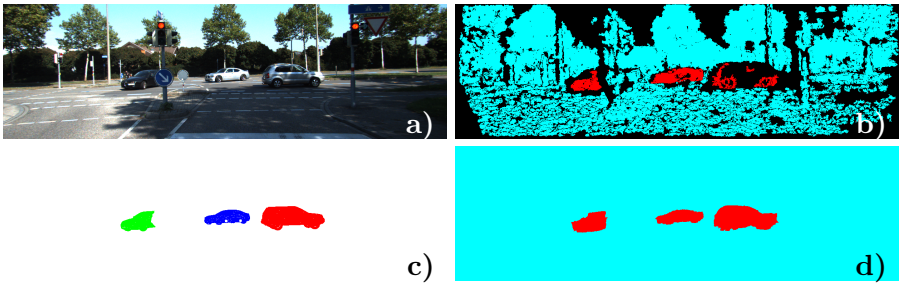



**Figure 4.5.:** *Example of the motion segmentation. Sparse motion indicators as obtained during ego-motion computation* ***(b)****, dense segmentation by interpolation* ***(d)*** *and moving ground truth objects as provided by KITTI [MG15]* ***(c)****.*

**Ego-Motion Estimation.** The filtered scene flow field before interpolation provides very accurate matches across all images. 3D-2D correspondences between the reference frame and the temporally subsequent frame are computed by triangulation using the stereo matches. For this, the depth of these correspondences is limited to 35 meters because the resolution of disparity for farther distances is less. This leads to a Perspective-n-Point (PnP) problem, which is solved iteratively using Levenberg-Marquardt and RANSAC to find the relative pose between the left cameras at time $t_0$ and $t_1$ by minimizing the re-projection error of all correspondences. For RANSAC, a correspondence is considered as outlier if the re-projection error is above 1 pixel. After a first estimation, the set of inliers is recomputed with a relaxed threshold of 3 pixels and the pose $P = [R|t] \in \mathbb{R}^{3\times4}$ is re-estimated. The two-stage process helps to avoid local optima and to find a trade-off between diverse and robust correspondences.

**Motion Segmentation.** An initial sparse motion segmentation can directly be obtained as a side product of the ego-motion estimation. Outliers in the correspondences are considered in motion, while points in conformity with the estimated ego-motion are marked as static, i.e. $m(\mathbf{p}) \in \{0, 1\}$. Within the boundary-aware interpolation, a dense segmentation is computed (cf. Figure 4.5). Pixels labeled as moving are spread up to the boundaries of the object within they are located. Because the segmentation is only a binary labeling, no complex interpolation model is needed.

$$\hat{m}(\mathbf{p}) = \frac{\sum_{\mathbf{p}' \in \mathcal{N}_{motion}} D(\mathbf{p}, \mathbf{p}') \cdot m(\mathbf{p}')}{\sum_{\mathbf{p}' \in \mathcal{N}_{motion}} D(\mathbf{p}, \mathbf{p}')} \tag{4.12}$$

An unknown pixel is assigned with the weighted mean of its local neighborhood. The weights are again based on the geodesic distances $D$ between matches. This interpolation method is similar to the Nadaraya-Watson estimator in [RWHS15]. The interpolated motion field is then thresholded to obtain a dense, binary motion segmentation. The quality of this segmentation is evaluated in

**Table 4.1.:** *A comparison between the dual- and multi-frame pipeline. The individual steps of both pipelines are contrasted. The differences are visualized in Figure 4.6.*

| | **SFF** dual-frame | **SFF++** multi-frame |
|---|---|---|
| Initialization | kD-Trees | Previous result |
| Matching | Two time steps | Three time steps and visibility reasoning |
| Consistency Check | Fully inverse | Left-right |
| Interpolation | Edge-preserving | Edge-preserving and robust |
| Refinement | Variational and ego-motion | Not needed |

Section 4.3. Finally, the inverse estimated ego-motion is applied to all points that are labeled as static to perform ego-motion refinement. Because pixels that move out of view are mostly static, this approach is very effective.

## 4.2. Multi-frame Setup and Robust Interpolation

The experiments in Section 4.3 show that SFF, presented in the previous section, yields competitive scene flow results, especially when the ego-motion model is applied. However, there are two main problems with the presented pipeline: 1.) Pixel-wise matching without regularization is error-prone under some circumstances (saturation, lighting variations, homogeneous or repetitive textures, etc.) and even impossible in occluded and other invisible image regions (e.g. out-of-bounds motions, cf. Figure 4.8a). 2.) The accuracy of the interpolation suffers from increasing gap sizes in the filtered scene flow field and from remaining outliers after the consistency check. The ego-motion model can compensate inaccurate interpolation to some extend, but not sufficiently. The robust extension to multiple frame pairs in this section, addresses both major issues.

The concept of choice to handle non-matchable regions is sparse-to-dense interpolation. However, even though the consistency check removes outliers reliably, the gaps can not be refilled correctly by the interpolation in some scenarios. Therefore, the problem is tackled before it occurs by using image information from multiple frames to avoid mismatches and resolve ambiguity in non-matchable regions. This results in more accurate and better distributed matches of higher density as shown in Section 4.3.1. Further, the concepts for interpolation from RICFlow [HSL16] are transferred to the scene flow domain to improve robustness during interpolation.

When using more than two stereo image pairs, a constant motion is assumed. That means, the observed motion at both time steps (from $t-1$ to $t$ and from

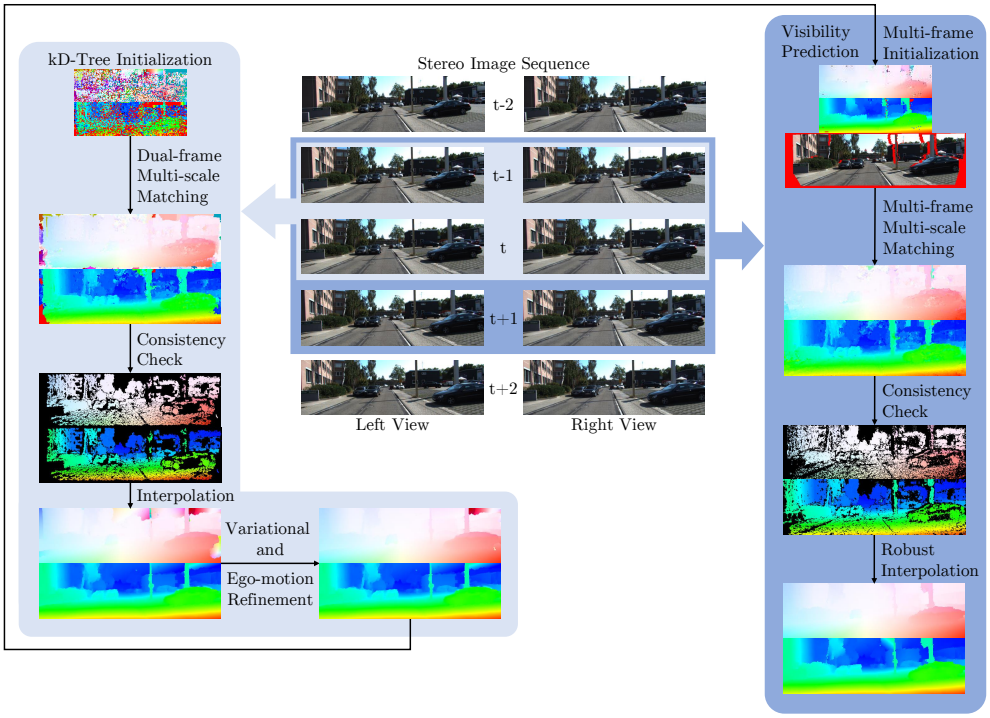



**Figure 4.6.:** *Overview of the dual- and multi-frame approach. A sequence of stereo image pairs is the input to both methods. SFF (left) uses two frame pairs, initializes on a sub-scale and applies multi-scale propagation and random search for matching. The matched result is filtered in a consistency check, interpolated and refined by variational optimization and the optional ego-motion model. The final result is used to initialize the multi-frame process SFF++ (right) which uses three frame pairs. Matching is done with explicit visibility reasoning over all images. The consistency check is adjusted to the multi-frame setup, and the interpolation uses new concepts for increased robustness. Note that there is no refinement necessary in the improved robust multi-frame pipeline. Each step is illustrated by the corresponding optical flow and one disparity map at the reference time of the respective scene flow field. Table 4.1 summarizes the differences between the dual-frame and multi-frame approach.*

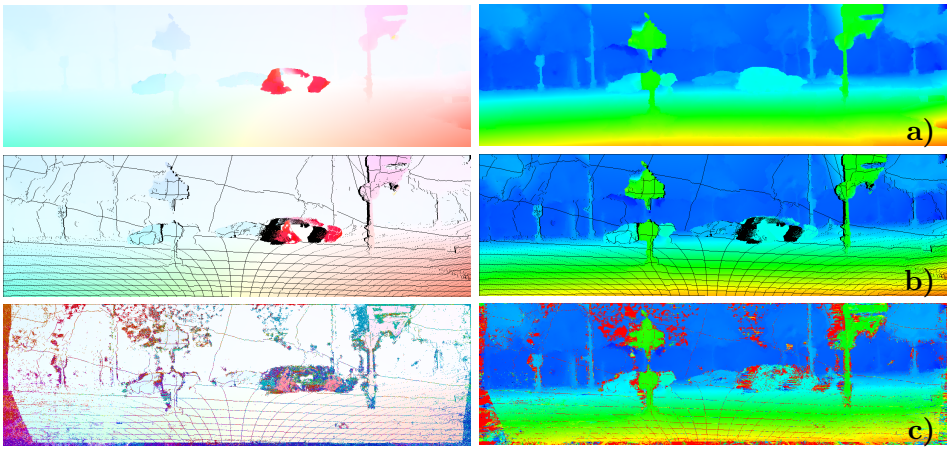



**Figure 4.7.:** *Previous scene flow result* ***(a)****, temporally warped scene flow* ***(b)****, and the multi-frame initialization* ***(c)*** *(on full resolution for visualization purposes).*

$t$ to $t+1$) is assumed to be the same. The error made by this assumption converges towards zero for continuous motions with increasing frame rates. The multi-frame approach is additionally designed to process video streams in an online manner. That means that the first two frame pairs of a sequence are processed with the dual-frame approach from Section 4.1. All subsequent frames are then added in an incremental way and processed within a sliding temporal window of three frame pairs. The additional information of the extra images are exploited during matching in three ways. Firstly, the previous results are used during initialization. Secondly, the previous scene flow is used to predict the visibility of the scene. Lastly, the constant motion assumption together with the visibility prediction are used to match scene flow across all six relevant images.

The overall structure of the multi-frame approach remains the same as in the dual-frame method but with two more images. Using a set of six input images $\mathbf{I} = \{I_0^l, I_0^r, I_1^l, I_1^r, I_{-1}^l, I_{-1}^r\}$, accurate matches are found, possible outliers are removed, and the filtered scene flow is interpolated back to a dense scene flow field. The overview of the multi-frame pipeline is shown in Figure 4.6. A comparison between SFF and SFF++ is given in Table 4.1.

### 4.2.1. Multi-frame Matching

**Initialization.** Improved initialization is the first extension of the multi-frame approach. The previous scene flow result and the assumption of constant motion are used to propagate each 3D point according to its 3D motion to get an initial prediction of the scene flow at the current time step [RGSY+19]. This temporally propagated scene flow prediction is used as an additional choice during the initialization process as described in Section 4.1.1. The process is visualized in Figure 4.7. By not relying on the previous result alone, error propagation is successfully avoided (cf. Figure 4.7).

**Matching Cost.** As before, the matching process is based on the visual similarity of corresponding pixels (cf. Equation 4.1). The correspondences between the reference view and the additional images are added to the scene flow matching cost from Equation 4.2 to obtain

$$\begin{aligned} C_{multi}(\mathbf{p}, \boldsymbol{s}) = & C\left(I_0^l, \mathbf{p}, I_1^l, \mathbf{p} + (u, v)^T\right) \\ & + C\left(I_0^l, \mathbf{p}, I_0^r, \mathbf{p} + (-d_0, 0)^T\right) \\ & + C\left(I_0^l, \mathbf{p}, I_1^r, \mathbf{p} + (u - d_1, v)^T\right) \\ & + C\left(I_0^l, \mathbf{p}, I_{-1}^l, \mathbf{p} + (u_{-1}, v_{-1})^T\right) \\ & + C\left(I_0^l, \mathbf{p}, I_{-1}^r, \mathbf{p} + (u_{-1} - d_{-1}, v_{-1})^T\right). \end{aligned} \tag{4.13}$$

All pixels of the image domain $\Omega$ are matched to all five other viewpoints to derive corresponding pixel locations in the scene flow representation in 2D (cf. Equation 4.13). For matching with the previous frame pair, inverse scene flow (in image space) $u_{-1}$, $v_{-1}$, $d_{-1}$ is computed by projecting the flow to 3D, inverting it, and projecting it back to 2D, according to the constant motion assumption. Note that $u_{-1} = -u$ if and only if $d_0 = d_1$. The 2D optical flow is not just inverted directly. The (pixel-wise) constant motion assumption allows to match across multiple time steps without increasing the search space or complexity of the scene flow domain.

**Visibility Prediction.** One additional major extension is the explicit visibility reasoning. Since the energy is based on visual data only, it is impossible to match regions, that are not visible in one of the images. Depending on the magnitude of camera movement, the baseline, and other circumstances, these non-matchable areas can become considerably large. Figure 4.8a shows an invisibility mask for pixels whose imaged 3D point is not visible in at least one of the views of $I_0^r$, $I_1^l$, or $I_1^r$. This gives an impression of the limitations of dual-frame matching methods. However, Figure 4.8a also visualizes the remaining invisibility when considering one additional time step. More than two frame pairs can compensate for missing visual evidence when a pixel correspondence that is invisible in one image can be observed in another. Assuming to know which pixels are occluded or out-of-bounds of the image domain in each view, the matching cost for a single image correspondence (Equation 4.1) in Equation 4.13 can be replaced by

$$C_{\text{vis}}\left(I, \mathbf{p}, I', \mathbf{p}', occ', oob'\right) = \begin{cases} \theta_{occ}, & \text{if } occ'(\mathbf{p}) \\ \theta_{oob}, & \text{if } oob'(\mathbf{p}) \wedge \mathbf{p}' \notin \Omega \\ \theta_{penalty}, & \text{if } oob'(\mathbf{p}) \neq \mathbf{p}' \in \Omega \\ C\left(I, \mathbf{p}, I', \mathbf{p}'\right), & \text{else,} \end{cases} \tag{4.14}$$

with $occ'$ and $oob'$ being binary occlusion and out-of-bounds masks that indicate for each pixel $\mathbf{p}$ whether the corresponding point in view $I'$ is visible. For the full multi-frame matching cost in Equation 4.13, two invisibility masks are

**(a)** *Invisibility Example. Reference image (left), invisibility mask in the dual-frame scenario where no visibility handling is applied (middle), and remaining non-matchable areas due to occlusions in the multi-frame scenario where visibility handling is applied (right).*


**(b)** *Visibility Prediction*

**Figure 4.8.:** *In the dual-frame case, occlusions can obscure large parts of the image which remain mostly visible when using multiple frames (a). The explicit visibility handling of the multi-frame matching strategy predicts occluded and out-of-bounds regions (red) for all five images that are matched to the reference frame* $I_0^l$ *(b).*

needed for each of the five corresponding images. The adjusted cost term uses a constant invisibility cost $\theta_{occ}$ in case of occlusions to avoid trivial solutions where all points would be occluded. Same holds for out-of-bounds motions, with the additional option to assign a very large penalty cost $\theta_{penalty}$ if motion under test is inconsistent with the out-of-bounds mask $oob'$. Otherwise, the normal matching cost of Equation 4.1 is used. In practice, $\theta_{occ} = \theta_{oob} = 10\,000$ is chosen based on empirical studies of the matching cost of ground truth scene flow vectors and $\theta_{penalty} = 10^6$ forces out-of-bounds motion to be consistent with the prediction.

Because the algorithm is designed to process sequences sequentially, the estimated scene flow of the previous time step can be used to predict the visibility for all six images. To this end, temporally propagated scene flow prediction is used to check which parts of the scene leave the image domain $\Omega$ for a specific view. These areas are marked as invisible in the associated out-of-bounds mask. It also allows to reason about occlusions by z-buffering, since full 3D information including depth is available. If multiple motions have the same target pixel in the target view, all but the closest are occluded. Examples of the visibility prediction are given in Figure 4.8b. For each of the five relevant frames, pixels of the reference frame that can not be observed from the respective view are masked. All pixels of the reference frame are visible by definition.

The impact of the multi-frame strategy combined with the explicit visibility reasoning is shown in Section 4.3.1 by comparing to the basic dual-frame SFF from Section 4.1 [SWKB+18].

**Consistency Check.** The next step of the pipeline is the consistency check to remove possible outliers. As before, a consistency scene flow field is computed and compared to the corresponding scene flow vectors. Due to the changed setup to multiple frames from a stereo sequence, the way the consistency field is computed needs to be changed. The temporal order of the images (cf. Section 4.1.1) is not inverted anymore, only the stereo viewpoint is changed. This leads to a left-right consistency check. Although a left-right check alone is less reliable than the previous consistency check, this has an important advantage. With inverted temporal order, scene flow correspondences according to the optical flow can not be established where the motion leaves the image boundaries. Thus, out-of-bounds regions are always filtered. This was no issue in the dual-frame approach, but using multiple frames, matching in these regions is actually possible. With the left-right consistency check, it is possible to maintain the correct correspondences in out-of-bounds regions. Matching for the consistency scene flow field is done exactly as before, using multiple scales, visibility reasoning, etc. Afterwards, each scene flow estimate of the reference frame is compared to its corresponding vector of the consistency field component-wise. If any error exceeds $\tau_c = 1$ pixel, the whole scene flow vector is removed. SFF did benefit from an additional consistency check for the disparity only to obtain more matches for the interpolation of geometry. This idea is reused to filter the dense multi-frame matches in a second independent

consistency check with a disparity map computed with the algorithm from [YMU14]. Afterwards, the additional matches and the results of the first full consistency check are merged. The auxiliary stereo algorithm is changed from SGM [Hir08] to SPS [YMU14] because it is faster, more accurate, and also able to match occluded and other invisible regions. The merged correspondences are very accurate and serve as input for the robust interpolation (cf. Figures 4.6 and 4.12).

### 4.2.2. Robust Interpolation

As in SFF, sparsification is applied because sparse-to-dense interpolation works best if the input matches are not too dense already [BTS15; HSL16; SWKB+18]. The sparsification works as an additional outlier filter and increases the spatial support for the same number of neighbors during interpolation.

The improved interpolation is robust against possible remaining outliers and edge-preserving to create sharp motion boundaries at object edges. As in the less robust approach, interpolation is separated for the 3D geometry and the 3D motion. Because of this separation, different sets of input seeds can be used for each. Other than SFF in Section 4.1 [SWKB+18] where the interpolation algorithm of [RWHS15] was transferred to the higher dimensional scene flow problem, the extension adopts the robust concepts of RICFlow [HLS17] for the scene flow problem. It is therefore coined as *RIC3D*.

In short, the reference frame is segmented into superpixels of size 25. Each superpixel is associated with a local neighborhood of the 200 closest input matches as shown in Figure 4.9. The distance is computed as a geodesic distance based on an edge map. Additionally, each superpixel is initialized with two scene flow models, one for the 3D geometry and one for the 3D motion. Afterwards, the models for each superpixel are adjusted by propagation and random search. Based on the final models, dense scene flow can be computed for each pixel.

**Interpolation Models.** The geometric model is a slanted plane for each superpixel. It is initialized with a constant depth, i.e. parallel to the image plane. The motion model is a rigid transformation of 3D rotation and 3D translation for each superpixel. This rigid model is less universal, but more robust compared to the affine model in Section 4.1.2. Rigid motions are not presumed, though. The approach is not limited to rigid motions. Due to the small size of the superpixel segmentation, non-rigid motions can be approximated closely. Same holds for the planarity. Arbitrarily curved surfaces are approximated by very small plane segments. The superpixel motion is initialized with translation only. For initialization, three different strategies have been tried: 1.) The basic approach of [HLS17] in which the initial values are obtained from the closest input seed. 2.) One using the local weighted medoid. 3.) One estimating the weighted geometric median. Though the multivariate estimators are much more accurate than the nearest neighbor method, the robust propagation with random search can compensate for less accurate initialization. Independent of

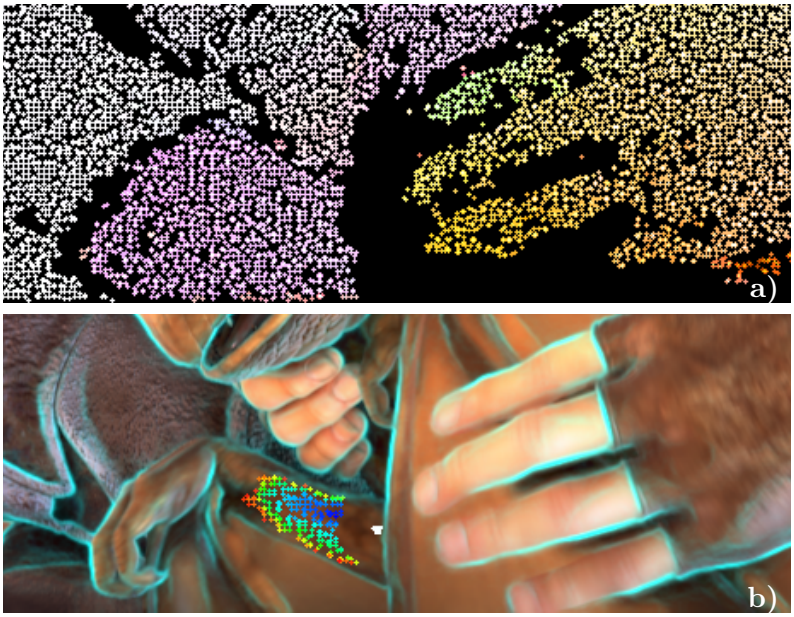



**Figure 4.9.:** *Sparsified input matches for interpolation* ***(a)*** *and the edge-aware neighborhood of seeds for the superpixel in white* ***(b)****. Note that this is the actual size of segments to which the interpolation models are applied. The supporting seeds are colored according to their distance to the superpixel.*

the superpixel initialization, the final interpolation result is almost the same in all cases. In practice, the robust geometric median is used, since it provides a reasonable tradeoff between accuracy and run time.

**Robust Model Estimation.** To optimize the models for interpolation (plane + rigid transformation), a robust, stochastic approach in a RANSAC-like fashion is applied. For each superpixel, random seeds from a local edge-aware neighborhood are sampled and used to predict the models. Apart from random sampling, propagation tests the estimated models at neighboring superpixels. To determine the fitness of a model, a truncated error for all supporting input seeds $\mathbf{p}$ of the local neighborhood $\mathcal{N}$ are summed

$$C(M) = \sum_{\mathbf{p} \in \mathcal{N}} \min\left(\tau, \exp\left(-\frac{1}{\alpha} \cdot D_{sp}(\mathbf{p})\right) \cdot \epsilon(\mathbf{p})\right). \tag{4.15}$$

In this formula, $M$ is the respective model under test, $\tau$ is the truncation error threshold of 4 pixels, and $\epsilon(\mathbf{p})$ is the re-projection error when applying model $M$ to input seed $\mathbf{p}$ which is weighted by a geodesic distance $D_{sp}$ between the seed $\mathbf{p}$ and the position of the respective superpixel with weight coefficient $\alpha = 0.6$. By minimizing the cost of all models for all superpixels using propagation and random model generation, robust interpolation models for geometry and motion are obtained. As before with the multi-frame matching, the novel robust interpolation is compared directly to the previous interpolation method for scene flow matches of Section 4.1.2 in Section 4.3.1.

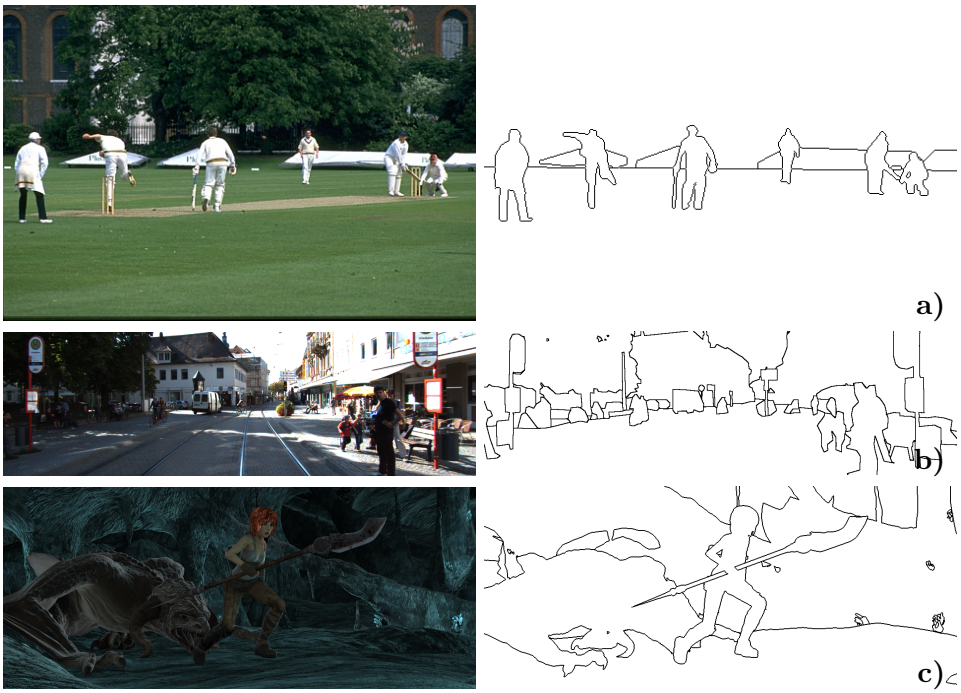
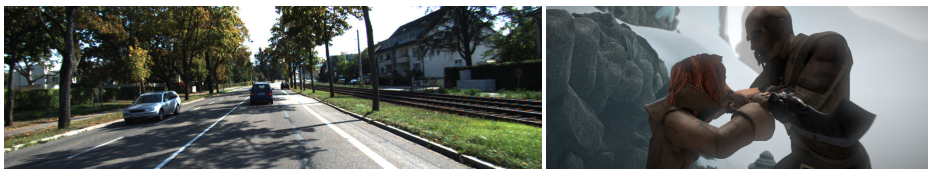


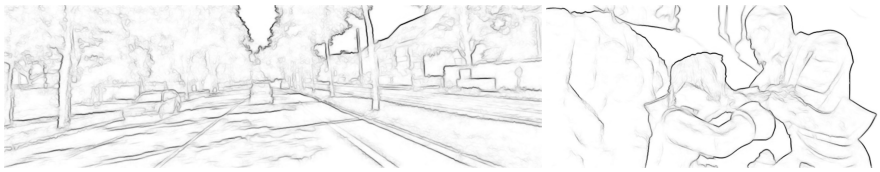

**Figure 4.10.:** *Examples of boundaries from BSDS [MFTM01] (a), KITTI [GLU12] (b), and Sintel [BWSB12] (c) used to train the universal edge detector.*

**Universal Boundary Detector.** It is shown in Section 4.3.1 that the boundary detector from Section 4.1.2, trained on semantic boundaries of KITTI, performs much better than the original detector [DZ13] on that particular domain. However, improved robustness requires reliability across multiple domains. Thus, a third variant with the goal to perform equally well on several different data sets is trained. Towards this end, a joint set of training images from the 200 images of the original BSDS data set [MFTM01], semantic boundaries for 424 images of KITTI and for 100 images of Sintel are used. The semantics for KITTI are the same as before. For Sintel, the sequences *cave_2* and *sleeping_1* of the *clean* rendering pass are used, which are excluded during evaluation. Semantics are created by merging the provided mesh and material segmentation, e.g. all segments belonging to the dragon are labeled with the same ID. Examples of ground truth boundaries for each of the data sets are given in Figure 4.10. By combining these three sets, a total of 724 images for training are obtained. The combined edge model is much more versatile and applicable to different data sets. Examples of the detected edges are visualized in Figures 4.9, 4.11 and 4.15. Textures and shadows are suppressed, while at the same time, object boundaries are detected accurately.

For optional refinement of the dense 3D motion field, variational optimization of Section 4.1.3 and the ego-motion model of Section 4.1.4 are considered. However, the improvements in matching and interpolation make both refinement steps obsolete as as it is shown in Section 4.3.1.



**(a)** *Input images from different data sets.*



**(b)** *Edge result from SED trained on BSDS.*

**(c)** *Edges obtained using the KITTI model.*

**(d)** *Results from the unified boundary detector (mixed).*

**Figure 4.11.:** *Whereas SED [DZ13] (b) detects all image gradients, the KITTI boundary detector (c) suppresses lane markings and shadows. The unified detector of Section 4.2.2 (d), achieves a good tradeoff between the advantages of (c) and generalization abilities across different domains.*

## 4.3. Evaluation and Results

The dual-frame approach (Section 4.1) and the multi-frame method (Section 4.2) are both evaluated and compared to previous work in this section. The detailed analysis consists of various experiments. First, individual components are evaluated in an extensive ablation study. Second, the approaches are compared to (former) state-of-the-art on two diverse data sets. Finally, limitations of the sparse-to-dense methods are disclosed which further motivate the next chapters of this thesis.

**Evaluation Setup.** For all experiments, the explicit values of the previous sections and the following parameters are used, even across different data sets. For $k = 3$ subscales and full resolution, 12 iterations of propagation and random search are traversed. The consistency threshold $\tau_c$ is set to 1 and a minimal region size of $s_c = 150$ for the region filter is used. During interpolation, the geometry and motion neighborhoods consist of 160 and 80 seeds respectively and the Gaussian kernel uses $\alpha = 2.2$ to weight the geodesic distances. For the variational energy minimization, the parameters are set to $\kappa = 5$, $\gamma = 0.77$, $\lambda = 10$ and $\varepsilon = 0.001$. The optimization framework runs through two outer and one inner iteration with 30 iterations for the SOR solver using a relaxation factor of $\omega = 1.9$. The binary motion segmentation is obtained by thresholding the interpolated motion field with $\tau_m = 0.4$ when applying the ego-motion model. All these parameters are found by performing a grid search.

### 4.3.1. Ablation Studies

The component-wise evaluation includes experiments on the following components:

- The individual steps of the sparse-to-dense pipeline, i.e. matching, filtering, and interpolation.
- The two kinds of proposed boundary detectors, i.e. trained on the semantics of KITTI and a unified model trained on a mixture of data sets.
- The (optional) post-processing steps, i.e. variational optimization, the ego-motion model, and the quality of the motion segmentation.

**Sparse-to-Dense Pipeline.** The modularity allows to replace or leave out various parts of the pipeline. This way, it is easy to evaluate the effect of each component separately. This is done in Table 4.2 by evaluating the results of all 200 training images of the KITTI data set [MG15] for several variants of the method. In particular, the number of frames for the matching are varied between *dual* and *multi*. The interpolation mechanism is also altered from the original one presented in Section 4.1.2 (*epic3d*) to the robust version presented within Section 4.2.2 (*ric3d*). Further, the impact of the training data for the boundary detector is evaluated. The original version is trained on BSDS [MFTM01] (*bsds*). In Section 4.1.2, a specialized version optimized for the KITTI data set is presented (*kitti*). Also, results for the universal boundary

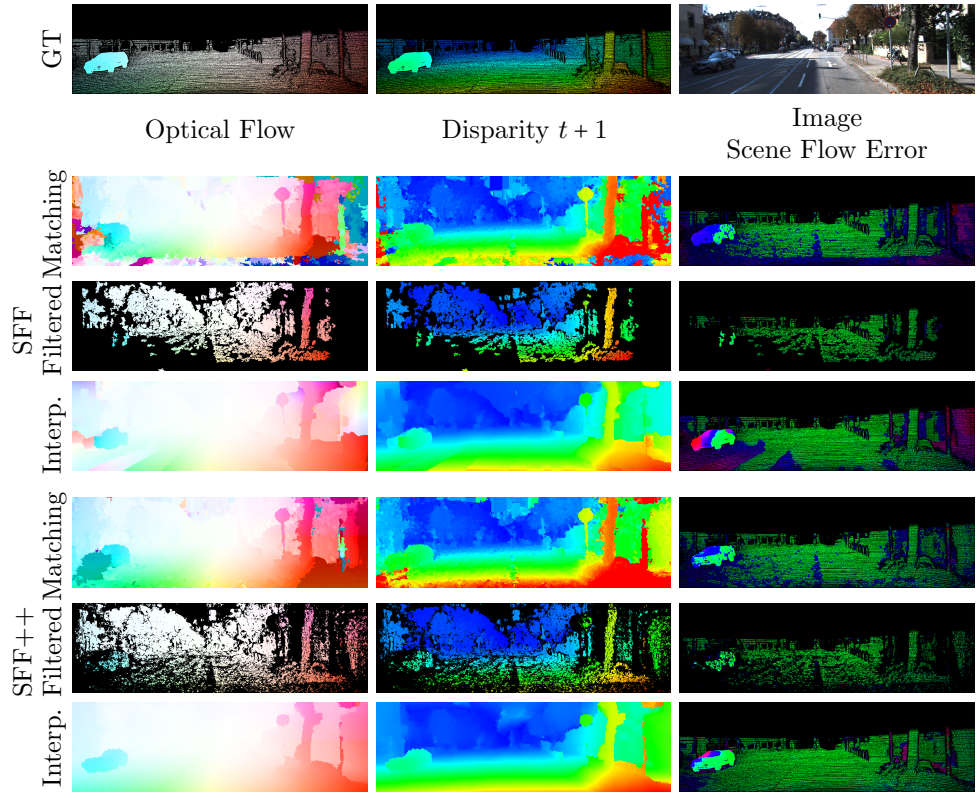



**Figure 4.12.:** *Visual Comparison of SFF and the improved SFF++. Optical flow, disparity at $t+1$ and scene flow error maps are shown for matching, filtering, and dense interpolation.*

detector (cf. Section 4.2.2) trained on a mix of data from BSDS, KITTI, and Sintel [BWSB12] are given (*mixed*). Finally, the variational refinement (*var*) and the ego-motion optimization (*ego*) are successively added for different combinations of the previous components.

The dedicated KITTI boundary detector outperforms the original version that was trained on BSDS only. The universal boundary detector that was trained on mixed data performs almost equally well on KITTI and much better on other data sets like Sintel (cf. Section 4.3.3. Further, the novel, more robust interpolation improves the accuracy of the scene flow result greatly, making the ego-motion model and even the variational refinement in almost all cases obsolete. Multi-frame matching improves the results even more. Next, the impact of the multi-frame setup is analyzed in more detail.

For a visual comparison of the robust multi-frame approach SFF++ [SWUK+20] (Section 4.2) to SFF [SWKB+18] (Section 4.1), optical flow and disparity at the next time step are visualized in Figure 4.12 along with a scene flow error map for full matching, filtered matches, and dense interpolation. The error maps give correct estimates in green and outliers in blue to red according to the KITTI metric. The multi-frame approach with explicit visibility reasoning is able to match occluded areas (e.g. next to the tree and

**Table 4.2.:** *Results of the scene flow estimation pipeline for different combinations of matching and interpolation settings, before and after variational optimization and ego-motion refinement. The focus here is on the comparison between dual- and multi-frame matching and on the improvement by robust interpolation.*

| | Match | Interp. | Edges | Var | Ego | **SF-bg** | **SF-fg** | **SF-all** |
|---|---|---|---|---|---|---|---|---|
| | dual | epic3d | bsds | ✗ | ✗ | 26.93 | 32.16 | 27.73 |
| | dual | epic3d | bsds | ✓ | ✗ | 26.14 | 30.71 | 26.84 |
| | dual | epic3d | bsds | ✓ | ✓ | 13.39 | 30.92 | 16.07 |
| | dual | epic3d | kitti | ✗ | ✗ | 25.60 | 28.99 | 26.12 |
| | dual | epic3d | kitti | ✓ | ✗ | 24.78 | 27.37 | 25.18 |
| (SFF) | dual | epic3d | kitti | ✓ | ✓ | 12.04 | 28.31 | 14.53 |
| | dual | ric3d | kitti | ✗ | ✗ | 12.10 | 22.86 | 13.74 |
| | dual | ric3d | kitti | ✓ | ✗ | 12.11 | 22.96 | 13.77 |
| | dual | ric3d | kitti | ✓ | ✓ | 11.57 | 23.46 | 13.40 |
| | multi | ric3d | kitti | ✗ | ✗ | 10.75 | 19.37 | 12.07 |
| | multi | ric3d | kitti | ✓ | ✗ | 11.26 | 19.60 | 12.54 |
| | multi | ric3d | kitti | ✓ | ✓ | 11.27 | 20.11 | 12.62 |
| (SFF++) | multi | ric3d | mixed | ✗ | ✗ | 10.93 | 19.67 | 12.27 |
| | multi | ric3d | mixed | ✓ | ✗ | 11.42 | 20.12 | 12.75 |
| | multi | ric3d | mixed | ✓ | ✓ | 11.29 | 20.62 | 12.72 |

traffic signs) and out-of-bounds regions (e.g. the front of the car) that are both not visible in some of the relevant frames (cf. Figure 4.8b). In addition, the robust interpolation can handle regions with almost no input seeds reliably (e.g. the lower left part of the image). Table 4.4 compares the same in terms of average end-point error and average outliers on all KITTI training sequences for all ground truth pixels (KITTI *occ* data) and occluded regions only (*occ* without *noc*). The novel method is already very accurate during matching with a much smaller average end-point error compared to SFF. Unlike SFF, the multi-frame approach is able to match almost 40 % of the invisible areas. Though overall filtered results after the consistency check appear to be worse, the multi-frame approach SFF++ is able to retain 9.8 % of the matches in occluded areas where SFF filters almost everything. That makes the spatial distribution of the multi-frame matches preferable over that of the dual-frame approach (cf. Figure 4.12). Because of this and due to the robust interpolation, final results are much better with only about one third of the scene flow outliers in occluded areas and less than half the percentage of overall outliers.

The direct comparison in these experiments shows that the multi-frame matching strategy with explicit visibility reasoning and robust interpolation is superior to the previous variant of SFF [SWKB+18] (Section 4.1). Even with the significant boost of the optional ego-motion model of SFF (Section 4.1.4), the method without this model outperforms the dual-frame version on both evaluated data sets (see Sections 4.3.2 and 4.3.3).

Additionally, the accuracy and densities for the two-stage consistency check

of SFF with respect to the KITTI ground truth are presented in the last two rows of Table 4.3 for the sparse scene flow matches (matches) and the separately filtered sparse stereo correspondences (disparity). This validates the much higher density when filtering disparity separately while maintaining the accuracy.

**Boundary Detection.** The impact of the two proposed boundary detectors are evaluated by running SFF and SFF++ twice using different boundary models. First, SFF is evaluated using the standard edge detection as in [DZ13] (*bsds*) and using the structured random forest trained on semantic edges of the KITTI data set (*kitti*). All variants using improved edge detection outperform their according variant using basic image edges. The major improvements are visualized in Figure 4.4. High image gradients at lane markings or shadows (especially shadows of vehicles) are effectively suppressed when using the semantic boundary detector, while at the same time it accurately detects all kinds of objects. This helps greatly to smoothly recover the street surface during interpolation, to sharpen discontinuities in depth and motion in general, and it allows for accurate boundaries when interpolating the motion segmentation. Second, results for SFF++ are presented using the *kitti* model and the unified boundary detector which is trained on a mixture of data (*mixed*). A minor degradation of the results can be observed when switching from the *kitti* model to the universal detector (*mixed*). However, the *mixed* model performs equally well on diverse data sets (cf. Figure 4.11d). All results are presented in Tables 4.2 and 4.3.

**Post-Processing.** A similar study for dense refinement strategies is evaluated in Tables 4.2 and 4.3. In Table 4.3, SFF is evaluated without the variational optimization (no var), the full dual-frame approach (full) and with the optional ego-motion extension (full+ego). The variational optimization is primarily useful for optical flow and foreground regions. Table 4.2 validates that both types of post-processing are not needed within the multi-frame setup of SFF++ anymore. In fact, the results are slightly worse.

**Motion Segmentation.** Finally, the provided object maps of KITTI are used to test the performance of the motion segmentation (cf. Figure 4.5). To this end, precision and recall for the binary segmentation are computed. Precision is defined as the percentage of estimated pixels that are correctly labeled as moving. The recall is the relative amount of ground truth pixels that are labeled as moving and covered by the estimation. Over all frames, a precision of about 28 % and a recall of about 83 % are achieved. Most of the missed ground truth foreground pixels belong to objects which are far away and moving parallel to the direction of viewing. Therefore, the re-projection error of the 3D-2D correspondences during ego-motion estimation drops easily below the threshold. Two remarks have to be considered regarding the precision. Firstly, KITTI only annotates cars that are mostly visible, i.e. pedestrians, cyclists, other vehicles, or partly occluded cars are not included in the ground truth, but

**Table 4.3.:** *Evaluation after different steps of the post-processing of SFF on KITTI [MG15] training data. The new edge detector outperforms SED [DZ13]. The ego-motion model helps greatly to improve overall results. The bottom two rows show the amount of outliers in the sparse correspondences after the consistency check and before the interpolation. The density is computed with respect to available ground truth pixels in KITTI.*

| Variant | D1-bg | D1-fg | D1-all | D2-bg | D2-fg | D2-all | Fl-bg | Fl-fg | Fl-all | SF-bg | SF-fg | SF-all | Density | Edges |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| full+ego | **5.36** | **10.85** | **6.20** | **7.94** | 18.23 | **9.51** | **10.36** | 22.85 | **12.28** | **12.04** | 28.31 | **14.53** | 100.00 % | *kitti* |
| full | **5.36** | **10.85** | **6.20** | 15.91 | **18.03** | 16.23 | 22.33 | **21.69** | 22.23 | 24.78 | **27.37** | 25.18 | 100.00 % | |
| no var | **5.36** | **10.85** | **6.20** | 15.77 | 18.81 | 16.24 | 23.75 | 23.72 | 23.75 | 25.60 | 28.99 | 26.12 | 100.00 % | |
| full+ego | 5.48 | 11.99 | 6.47 | 9.07 | 19.98 | 10.74 | 11.63 | 25.47 | 13.75 | 13.39 | 30.92 | 16.07 | 100.00 % | *bsds* |
| full | 5.48 | 11.99 | 6.47 | 16.90 | 20.80 | 17.50 | 23.57 | 25.22 | 23.82 | 26.14 | 30.71 | 26.84 | 100.00 % | |
| no var | 5.48 | 11.99 | 6.47 | 16.73 | 21.38 | 17.44 | 25.00 | 27.04 | 25.32 | 26.93 | 32.16 | 27.73 | 100.00 % | |
| matches | 1.91 | **3.74** | 2.18 | 2.48 | 4.08 | 2.71 | 2.10 | 2.78 | 2.20 | 3.87 | 6.24 | 4.21 | 38.82 % | – |
| disparity | **1.42** | 4.06 | **1.82** | – | – | – | – | – | – | – | – | – | **57.81 %** | – |

**Table 4.4.:** *Comparison of intermediate results for SFF and the robust multi-frame extension SFF++ on the KITTI training data.*

| | | *All pixels* | | | | | | | | | *Occluded pixels only* | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **EPE [px]** | | | **Outliers [%]** | | | | **Density [%]** | | **Outliers [%]** | | | | **EPE [px]** | | |
| | | **D1** | **D2** | **Fl** | **D1** | **D2** | **Fl** | **SF** | **SF** | | **D1** | **D2** | **Fl** | **SF** | **D1** | **D2** | **Fl** |
| Matching | SFF | 7.2 | 11.3 | 38.9 | 12.6 | 29.2 | 32.8 | 39.8 | 100.0 | 100.0 | 22.1 | 92.1 | 95.5 | 99.5 | 34.3 | 50.1 | 195.7 |
| | **SFF++** | **2.7** | **4.3** | **9.2** | **11.4** | **20.9** | **23.7** | **31.8** | 100.0 | 100.0 | **14.9** | **39.8** | **55.3** | **62.1** | **5.3** | **7.5** | **25.9** |
| Filtered | SFF | **0.8** | **0.9** | **1.1** | **2.2** | **2.7** | **2.2** | **4.2** | 38.8 | 0.3 | 19.5 | 76.0 | 75.2 | 79.0 | 4.6 | 13.3 | 79.9 |
| | **SFF++** | 0.9 | 1.1 | 1.6 | 2.6 | 4.7 | 5.3 | 8.0 | **41.6** | **9.8** | **2.5** | **12.9** | **23.0** | **26.2** | **1.1** | **2.4** | **9.3** |
| Interpolated | SFF | **1.2** | 3.7 | 16.1 | 6.2 | 16.2 | 23.8 | 26.1 | 100.0 | 100.0 | 9.9 | 43.6 | 62.4 | 64.2 | 2.0 | 11.4 | 86.1 |
| | **SFF++** | **1.2** | **1.9** | **4.7** | **5.2** | **8.8** | **9.7** | **12.7** | 100.0 | 100.0 | **7.1** | **16.0** | **22.7** | **24.8** | **1.7** | **3.3** | **15.3** |

are marked as moving by SFF if they are in motion. Secondly, since areas that are wrongly classified as dynamic are filled with the basic scene flow estimate, which is still of high quality, the overall approach is tuned in favor of a high recall.

### 4.3.2. KITTI Scene Flow Benchmark

The famous KITTI data set [MG15] is used for the first part of the comparison to state-of-the-art in the actual application scenario. This benchmark does not provide reference motions for vulnerable road users like pedestrians or cyclists. The only dynamic objects are rigidly moving vehicles. Results of public submissions to the the KITTI scene flow benchmark [MG15] are presented in Table 4.5 and compared to state-of-the-art. The benchmark evaluates the average amount of outliers for disparity at both time steps (*D1*, *D2*), optical flow (*Fl*), and scene flow (*SF*) as explained in Section 2.3.2. Each category is further divided into regions of background (*bg*, static areas), foreground (*fg*, moving objects), and both (*all*, all available ground truth).

At the time of publication, SFF was ranked 6$^{\text{th}}$ and achieved the 3$^{\text{rd}}$ best result out of all dual-frame methods while at the same time being considerably faster than the top performing methods (cf. Table 4.5). Section 4.3.3 validates that SFF generalizes better to other data sets, where it often outperforms the former best dual-frame method Object Scene Flow (OSF) [MG15]. In Figure 4.13, a visual example of the results are given and compared to two better performing methods from the dual-frame [MG15] and multi-frame [VSR15] categories. It can be seen that the interpolation produces very sharp edges. This in combination with the matching method helps to obtain accurate scene flow, especially for (moving) objects. Methods with comparable overall performance on KITTI [LBAL+16; TSS17], perform worse on moving foreground objects than SFF.

As assumed, the robust multi-frame extension scores better than the dual-frame version. Especially for the important foreground regions, the results are even comparable to the deep learning approach of [RSKS17] which uses semantic segmentation and to that of [MG15] which explicitly estimates a single rigid motion for independent objects. In addition, the sparse-to-dense concept is at least three times faster than the better performing algorithms.

While investigating the discrepancy between the results on the validation data (Table 4.4) and on the test data (Table 4.5), it has been discovered that the multi-frame approach has difficulties when the constant motion assumption is harshly violated (cf. Section 4.4.1). This can happen in KITTI due to the low frame rate of 10 frames per second and because of strong sudden pitch or roll rotations on bumpy streets, e.g. the last example in Figure 4.14. However, in most cases the assumption holds and produces robust and accurate results.

Another visual impression is given in Figure 4.14 by the error maps for different examples of the KITTI test data for PRSM, OSF+TC, FSF+MS, SFF, and the extension SFF++. Even for the partly occluded, poorly illuminated vehicle in the second example, scene flow can be estimated reliably by SFF++. In this example also, scene flow at the occluded areas around the left traffic light

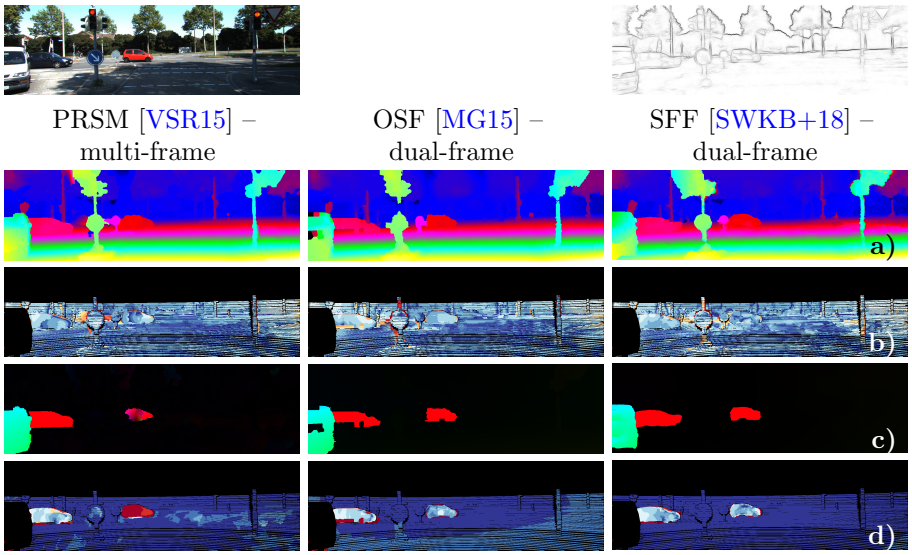
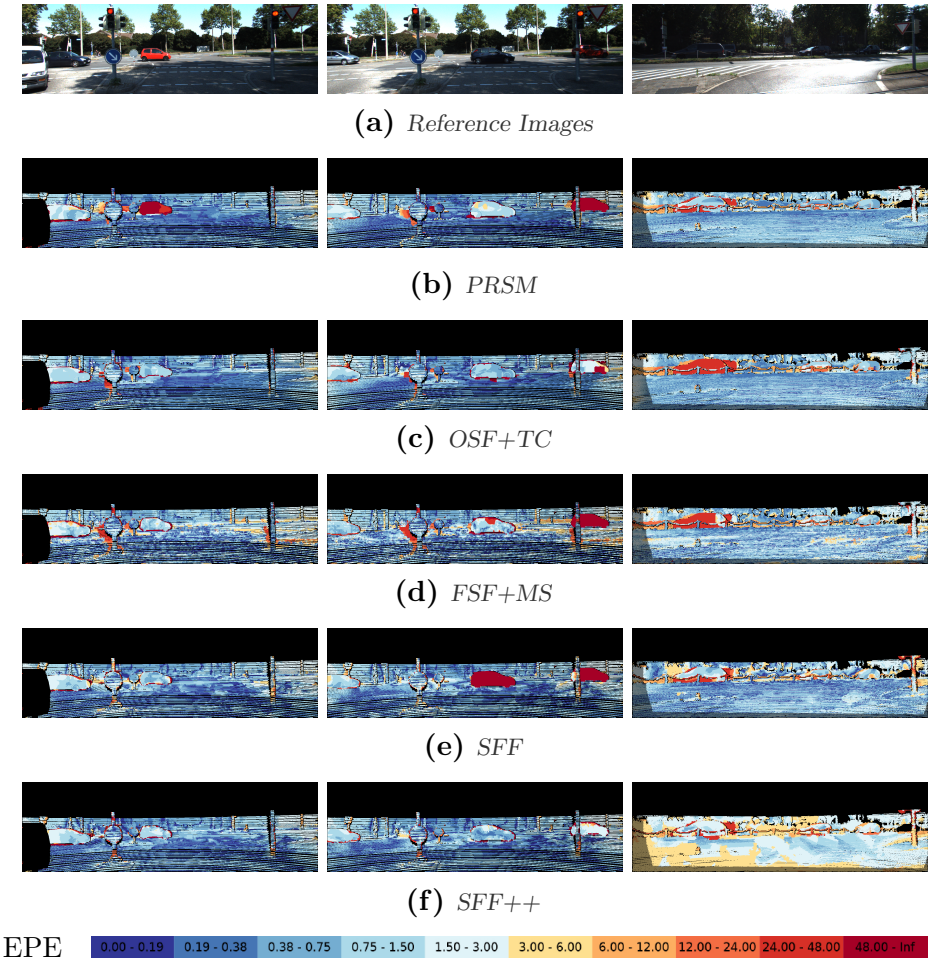



**Figure 4.13.:** *Exemplary visual comparison on KITTI scene flow benchmark [MG15]. Disparity* **(a)** *and optical flow* **(c)** *results along with the corresponding error maps* **(b)** *and* **(d)** *are shown for PRSM [VSR15], OSF [MG15] and SceneFlowFields (SFF). Moving objects are reliably detected and sharp boundaries are reconstructed precisely. More examples are visualized on the public homepage of KITTI.*




**(a)** *Reference Images*

**(b)** *PRSM*

**(c)** *OSF+TC*

**(d)** *FSF+MS*

**(e)** *SFF*

**(f)** *SFF++*

**Figure 4.14.:** *Comparison of the scene flow error on the public KITTI scene flow benchmark [MG15] for Piece-wise Rigid Scene Model (PRSM) [VSR15], OSF+TC [MHG18], FSF+MS [TSS17], and the methods presented in this chapter, SFF [SWKB+18] and SFF++ [SWUK+20].*

**Table 4.5.:** *Results on the KITTI scene flow benchmark [MG15]. The column* multi *indicates whether only two frame pairs are used by this method. Run times in parentheses are obtained on a GPU. SFF achieved the third best result among all dual-frame methods at time of publication and yields especially good results at foreground regions (SF-fg).*

| | | | D1 | | | D2 | | | Fl | | | SF | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | **Year** | **multi** | **bg** | **fg** | **all** | **bg** | **fg** | **all** | **bg** | **fg** | **all** | **bg** | **fg** | **all** | **Run time** |
| ISF [BJMA+17] | 2017 | | 4.12 | **6.17** | 4.46 | **4.88** | **11.34** | **5.95** | 5.40 | **10.29** | **6.22** | **6.58** | **15.63** | **8.08** | 600 s |
| PRSM [VSR15] | 2015 | yes | **3.02** | 10.52 | **4.27** | 5.13 | 15.11 | 6.79 | **5.33** | 13.40 | 6.68 | 6.61 | 20.79 | 8.97 | 300 s |
| OSF+TC [NŠ17] | 2017 | yes | 4.11 | 9.64 | 5.03 | 5.18 | 15.12 | 6.84 | 5.76 | 13.31 | 7.02 | 7.08 | 20.03 | 9.23 | 3000 s |
| OSF18 [MHG18] | 2018 | | 4.11 | 11.12 | 5.28 | 5.01 | 17.28 | 7.06 | 5.38 | 17.61 | 7.41 | 6.68 | 24.59 | 9.66 | 390 s |
| SSF [RSKS17] | 2017 | | 3.55 | 8.75 | 4.42 | 4.94 | 17.48 | 7.02 | 5.63 | 14.71 | 7.14 | 7.18 | 24.58 | 10.07 | 300 s |
| OSF [MG15] | 2015 | | 4.54 | 12.03 | 5.79 | 5.45 | 19.41 | 7.77 | 5.62 | 18.92 | 7.83 | 7.01 | 26.34 | 10.23 | 3000 s |
| SFF++ [SWUK+20] | 2020 | yes | 4.27 | 12.38 | 5.62 | 7.31 | 18.12 | 9.11 | 10.63 | 17.48 | 11.77 | 12.44 | 25.33 | 14.59 | 78 s |
| FSF+MS [TSS17] | 2017 | yes | 5.72 | 11.84 | 6.74 | 7.57 | 21.28 | 9.85 | 8.48 | 25.43 | 11.30 | 11.17 | 33.91 | 14.96 | **2.7 s** |
| CSF [LBAL+16] | 2016 | | 4.57 | 13.04 | 5.98 | 7.92 | 20.76 | 10.06 | 10.40 | 25.78 | 12.96 | 12.21 | 33.21 | 15.71 | 80 s |
| SFF [SWKB+18] | 2018 | | 5.12 | 13.83 | 6.57 | 8.47 | 21.83 | 10.69 | 10.58 | 24.41 | 12.88 | 12.48 | 32.28 | 15.78 | 65 s |
| PRSF [VSR13] | 2013 | | 4.74 | 13.74 | 6.24 | 11.14 | 20.47 | 12.69 | 11.73 | 24.33 | 13.83 | 13.49 | 31.22 | 16.44 | 150 s |
| SGM+SF [HFR14; Hir08] | – | | 5.15 | 15.29 | 6.84 | 14.10 | 23.13 | 15.60 | 20.91 | 25.50 | 21.67 | 23.09 | 34.46 | 24.98 | 2700 s |
| PCOF-LDOF [DPSL16] | 2016 | | 6.31 | 19.24 | 8.46 | 19.09 | 30.54 | 20.99 | 14.34 | 38.32 | 18.33 | 25.26 | 49.39 | 29.27 | 50 s |
| PCOF+ACTF [DPSL16] | 2016 | | 6.31 | 19.24 | 8.46 | 19.15 | 36.27 | 22.00 | 14.89 | 60.15 | 22.43 | 25.77 | 67.75 | 32.76 | (0.08 s) |

is predicted correctly with sharp boundaries around the traffic sign. Within the third example, larger parts of the background exceed the error threshold slightly due to the violated constant motion assumption. However, dynamic objects are still detected more reliably than in most other approaches.

### 4.3.3. MPI Sintel

One claim in this thesis is that the sparse-to-dense concept is very versatile and not restricted to a specific setup. Therefore, SFF and SFF++ are additionally evaluated on MPI Sintel [BWSB12]. Sintel has a lot of contrary properties to KITTI. Most prominently, MPI Sintel consists of non-realistic, synthetically rendered images. Further, images are captured from a totally different domain and have therefore different characteristics. Sintel contains small as well as very large motion displacements of deformable, articulated characters. Thus, many of the included motions are non-rigid and the geometries are less often planar, compared to KITTI. It is important to understand, that the parameters for the experiments on Sintel are the same as for KITTI, the edge detector for SFF being the only exception. On Sintel, for the dual-frame method SFF, the original *bsds* detector [DZ13] is used. For SFF++, the set of parameters is identical on both data sets. Both, the basic approach and the ego-motion extension (+ego), are tested for SFF. For all but two training sequences, every frame with a subsequent frame is processed. The *final* rendering pass for all images is used. Outlier rates as defined by the KITTI metric for disparity *D1-all* and optical flow *Fl-all* are measured. The change in disparity, i.e. the disparity at the future time step, is not available in the Sintel data set and is therefore not evaluated. The sequences *cave_2* and *sleeping_1* are left out in all experiments because they have not been evaluated in FSF [TSS17] due to varying camera parameters. The relative amount of outliers over all evaluated sequences is given in Table 4.6 and is compared to [MG15; TSS17; VSR15] using the results published by [TSS17].

The sparse-to-dense concept can keep up with state-of-the-art scene flow methods, although parameters have not been tuned for MPI Sintel. For sequences with close-up, non-rigid motion, e.g. *ambush_7* or *bandage_1*, the estimated depth even beats the multi-frame scene flow method that is ranked first on KITTI (PRSM). The contradicting properties of Sintel and KITTI are also reflected in the results for some approaches. OSF [MG15], that is strongly relying on the piece-wise rigid plane model, performs considerably worse than on KITTI. Same is assumed for methods that rely on specialized deep neural networks, e.g. [BJMA+17; RSKS17].

SFF++ outperforms the less robust dual-frame version SFF, again highlighting the improvements. It even achieves the best accuracy for disparity by quite a margin to the next best method FSF [TSS17] and the best performing method on KITTI, PRSM [VSR15]. By that, the sparse-to-dense concept joins the few approaches with top performance on both data sets next to PRSM [VSR15], and FSF [TSS17].

Figure 4.15 shows exemplary results of SFF++ for one frame of the sequences *alley_2* and *ambush_5*. The latter can be considered particularly challenging

**Table 4.6.:** *Results on MPI Sintel [BWSB12]. Average outliers for disparity and optical flow on each sequence separately and averaged over all sequences are given. The first row indicates whether a specialized set of parameters is used in the evaluation.*

| | Disparity | | | | | Optical Flow | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Sequence** | PRSM | OSF | FSF | SFF | SFF++ | PRSM | OSF | FSF | SFF | SFF +ego | SFF++ |
| Tuned | yes | yes | yes | (yes) | no | yes | yes | yes | yes | (yes) | no |
| Average | 15.99 | 19.84 | 15.35 | 18.15 | **13.60** | **13.70** | 28.16 | 18.32 | 29.24 | 22.20 | 18.47 |
| alley_1 | 7.43 | 5.28 | 5.92 | 8.81 | **3.98** | **1.58** | 7.33 | 2.11 | 5.94 | 3.95 | 2.11 |
| alley_2 | **0.79** | 1.31 | 2.08 | 1.73 | 1.27 | 1.08 | 1.44 | 1.20 | 2.85 | **0.87** | 1.01 |
| ambush_2 | 41.77 | 55.13 | 36.93 | 51.72 | **31.56** | **51.33** | 87.37 | 72.68 | 90.92 | 83.84 | 76.00 |
| ambush_4 | 24.09 | 24.05 | 23.30 | 37.78 | **22.25** | **41.99** | 49.16 | 45.23 | 60.03 | 42.65 | 61.88 |
| ambush_5 | 17.72 | 19.54 | 18.54 | 25.52 | **13.48** | 25.23 | 44.70 | **24.82** | 46.92 | 29.86 | 32.96 |
| ambush_6 | 29.41 | 26.18 | 30.33 | 37.13 | **23.17** | **41.98** | 54.75 | 44.05 | 57.06 | 47.65 | 59.26 |
| ambush_7 | 35.07 | 71.58 | 23.47 | **16.34** | 24.62 | **3.35** | 22.47 | 27.87 | 13.66 | 7.35 | 9.99 |
| bamboo_1 | **7.34** | 9.71 | 9.67 | 14.53 | 10.80 | **2.41** | 4.04 | 4.11 | 6.11 | 4.15 | 3.44 |
| bamboo_2 | **17.06** | 18.08 | 19.27 | 19.89 | 18.90 | 3.58 | 4.86 | 3.65 | 5.84 | 3.97 | **3.57** |
| bandage_1 | 21.22 | 19.37 | 20.93 | **16.42** | 17.46 | **3.30** | 18.40 | 4.00 | 3.82 | 4.03 | 4.10 |
| bandage_2 | 22.44 | 23.53 | 22.69 | 21.77 | **16.80** | **4.06** | 13.12 | 4.76 | 10.72 | 9.06 | 4.56 |
| cave_4 | **4.27** | 5.86 | 6.22 | 6.20 | 4.93 | 16.32 | 33.94 | 14.62 | 15.63 | **12.95** | 18.16 |
| market_2 | **5.27** | 6.61 | 6.81 | 6.71 | 6.26 | **4.77** | 10.08 | 5.17 | 7.11 | 6.09 | 5.51 |
| market_5 | 15.38 | 13.67 | **13.25** | 26.66 | 14.13 | 28.38 | 29.58 | **26.31** | 40.77 | 28.87 | 32.56 |
| market_6 | **8.99** | 10.29 | 10.63 | 14.53 | 10.18 | **10.72** | 16.39 | 13.13 | 28.92 | 16.69 | 13.91 |
| mountain_1 | 0.42 | 0.78 | 0.23 | 0.15 | **0.02** | **3.71** | 88.60 | 17.05 | 90.60 | 89.57 | 10.84 |
| shaman_2 | 25.49 | 28.27 | 24.77 | **21.13** | 23.94 | **0.46** | 1.67 | 0.56 | 8.85 | 4.31 | 1.80 |
| shaman_3 | 33.92 | 52.22 | **27.09** | 35.37 | 29.02 | 1.75 | 11.45 | **1.31** | 15.91 | 8.51 | 5.53 |
| sleeping_2 | **1.74** | 2.97 | 3.52 | 3.07 | 2.24 | **0.00** | 0.01 | 0.02 | 0.61 | 0.03 | **0.00** |
| temple_2 | **4.92** | 5.54 | 5.96 | 6.98 | 4.95 | **9.51** | 10.52 | 9.66 | 29.58 | 12.57 | 12.05 |
| temple_3 | 11.04 | 16.62 | 10.65 | 8.61 | **5.76** | 32.10 | 81.39 | 62.34 | 72.28 | 49.18 | **28.64** |

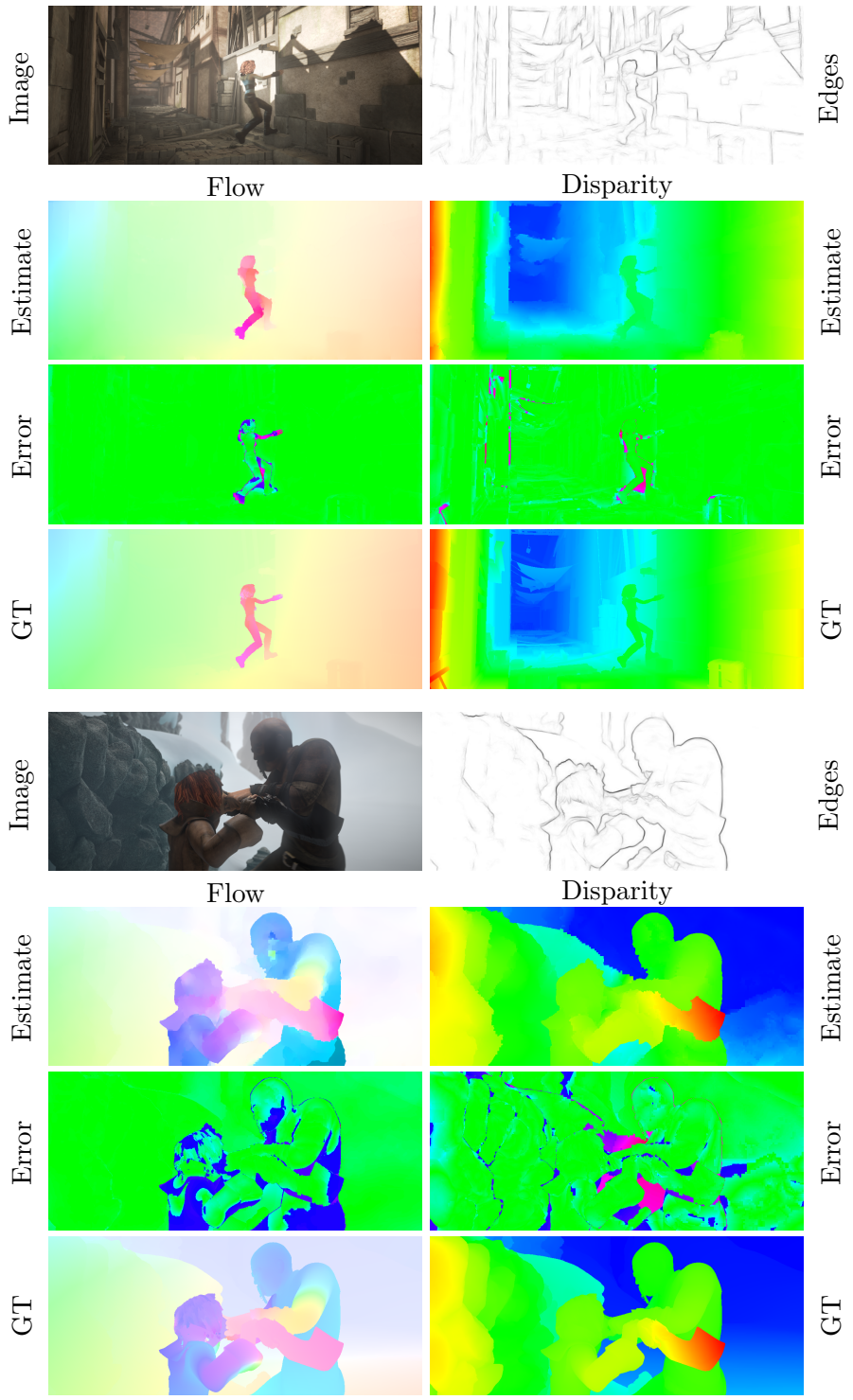



**Figure 4.15.:** *Exemplary results of SFF++ on the Sintel data set [BWSB12].*

(cf. Table 4.6). Figure 4.15 also demonstrates that a bit of accuracy is traded in for increased robustness. Small details, e.g. the arm of the girl, are lost during interpolation.

## 4.4. Summary

Within this chapter, the sparse-to-dense concept is successfully applied for the problem of scene flow estimation in automotive driving scenarios. Dense matching across multiple images followed by a consistency check for filtering is used to obtain highly accurate, sparse matches. Different models for interpolation are tested, the more continuous variant (*EPIC3D*) and the more robust, patch-wise interpolation (*RIC3D*). Both follow the same principle to split the densification into interpolation of geometry, followed by interpolation of 3D motion. An important aspect during interpolation is the preservation of discontinuities, which is obtained by adhering to (partially semantic) image boundaries that serve as a reliable proxy for geometric and motion boundaries. The initially required refinement steps (optimization and ego-motion estimation) become unnecessary when the more robust interpolation and multi-frame matching with visibility reasoning is applied. In all cases, competitive accuracy with increased run time and better generalization is achieved.

### 4.4.1. Limitations

Despite all efforts, it is not possible to avoid any assumption during estimation of scene flow due to the ill-posedness of the problem. However, the sparse-to-dense approaches are designed with as little restrictions as possible. The initial matching is purely data-based and requires no assumptions on smoothness at all. The later imposed regularization during interpolation makes use of the piece-wise rigid plane model but applies it to an extreme over-segmentation of the scene into tiny superpixels of 25 pixels to reduce the negative effects greatly. Further, to make use of additional image information from multiple time steps, SFF++ assumes a constant motion within a temporal window of three stereo frames. Hence, SFF++ is subject to a set of mostly theoretical limitations that are described here.

As discussed several times, SFF++ remains unaffected by the drawbacks of a piece-wise rigid, planar motion assumption even though this model is used during interpolation. The reason is the size of the superpixel segments. An impression of the size of the superpixels is give in Figure 4.9. During all experiments, it was never observed that the superpixels are the limiting factor in the estimation of non-planar surfaces or non-rigid motions. However in theory, the small superpixels still introduce a small error.

Another hypothetical failure case arises from the complete separation of matching and regularization. If unregularized matching leads to the removal of entire image regions during the consistency check, the later imposed regularization of the interpolation can not recover the content of these regions. This phenomenon was regularly observed for the dual-frame approach, e.g. when

highly dynamic objects leave the image domain. At the same time, these cases were one of the motivations for the shift to multiple frames. In the multi-frame scenario, it is much less likely that matching fails consistently for entire regions. This is also supported by the study of visible areas in Figure 4.8a.

Lastly, the constant motion assumption is the only assumption-based limitation that causes practical impact on the performance of SFF++. For the KITTI data, some degradation of the estimated scene flow is noticed due to the violation of this assumption. One of these examples is shown in the right column of Figure 4.14. Anyway, this problem was encountered rarely, mostly in the presence of potholes or crossing rails that lead to an unexpectedly high rotational acceleration which is amplified by the limited frame rate of KITTI.

### 4.4.2. Next Steps

For immediate improvement of the sparse-to-dense concept, the recent advances in deep learning motivate to replace parts of the pipeline by dedicated neural networks. Therefore, the next chapter follows this direction for two specific parts.

Chapter 5

# Learning-based Replacements for the Sparse-to-Dense Pipeline

*"On ne connaît que les choses que l'on apprivoise."*
— Antoine de Saint-Exupéry, *Le Petit Prince*

## Contents



This chapter is backed by the following publications:

**René Schuster**, Oliver Wasenmüller, Christian Unger, and Didier Stricker. "SDC – Stacked Dilated Convolution: A Unified Descriptor Network for Dense Matching Tasks." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019. **Oral**.

**René Schuster**, Oliver Wasenmüller, Christian Unger, and Didier Stricker. "An Empirical Evaluation Study on the Training of SDC Features for Dense Pixel Matching." In: *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019.

**René Schuster**, Oliver Wasenmüller, Christian Unger, and Didier Stricker. "SSGP: Sparse Spatial Guided Propagation for Robust and Generic Interpolation." In: *Winter Conference on Applications of Computer Vision (WACV)*. 2021.

---

The modularity of the sparse-to-dense concept allows to replace parts of the pipeline individually. This has been done in the previous experiments, e.g. by replacing the matching cost, or by interchanging the interpolation algorithm. In continuation of this, more components are replaced to further improve the method and to eliminate remaining limitations. In more detail, advances in deep machine learning are employed to learn a better feature representation of pixels from data, and to unify the problem of sparse-to-dense interpolation.

One very important aspect here is that imagery with scene flow annotations is very scarce. Apart from the data sets discussed in Section 2.3.1, no labeled data exists. Furthermore, FlyingThings3D (FT3D) is a non-realistic, synthetic data set from a domain which differs a lot from the target application. This leaves a total amount of 200 realistic sequences with annotations for scene flow. Therefore, training a network to directly solve the scene flow problem for the application domain is infeasible. Instead, encapsulated components are replaced by data-driven modules which allows to train these dedicated networks with more and more diverse data. Apart from solving the issue of lacking data, another major advantage is that the generalization is improved by using multi-domain data, working towards the goal of robustness.

## 5.1. Deep Pixel Representations for Dense Matching

Scene flow estimation is ultimately a dense matching problem. Robust dense matching of pixel positions under unconstrained conditions typically is a very challenging task for several reasons. Perspective deformations, changing lighting conditions, sensor noise, occlusions, and other effects can change the appearance of corresponding image points drastically. Thus, heuristic descriptors (e.g. SIFT [Low99] or CENSUS [ZW94]) can produce very dissimilar descriptors for corresponding image points. A key factor to overcome these issues is the size of context information that is considered by a descriptor. However, increasing the patch size introduces spatial invariance for state-of-the-art descriptors, which results in less accurate matching. Recently, deep neural networks have been shown to produce more robust and expressive features [BVS17; TFW+17; ZL15]. These networks rely on best practice design decisions from other domains, which results in the use of pooling or other striding layers. Such architectures typically achieve a medium sized receptive field only and reduce the spatial resolution of the resulting feature descriptor. Both properties lower the accuracy for the matching task.

To overcome all these limitations, a deep neural network with a large receptive field is proposed that utilizes a novel architecture block to compute highly robust, accurate, dense, and discriminative descriptors for images. Towards this end, dilated convolutions are stacked in parallel. The design follows two key

observations. Firstly, image patches with low entropy lead to poor descriptors and thus to incorrect matching. This fact strengthens the common belief that a robust descriptor should have a large receptive field to incorporate context knowledge for pixels under difficult visual conditions. Secondly, accurate matching requires a high spatial precision which is lost when striding layers, producing coarse, high-level features for deeper layers of the feature network, are applied. The proposed architecture block provides a large receptive field with only few trainable parameters while maintaining full spatial resolution. This leads to the following overall contributions:

- Stacking multiple, parallel dilated convolutions (SDC) to create a novel neural network block, which is beneficial for any dense, pixel-wise prediction task that requires high spatial accuracy.
- Combining these blocks into a fully convolutional architecture with a large receptive field that can be used for feature description.
- Vast sets of experiments to justify the design decisions, to compare to other descriptors, and to demonstrate the accuracy and robustness for scene flow, optical flow, and stereo matching on the well known public data sets KITTI [MG15], MPI Sintel [BWSB12], Middlebury [BSLR+11; SS02], HD1K [KNHK+16], and ETH3D [SSGS+17] with the unified network.

### 5.1.1. Feature Description in the Literature

A feature descriptor is a vector that represents the characteristics of the associated object in a compact, distinctive manner. It is not to be confused with an interest point (or key point, sometimes feature point) which identifies locations where a feature descriptor would be rather unique. In the context of dense matching, feature descriptors on pixel-level are required. Since single pixels carry only very little information, usually a region around each pixel is considered for the description.

Conventional descriptors are often based on image gradients to make them invariant to changes in lighting. A very common descriptor – SIFT [Low99] – computes histograms of gradients in regular grids around the center pixel. Using a multi-scale search and the major orientation of the gradients makes SIFT robust to changes in scale and rotation. However, SIFT is not designed to describe all pixels of an image in a dense manner. The full description is rather slow and sensitive to deformations, occlusions, and motions. Robustness is also a problem for faster hand-crafted feature extractors like SURF [BTV06] and DAISY [TLF10]. Binary descriptors (e.g. BRIEF [CLSF10], ORB [RRKB11], or CENSUS [ZW94]) are even faster since they are more compact. At the same time, they are less expressive and less distinctive.

To improve robustness, many approaches apply deep learning for feature extraction on patch-level. In [HLJS+15; ZK15], features are learned jointly with a decision metric to distinguish corresponding and non-matching image patches with a siamese architecture [CHL05]. For the same reason as L2Net [TFW+17], the proposed architecture does not include a decision network because the features are supposed to be used within any pipeline. The architecture of

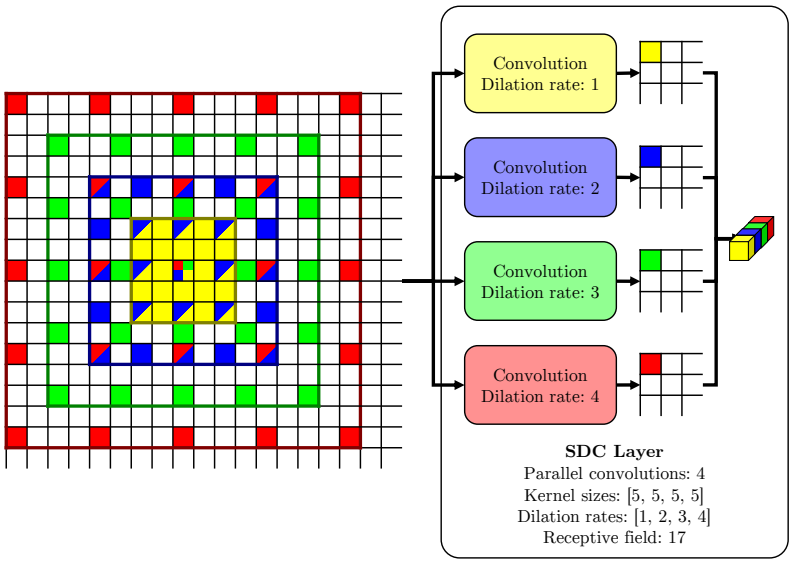



**Figure 5.1.:** *The architecture of a single SDC layer. The contribution is the combination of parallel convolutions with different dilation rates. The outputs are stacked along the feature dimension to produce a multi-scale response.*

L2Net [TFW+17] avoids pooling layers but requires strided convolution to achieve a medium sized receptive field of 32 pixels. Additionally, it has been experimented with a two-stream design where the input of the second branch is the up-scaled central part of the original patch similar as in [ZK15]. In contrast, the Stacked Dilated Convolution (SDC) architecture exploits multi-scale information inherently as described in Section 5.1.2.

Deep features for the optical flow task are proposed by [BVS17; GW16]. PatchBatch [GW16] introduces batch normalization for patch description for the first time, and FlowFieldsCNN[BVS17] utilizes a new thresholded hinge loss. Both architectures consist of several convolutions and pooling layers to obtain considerably large receptive fields. As motivated earlier, the SDC design can easily increase the size of the receptive field without losing the spatial accuracy as it happens during pooling.

For stereo matching, existing work uses very light-weight architectures with small receptive fields in favor of speed [LSU16; ZL15]. For the limited search in stereo matching, the expressiveness of these networks might be sufficient. In contrast, a universal descriptor network for different tasks and domains is supposed to use more context information.

### 5.1.2. SDC Feature Network

Historically, a large receptive field in convolutional neural networks (CNNs) is primarily obtained by using striding layers. These are typically pooling layers and more recently, pooling has been replaced by strided convolution [SDBR15]. Striding layers also improve run time by reducing the size of intermediate

representations and introduce some translational invariance. For tasks like image classification, these benefits come at no cost since only a single prediction per image is required. For tasks which require a dense per-pixel prediction, strided layers have the disadvantage of reducing the spatial resolution. This makes pixel-wise prediction overly smooth and less accurate.

The obvious way to obtain a large receptive field without striding is to use larger kernels. The drawbacks of this approach are a drastic increase in run time and number of parameters which makes such networks slow and prone to overfitting. This problem can be surpassed by dilated convolution because even though the kernels are large, they are sparse (in a regular way). Yet, a sequence of dilated convolutions can introduce gridding effects (different output nodes use disjoint subsets of input nodes) if dilation rates are not selected properly [WCYL+18]. As a consequence, SDC is proposed, which applies multiple dilated convolution in parallel and concatenates the outputs. This way, each subsequent layer has full access to previous features of different dilation rates.

**SDC Layer.** As in previous works [CPKM+18; LZC18], is is argued that convolution with dilation rate $r$ and stride $r$ is equal to convolution with dilation rate 1 (no dilation) of sub-sampled input by factor $r$ (no smoothing). Dilated convolution without striding thus produces a sub-scale response at full spatial resolution. This key observation is exploited by the SDC design in which the output of convolutions with different dilation rates are stacked to produce a multi-scale response (see Figure 5.1). Whereas others apply pooling over multiple scales, in SDC the entire multi-scale information is passed to the next layers.

It is noted that convolution with parallel dilated kernels is similar to convolution with a single larger, sparse kernel (merging the dilated kernels). However, expressiveness is lost where the different dilated kernels overlap (see Figure 5.6). Further, only very few deep learning frameworks support sparse convolution in an efficient way. Nonetheless, an experimental comparison between both designs is provided in the experiments.

**Feature Network.** Following the interpretation of dilated convolution, several SDC layers are sequentially stacked to compute, aggregate, and pass information for multiple scales from end to end. This naturally results in an exponentially growing receptive field, but avoids gridding effects because every convolution is fed with the results of every previous convolution of all dilation rates. Five such SDC layers are used. Each SDC layer applies four parallel convolutions with $5 \times 5$ kernels, the same number of output dimensions, and dilation rates of $1, 2, 3,$ and $4$. Exponential Linear Unit (ELU) [CUH16] is used for all activations. Batch normalization is not applied, because the network is trained with a small batch size (cf. Section 5.1.3). The SDC layers have $64, 64, 128, 256,$ and $128$ output channels, respectively. The final feature vector of the last layer is normalized to unit range, i.e. values in $[-1, 1]$. The experiments in Section 5.1.4 justify the decision for this design. The overall setup yields a receptive field of 81 pixels, which surpasses competitive work.

Because striding is not applied, dense image features can be computed in a single forward pass without patch extraction. This makes the design much faster than previous deep descriptors [BVS17; GW16; TFW+17] during inference.

The design provides another advantage that can be used within SDC layers: The same kernels are useful for different scales (especially low level vision filters in the first layers). Thus, it is reasonable to share weights between the parallel convolutions within one SDC block. The only requirement is that the parallel convolutions are of the same shape. By sharing weights, the amount of parameters is divided by the number of parallel convolutions (factor 4 in the above case). This allows to construct very light-weight feature networks with a comparatively large receptive field. To demonstrate that, the network size is driven to an extreme. In the experiments in Section 5.1.4, a second network with only about 5 % of the parameters of the original design is trained and dubbed as *Tiny*. The *Tiny* network has only four SDC blocks, each with only three parallel dilated convolutions of $3 \times 3$ kernels and dilation rates $1, 2,$ and $3$ which share their weights, yielding a receptive field of 25 pixels.

### 5.1.3. Training

One goal is to train a universal feature descriptor. Thus, a unified feature network is trained on multi-domain data. Training splits of the following data sets are used: Scene flow quadtuplets of KITTI 2015 [MG15], optical flow and stereo pairs from MPI Sintel [BWSB12], Middlebury stereo data version 3 [SS02], Middlebury Optical Flow data [BSLR+11], HD1K Benchmark Suite for optical flow [KNHK+16], and the two-view stereo data from ETH3D [SSGS+17]. This is the union of data sets which are used in the Robust Vision Challenge 2018[1] for optical flow and stereo.

Since image sizes, sequence count, and sequence lengths vary strongly between data sets, image pairs are sampled non-uniformly. Considering the different characteristics (cf. Section 5.1.5) of each data set, to avoid imbalance and over-representations, the following probabilities are used to sample from each data set:

| | |
|---|---|
| KITTI [MG15]: | 0.5 |
| Sintel [BWSB12]: | 0.175 |
| Middlebury Optical Flow [BSLR+11]: | 0.025 |
| Middlebury Stereo [SS02]: | 0.05 |
| HD1K [KNHK+16]: | 0.175 |
| ETH3D [SSGS+17]: | 0.075 |

Images are not sampled twice until all images of the respective data set are selected, i.e. within each data set, all images are used equally often. For each image in each epoch, 100 randomly sampled reference patches are selected. If a data set provides ground truth for multiple tasks (e.g. KITTI [MG15]), one of the tasks from stereo, optical flow, or scene flow is selected randomly.

[1] www.robustvision.net/rvc2018.php

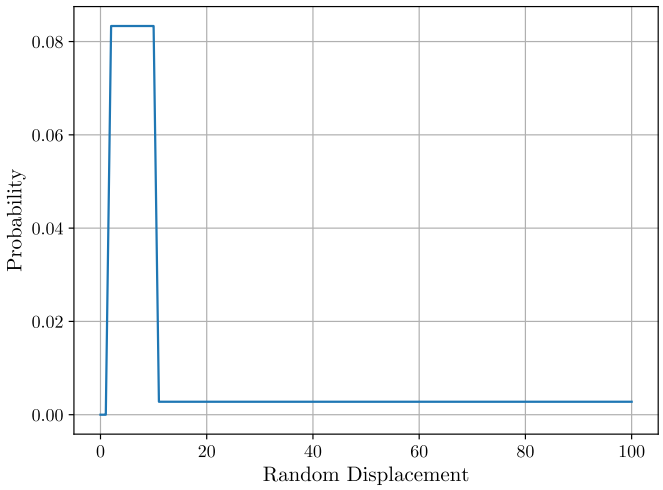



**Figure 5.2.:** *Probability distribution of the random offset used to generate the negative patch correspondence.*

The reference patches are sampled from pixel positions where ground truth exists and where the ground truth displacement points to a visible position in the corresponding view, i.e. occlusions and out-of-bound displacements are excluded wherever possible. For each of the 100 reference patches, the corresponding match is extracted according to the ground truth. Images are padded with reflection at image boundaries and bilinear interpolation is used at sub-pixel positions. The negative patch is extracted by altering the ground truth displacement with a random offset. This random offset is at least 2 pixels and at most 100 pixels large in magnitude. For stereo correspondences, the displacement and the random offset are 1-dimensional along the horizontal direction according to the epipolar constraint. Other correspondences have 2-dimensional displacement, i.e. circular around the ground truth correspondence. Since close-by correspondences are harder to distinguish, the random offset is also sampled non-uniformly. In detail, the random offset is split into two ranges, a close one ($[2, 10]$ pixels) and a distant range ($]10, 100]$ pixels). The close range is selected three times more often than the far range and within each range, it is sampled uniformly. This leads to the overall probability distribution for the magnitude of the negative offset shown in Figure 5.2.

Each image is processed completely (all 100 patch triplets) before selecting the next image pair from one of the data sets to reduce IO operations in the training data pipeline. Then, chunks of 3200 triplets are shuffled to reintroduce randomness across data sets and images. Smaller displacements (less than 2 pixels) are not considered for several reasons. Minimal changes in appearance might confuse the network, rounding and interpolation introduce small inaccuracies, and most applications tolerate a matching accuracy less than 2 pixels end-point error. Visual examples of some training triplets are given in Figure 5.3.

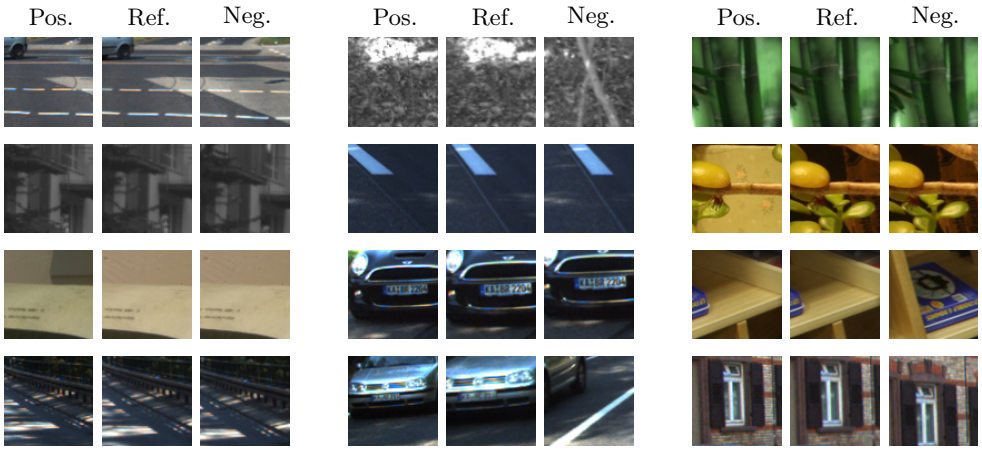



**Figure 5.3.:** *Randomly sampled training triplets. For each reference patch, a positive and negative match are selected according to the ground truth and by adding a random offset to the ground truth.*

The 200 training images from KITTI [MG15], are randomly split into a subset for actual training (70 %), one for validation (20 %), and one for testing in the experiments in Section 5.1.4 (10 %). This is the exact list of sequences for each subset:

- Training: 0, 1, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 18, 19, 20, 21, 22, 24, 25, 27, 28, 29, 30, 31, 33, 34, 35, 37, 38, 39, 40, 41, 43, 44, 45, 47, 49, 50, 51, 54, 55, 56, 58, 59, 62, 63, 66, 67, 68, 70, 71, 74, 75, 76, 78, 79, 82, 83, 85, 86, 87, 88, 89, 90, 93, 95, 96, 97, 99, 101, 102, 103, 104, 105, 107, 109, 111, 113, 114, 116, 117, 118, 120, 123, 125, 128, 129, 130, 132, 134, 135, 137, 138, 139, 140, 141, 143, 144, 145, 147, 148, 149, 150, 151, 152, 154, 155, 156, 157, 160, 161, 162, 163, 164, 165, 167, 168, 169, 170, 171, 172, 175, 177, 178, 179, 180, 182, 183, 185, 187, 188, 189, 191, 194, 195, 197, 198, 199
- Validation: 2, 16, 17, 23, 26, 32, 36, 48, 52, 53, 57, 60, 61, 64, 69, 72, 73, 77, 80, 81, 84, 91, 100, 108, 110, 112, 122, 126, 127, 131, 133, 136, 142, 153, 158, 159, 166, 176, 192, 196
- Testing: 4, 42, 46, 65, 92, 94, 98, 106, 115, 119, 121, 124, 146, 173, 174, 181, 184, 186, 190, 193

A triplet training approach [HA15] is used in which the reference patch, the matching patch and the non-matching patch are fed to three SDC networks with shared weights. For training stability, the input is normalized by subtracting the mean and dividing by the standard deviation of all training images. The mean pixel is [0.3534, 0.3448, 0.3295] and the mean standard deviation is [0.2492, 0.2465, 0.2446] for the red, green, and blue color channels, respectively. As objective function, the thresholded hinge embedding loss of [BVS17] is used and defined in Equation 5.1.

$$\begin{aligned}\mathcal{L}(r,p,n) = &\, max\left(0, \|f(r) - f(p)\|_2^2 - \tau\right) \\ &+ max\left(0, m + \tau - \|f(r) - f(n)\|_2^2\right),\end{aligned} \tag{5.1}$$

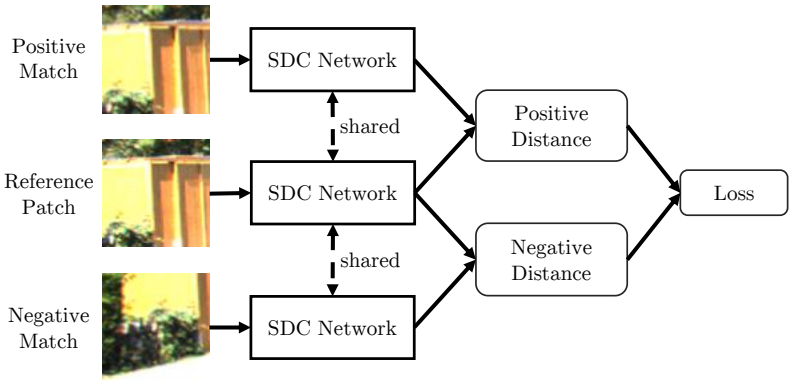


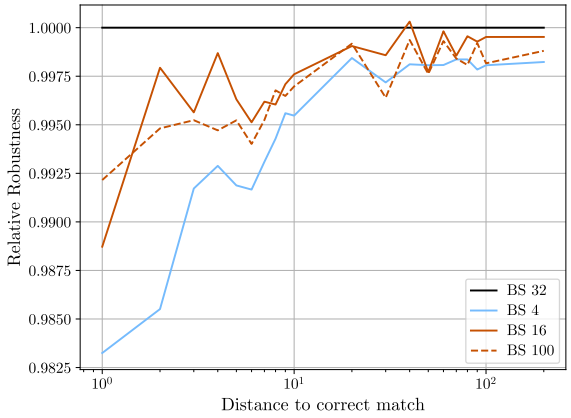

**Figure 5.4.:** *Visualization of the triplet training. For each patch triplet, the loss is computed based on the distance of the feature descriptors for corresponding and non-corresponding patches.*

where $\{r, p, n\}$ is the patch triplet, $f$ is the feature transformation of the network, $\tau$ is the threshold, and $m$ is the margin between matching and non-matching features. The experiments compare this loss to the SoftMax-Triplet loss [HA15] and the SoftPN loss [BJTM16]. Both show a similar performance while being less stable during training. An overview of the training strategy is given in Figure 5.4.

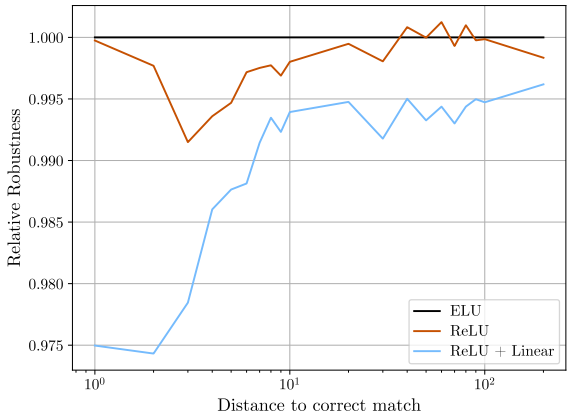
Adam [KB15] is the optimizer of choice. The network is trained with a batch size of 32 and with an initial learning rate of 0.01 that is exponentially decreased by a power of 0.7 every 100k iterations, continuously. Training is performed for one million iterations after which the convergence saturates, or until the point of overfitting, which is rarely observed in any of the experiments. Overfitting is avoided by the random sampling strategy of image pairs and patch triplets, which provides many diverse combinations. Photometric data augmentation can not further improve the training process. Instead, a small decrease in performance is noted. To speed up training, the input patches and intermediate feature representations are cropped to the maximum required size for the respective dilation rate. The complete training of the SDC network takes about three days on a single GeForce GTX 1080 Ti.

### 5.1.4. Experiments and Results

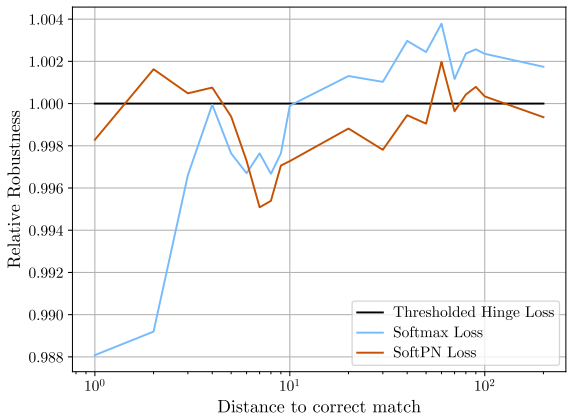
A series of diverse experiments is conducted to validate the design decision, the superior performance of the proposed approach compared to other feature descriptors, and to test SDC features with different algorithms for different matching tasks on a large number of diverse data sets. For all experiments, a single unified descriptor network with a unique set of trained weights is used. Unlike other networks [BVS17; SYLK18], SDC is not re-trained or fine-tuned on each individual data set.




**(a)** *Batch sizes (BS).*




**(b)** *Activation functions.*




**(c)** *Triplet loss functions.*

**Figure 5.5.:** *Validation of design decisions for the SDC network by comparing the relative robustness of different configurations.*

**Table 5.1.:** *Comparison of the accuracy for representative state-of-the-art descriptors and the SDC design. For learning approaches, further information about receptive field size (RF) in pixels, number of parameters (Size), and accumulated sub-sampling factor due to striding are provided.*

| Network | Accuracy | RF | Size | Factor |
|---|---|---|---|---|
| **SDC** | **97.2 %** | 81 | 1.95 M | **1** |
| Fake-big | 97.0 % | 81 | 6.3 M | 1 |
| LargeNet | 96.8 % | 81 | 22.5 M | **1** |
| L2Net [TFW+17] | 96.7 % | 32 | 1.34 M | 4 |
| **Tiny** | 96.0 % | 25 | **0.12 M** | **1** |
| Fake-small | 96.0 % | 81 | 0.4 M | 1 |
| PatchBatch [GW16] | 95.7 % | 51 | 0.92 M | 8 |
| DilNet | 95.5 % | **96** | 5.43 M | **1** |
| 2Stream [ZK15] | 92.3 % | 64 | 2.41 M | 2 |
| FFCNN [BVS17] | 90.6 % | 56 | 4.89 M | 4 |
| BRIEF [CLSF10] | 93.7 % | – | – | – |
| DAISY [TLF10] | 92.1 % | – | – | – |
| SIFT [Low99] | 89.0 % | – | – | – |

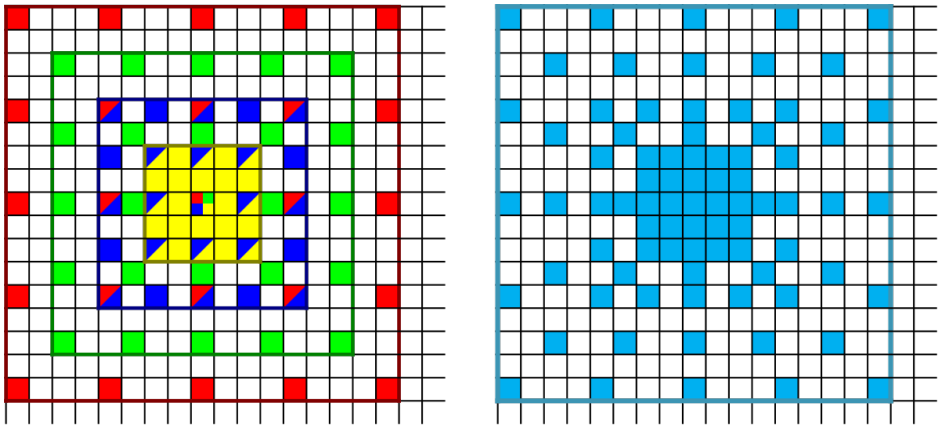
### Design Decisions

Variation of depth and width of the SDC network structure is covered by the *Tiny* version. In the following, batch size, activation functions, and loss functions are varied. One parameter is altered at a time and compared to the original design. For better comparison, relative robustness is introduced which is the robustness ratio of a model and a reference model. A relative robustness greater than 1 means that the model is better than the reference model. The original SDC network is used as the reference which will result in a baseline of 100 % of relative robustness for this model.

Results for different batch sizes are given in Figure 5.5a. The impact of the batch size is minor. Even for very small mini batches, the relative robustness drops by less than 2 percentage points. Increasing the batch size is not improving the performance either.

For alternative activation functions, Rectified Linear Unit (ReLU) [NH10] instead of Euclidean Linear Unit (ELU) [CUH16] is considered. Since the final feature vectors are normalized to unit range, the rectification of ReLU restricts the feature space to the non-negative orthant of the 128-dimensional hypercube which is only $2^{-128}$ of the full volume. Therefore, a network with ReLU and linear activation in the last layer is also trained and compared. The comparison is shown in Figure 5.5b.

As mentioned earlier, instead of triplet training with a thresholded hinge embedding loss [BVS17], the softmax loss [HA15] and the PN loss [BJTM16] are also tested (see Figure 5.5c). Both alternative losses perform better for some displacements, and worse for others. However, the difference in robustness is small and training with these real triplet losses is less stable enforcing a lower learning rate and thus an increased overall training time.



SDC Design | Single Sparse Kernel

**Figure 5.6.:** *Two variants of sparse* 17×17 *kernels: Four parallel dilated* 5×5 *kernels (left), and a single kernel (right).*

#### Accuracy, Robustness, ROC

In this section, the SDC descriptor network is compared to other descriptors. Representative classical, heuristic descriptors are the Scale-Invariant Feature Transform (SIFT) [Low99], DAISY [TLF10], and Binary Robust Independent Elementary Features (BRIEF) [CLSF10]. Furthermore, the following architectures of previous work that contain striding layers are trained:

- *2Stream*: The central-surround network from [ZK15].
- *PatchBatch*: The architecture of [GW16], which utilizes batch normalization.
- *L2Net*: The basic variant of [TFW+17] with only a single stream and without batch normalization, which is found to perform the best among all variants of this network.
- *FFCNN*: The FlowFieldsCNN architecture [BVS17], which showed great improvements over classical descriptors for optical flow estimation.

In addition, two alternative architectures that avoid striding layers are designed and evaluated.

- *DilNet*: An example of dilated convolution in a sequence: Conv(7,64,1,1) – Conv(7,64,1,2) – Conv(7,128,1,3) – Conv(7,128,1,4) – Conv(7,128,1,3) – Conv(7,256,1,2) – Conv(7,128,1,1).
- *LargeNet*: An example for single, large convolutions without dilation: Conv(17,64,1,1) – Conv(17,64,1,1) – Conv(17,128,1,1) – Conv(17,256,1,1) – Conv(17,128,1,1).

The four numbers of each convolution layer Conv($k$,$n$,$s$,$d$) describe square kernel size $k$, number of kernels $n$, stride $s$, and dilation rate $d$. Note that *DilNet* and *LargeNet* try to mimic the shape of the SDC network. More details about each network are given in Table 5.1.

**Large Sparse Convolution.** Multiple dilated convolutions in parallel are similar to a single larger convolution with a sparse kernel (see Figure 5.6). The

difference is that pixels where the parallel kernels overlap are only considered once in a single kernel (for $3 \times 3$ kernels this is the center pixel only) and that a single kernel merges all information into a single output. With parallel convolutions, it is possible to add them, stack them, or combine them as needed. For comparison of both approaches, two alternative variants are designed, which use single, larger convolutions with the exact same receptive field as the SDC network. The first one produces the same output dimensions at each layer, i.e. $64, 64, 128, 256,$ and 128 feature channels. This results in a four times larger network compared to the SDC design approximately (disregarding overlapping pixel positions). The second variant is designed to have the same network size as SDC, which results in 4 times less output channels per layer, i.e. $16, 16, 32, 64,$ and 32. These networks are called *Fake-big* and *Fake-small* respectively, because these networks try to imitate the original SDC design. The complete architectures look like this:

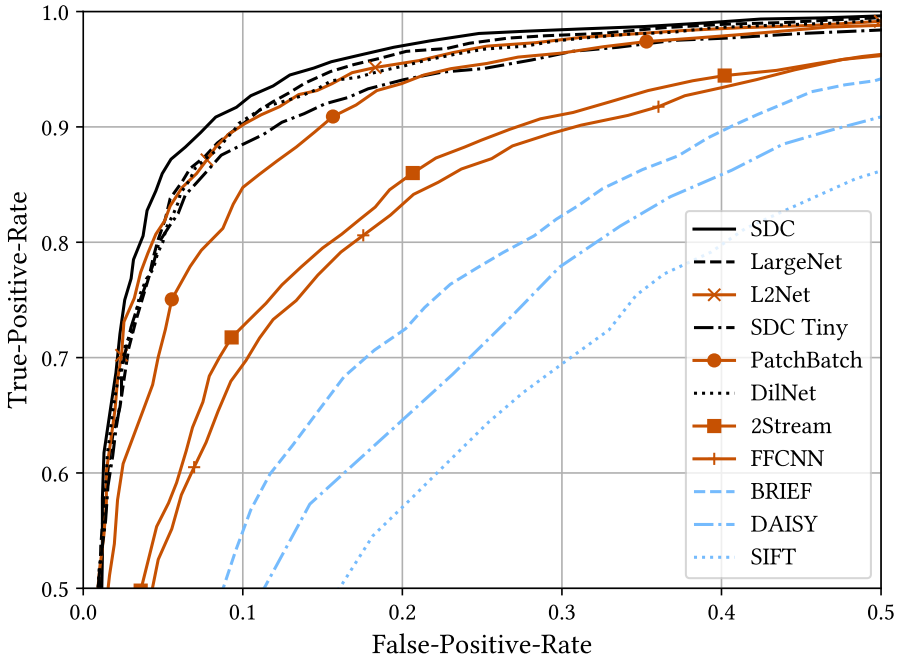
- *Fake-big*: SparseConv(17,64,1,1) – SparseConv(17,64,1,1) – SparseConv(17,128,1,1) – SparseConv(17,256,1,1) – SparseConv(17,128,1,1)
- *Fake-small*: SparseConv(17,16,1,1) – SparseConv(17,16,1,1) – SparseConv(17,32,1,1) – SparseConv(17,64,1,1) – SparseConv(17,32,1,1)

Sparsity is enforced according to the pattern shown in Figure 5.6.

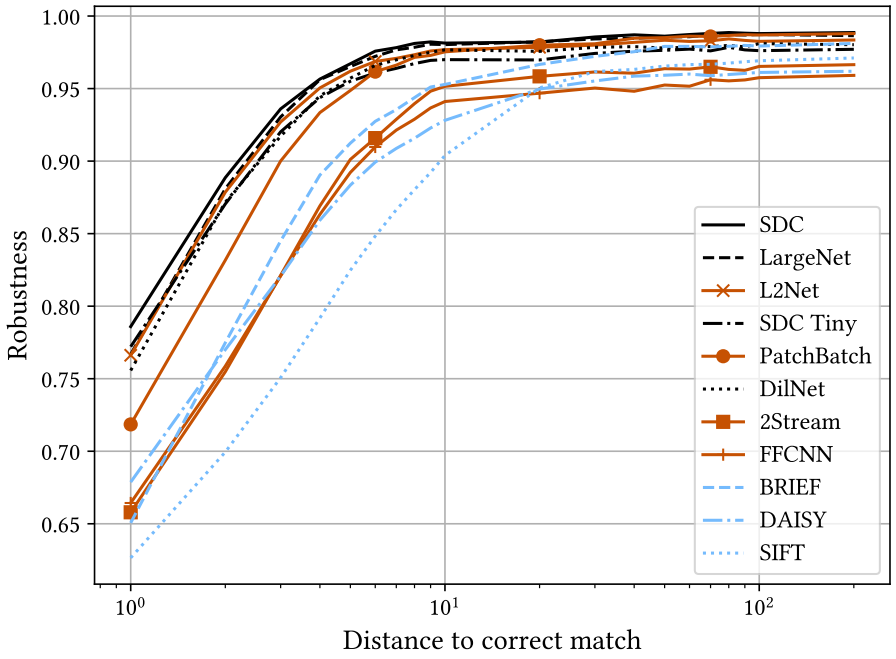
First, the accuracy of all descriptors is evaluated. Accuracy is defined as the percentage of correctly distinguished patch triplets, i.e. the positive feature distance is smaller than the negative one. Towards that end, 2000 patch triplets are sampled from the test images (cf. Section 5.1.3). The results are given in Table 5.1. SDC outperforms all other feature descriptors in terms of accuracy. The receptive field (RF) is comparatively large, while the network size is comparatively small and also sub-sampling is avoided. The *Tiny* version is extremely compact without much loss of accuracy. The surprisingly good result of L2Net [TFW+17] is worth mentioning, indicating that strided convolution should be preferred over pooling. Also, some of the learning approaches perform worse than the classical descriptors. The SDC network outperforms the *Fake* networks as well. Considering that *Fake-big* is a much bigger network, this is even more evidence that the design is very powerful. The concatenation of multi-scale features and the mixture of multi-scale information at every level is beneficial for image description.

The Receiver-Operating-Characteristics (ROC) is also compared for all descriptors based on the same test triplets. Each triplet is split into two pairs, a positive and a negative one. True-Positive-Rates over False-Postive-Rates for varying classification thresholds are given in Figure 5.7a. Again, SDC features achieve top performance with a large margin over heuristic descriptors and most neural networks.

However, matching is not really a classification task. The distance of corresponding descriptors does not matter, as long as it is smaller than these of non-matching descriptors. To take this into account, a final experiment is set up to show the matching robustness of the descriptors as introduced by [BVS17]. Each positive corresponding patch pair of the test data is tested against all other correspondences within a certain distance to the correct




**(a)** *ROC curves.*




**(b)** *Robustness curves.*

**Figure 5.7.:** *In the comparison of ROC and robustness curves, the SDC design outperforms state-of-the-art feature networks and heuristic descriptors.*

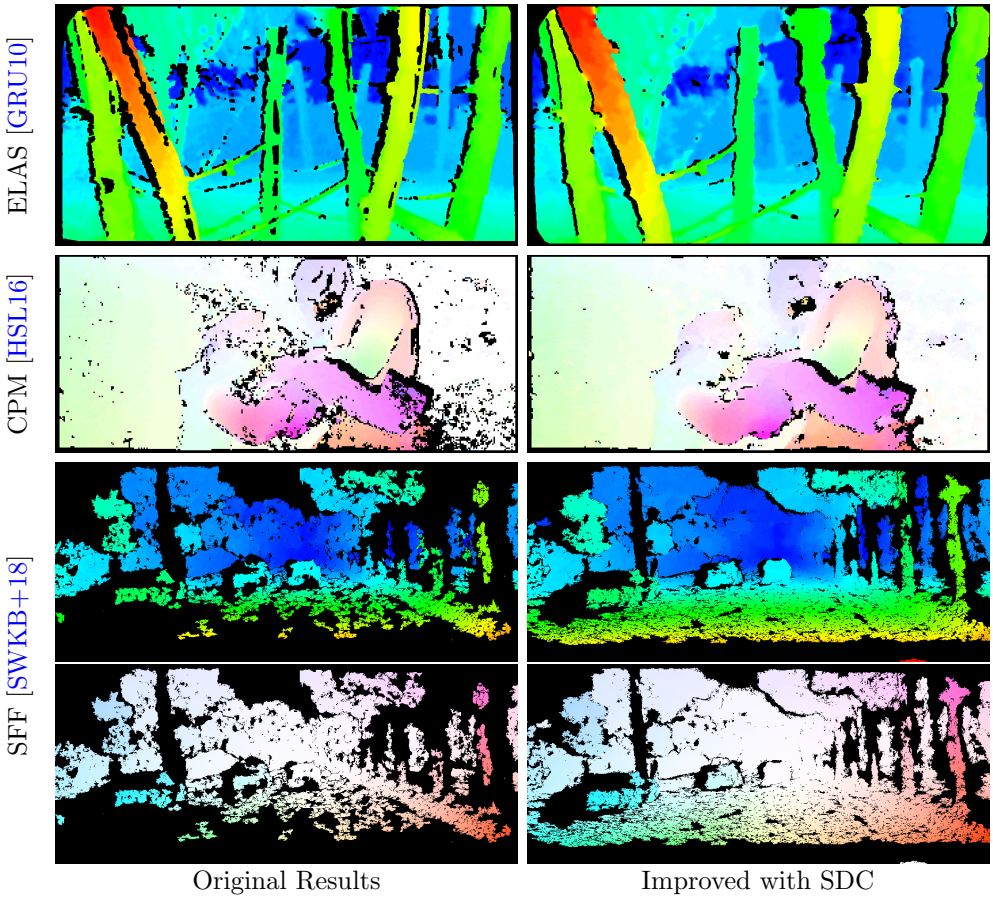



**Figure 5.8.:** *The new SDC feature descriptor improves pixel-wise matching in terms of accuracy and density in state-of-the-art algorithms. From top to bottom: Disparity map for ELAS [GRU10] on ETH3D [SSGS+17], optical flow for CPM [HSL16] on Sintel [BWSB12], and scene flow (disparity and optical flow components) for SFF [SWKB+18] (Section 4.1) on KITTI [MG15].*

match. The results are shown in Figure 5.7b. Naturally, the robustness is higher for larger distances to the correct patch. This experiment validates the effectiveness of the design once again. SDC achieves the highest robustness throughout the whole range of displacements. The top performance of SDC is then followed by a dense cluster of other deep descriptors including the *Tiny* variant. Noteworthy is the strong performance of all networks which are explicitly designed to avoid sub-sampling (no strides greater than 1), especially for small offsets.

### Cross-Task and Cross-Domain Matching

For the third part of the experiments, the SDC feature descriptor is applied in actual matching tasks. In total, five algorithms for three dense match-

ing tasks with overall six data sets are tested. For stereo matching, ELAS [GRU10] and SGM [Hir08] are evaluated on KITTI [MG15], Middlebury [SS02], and ETH3D [SSGS+17]. CPM [HSL16] and FlowFields++ [SBWS18b] are selected to represent optical flow matching algorithms and are evaluated on KITTI [MG15], Middlebury [BSLR+11], HD1K [KNHK+16], and MPI Sintel [BWSB12]. Finally, SceneFlowFields (SFF) [SWKB+18] (Section 4.1) is tested on KITTI [MG15]. Where possible, the non-occluded areas (*noc*) and the full image (*all*) are evaluated separately, because visual matching is only possible in visible regions. On KITTI, these regions are further split into static background (*bg*) and dynamic foreground (*fg*). For the Middlebury stereo data, all levels of resolution are evaluated: Full ($F$), half ($H$), and quarter resolution ($Q$). For Sintel, the more realistic *final* rendering pass is considered only. Baseline results for the common error metrics are computed for all data sets. Then, the feature descriptor of every algorithm is changed to SDC and the experiment is repeated. It is important to note that nothing but the descriptor is changed. For the sake of comparability, no algorithm is fine-tuned, though fine-tuning is expected to improve the results in general.

**Stereo Matching.** ELAS [GRU10] uses first order image gradients for feature description. The default parameter set called *MIDDLEBURY*, which includes interpolation after the consistency check, is used for the experiments. In addition, an open source implementation of SGM[2] is obtained which uses the symmetric CENSUS transform [SLR13] of $9 \times 7$ image patches as a descriptor.

Results for both algorithms on all stereo data sets are given in Table 5.2. Green color indicates where the proposed features outperform the baseline; a decrease in accuracy is marked in red. In case of ELAS [GRU10], the impact of SDC features is advantageous in all cases, and even significant most of the time. SGM [Hir08] shows a couple of negative test cases. These are first of all, the full resolution ($F$) images of Middlebury [SS02] which produce bad results for both descriptors on both data sets, since the default parameters of ELAS [GRU10] and SGM [Hir08] are not adjusted to the maximum possible disparity of that resolution. This might also apply to the half-resolution images ($H$) to some extend. As a consequence, this data should not be considered in the comparison. Then there are the foreground regions of KITTI [MG15], where the deep SDC features perform slightly worse than CENSUS. This might be, because foreground regions are underrepresented in the training data, and thus in the randomly sampled training patches. Lastly, the non-occluded areas of ETH3D [SSGS+17] show minimally higher errors for SDC features. However, the large receptive field of SDC features can compensate for that in occluded regions to improve the overall results. In summary, SDC features improve dense stereo matching for both algorithms on all data sets in most categories.

**Optical Flow Correspondences.** CPM [HSL16] computes sparse matches in non-overlapping $3 \times 3$ blocks that can be used for interpolation with EPICFlow [RWHS15] or RICFlow [HLS17]. The original feature descriptor is SIFT

[2] www.github.com/gishi523/semi-global-matching

**Table 5.2.:** *Evaluation of stereo matching algorithms. ELAS [GRU10] and SGM [Hir08] with the default descriptors are compared to SDC features on KITTI [MG15], Middlebury [SS02], and ETH3D [SSGS+17].*

| Data set | | | ELAS [GRU10] | | | | SGM [Hir08] | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Original | | **SDC** | | Original | | **SDC** | |
| | | | >3px | EPE | >3px | EPE | >3px | EPE | >3px | EPE |
| KITTI | *noc* | *bg* | 6.56 | 1.30 | 4.30 | 1.08 | 4.32 | 1.02 | 3.44 | 0.98 |
| | | *fg* | 12.21 | 1.88 | 8.25 | 1.41 | 6.46 | 1.15 | 7.70 | 1.40 |
| | | *all* | 7.39 | 1.38 | 4.88 | 1.13 | 4.36 | 1.04 | 4.06 | 1.04 |
| | *all* | *bg* | 7.22 | 1.34 | 4.86 | 1.12 | 4.65 | 1.11 | 3.61 | 1.00 |
| | | *fg* | 14.34 | 2.02 | 11.28 | 1.62 | 7.25 | 1.54 | 8.61 | 1.68 |
| | | *all* | 8.29 | 1.45 | 5.83 | 1.19 | 5.03 | 1.18 | 4.34 | 1.10 |
| Middlebury | *noc* | *F* | 26.33 | 20.42 | 22.24 | 20.08 | 43.92 | 44.45 | 45.52 | 52.41 |
| | | *H* | 16.85 | 4.44 | 12.03 | 3.42 | 15.93 | 6.12 | 13.37 | 6.98 |
| | | *Q* | 11.62 | 2.03 | 10.12 | 1.91 | 10.43 | 1.81 | 8.80 | 1.75 |
| | *all* | *F* | 29.87 | 22.47 | 26.22 | 22.16 | 47.28 | 47.56 | 48.09 | 53.50 |
| | | *H* | 21.02 | 6.03 | 16.97 | 5.08 | 19.71 | 7.64 | 16.73 | 8.26 |
| | | *Q* | 15.91 | 2.91 | 15.19 | 2.86 | 14.77 | 2.74 | 12.26 | 2.46 |
| ETH3D | *noc* | | 6.03 | 0.98 | 2.17 | 0.60 | 2.83 | 0.65 | 3.11 | 0.75 |
| | *occ* | | 17.68 | 2.14 | 12.99 | 1.64 | 6.40 | 1.36 | 4.81 | 1.11 |
| | *all* | | 6.50 | 1.02 | 2.61 | 0.64 | 3.62 | 0.81 | 3.49 | 0.83 |

**Table 5.3.:** *Evaluation of optical flow matching with CPM [HSL16]. The SIFT descriptor [Low99] is compared to SDC features on KITTI [MG15], Sintel [BWSB12], Middlebury [BSLR+11], and HD1K [KNHK+16].*

| Data set | | | SIFT [Low99] | | | **SDC** | | |
|---|---|---|---|---|---|---|---|---|
| | | | >3px | EPE | Density | >3px | EPE | Density |
| KITTI | *noc* | *bg* | 10.69 | 2.17 | – | 8.37 | 2.30 | – |
| | | *fg* | 12.67 | 2.40 | – | 9.96 | 2.14 | – |
| | | *all* | 11.26 | 2.21 | 7.88 % | 8.64 | 2.30 | 9.83 % |
| | *all* | *bg* | 11.71 | 3.28 | – | 9.48 | 3.79 | – |
| | | *fg* | 12.67 | 2.40 | – | 9.96 | 2.14 | – |
| | | *all* | 11.87 | 3.13 | 6.79 % | 9.56 | 3.51 | 8.50 % |
| Sintel | *noc* | | 4.33 | 1.06 | – | 4.64 | 1.18 | – |
| | *occ* | | 45.03 | 10.49 | – | 49.52 | 12.56 | – |
| | *all* | | 5.30 | 1.28 | 8.93 % | 5.90 | 1.50 | 9.52 % |
| Middlebury | | | 4.11 | 0.79 | 10.10 % | 2.57 | 0.66 | 10.49 % |
| HD1K | | | 5.85 | 1.29 | 9.80 % | 4.46 | 1.17 | 10.54 % |

**Table 5.4.:** *Optical flow evalation with FlowFields++ [SBWS18b]. The SIFT descriptor [Low99] is compared to SDC features on KITTI [MG15], Sintel [BWSB12], Middlebury [BSLR+11], and HD1K [KNHK+16]. Results for dense matching, after consistency check, and after interpolation are shown.*

| Data set | | | Matching | | | | Filtered | | | | | | Interpolated | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | SIFT [Low99] | | **SDC** | | SIFT [Low99] | | | **SDC** | | | SIFT [Low99] | | **SDC** | |
| | | | >3px | EPE | >3px | EPE | >3px | EPE | Density | >3px | EPE | Density | >3px | EPE | >3px | EPE |
| KITTI | *noc* | *bg* | 23.22 | 12.07 | 15.25 | 6.56 | 8.04 | 1.89 | – | 6.91 | 2.00 | – | 9.56 | 3.08 | 8.52 | 2.97 |
| | | *fg* | 27.61 | 14.47 | 16.90 | 4.45 | 10.31 | 2.11 | – | 9.10 | 2.01 | – | 6.13 | 1.97 | 8.99 | 2.57 |
| | | *all* | 23.98 | 12.48 | 15.53 | 6.19 | 8.39 | 1.92 | 73.3 % | 7.27 | 2.00 | 86.1 % | 8.97 | 2.89 | 8.60 | 2.90 |
| | *all* | *bg* | 35.96 | 53.20 | 29.19 | 39.42 | 9.02 | 3.09 | – | 8.30 | 3.62 | – | 19.13 | 9.45 | 17.19 | 9.13 |
| | | *fg* | 29.17 | 21.81 | 18.64 | 24.15 | 10.32 | 2.11 | – | 9.10 | 2.01 | – | 6.46 | 2.14 | 9.19 | 2.71 |
| | | *all* | 34.93 | 48.45 | 27.59 | 37.11 | 9.22 | 2.94 | 63.3 % | 8.43 | 3.36 | 74.7 % | 17.21 | 8.34 | 15.98 | 8.16 |
| Sintel | *noc* | | 16.21 | 9.31 | 10.15 | 5.17 | 4.15 | 1.00 | – | 3.61 | 0.96 | – | 6.35 | 2.34 | 6.10 | 2.18 |
| | *occ* | | 83.18 | 120.78 | 78.85 | 89.72 | 42.59 | 10.10 | – | 43.81 | 10.71 | – | 45.04 | 22.26 | 46.80 | 21.33 |
| | *all* | | 21.88 | 18.75 | 15.97 | 12.33 | 5.15 | 1.23 | 75.7 % | 4.82 | 1.25 | 84.4 % | 9.62 | 4.03 | 9.55 | 3.80 |
| Middlebury | | | 5.47 | 1.21 | 3.79 | 0.76 | 2.24 | 0.51 | 93.1 % | 2.26 | 0.49 | 96.9 % | 1.69 | 0.28 | 1.79 | 0.30 |
| HD1K | | | 15.52 | 12.99 | 7.76 | 7.48 | 5.64 | 1.16 | 82.6 % | 4.10 | 1.02 | 94.4 % | 4.34 | 0.96 | 4.62 | 1.31 |

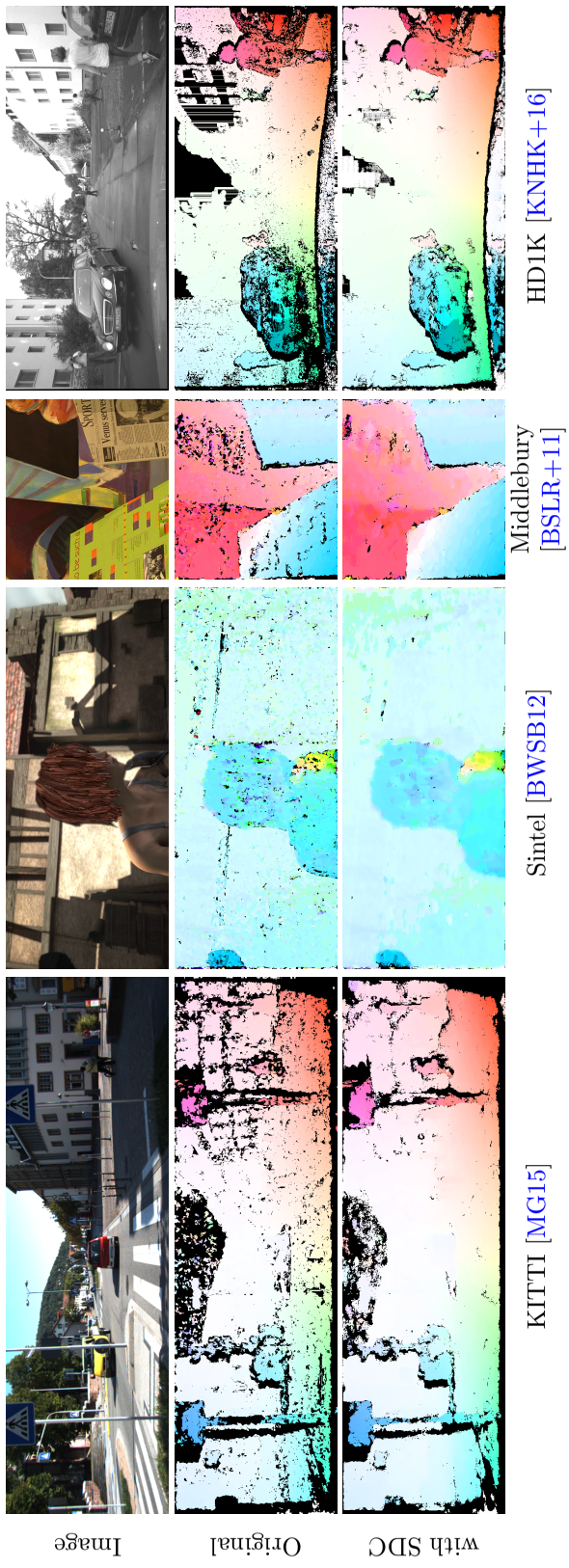



**Figure 5.9.:** *Exemplary visual comparison of filtered optical flow from FF++ [SBWS18b] on four different data sets. The second row shows results for the original method, while the bottom row shows results after changing the feature descriptor to SDC. Note that all parameters are the same for both experiments. Quantitative evaluation on full data sets is provided in Table 5.4.*

[Low99]. The generated matches of this algorithm are evaluated in Table 5.3. FlowFields++ (FF++) [SBWS18b] performs dense matching, followed by a consistency check and interpolation with RICFlow [HLS17]. The comparison between the originally used SIFT features [Low99] and the SDC features after each of these three steps are shown in Table 5.4. For the filtered results after the consistency check, the density is also given as percentage of covered ground truth pixels. Visual examples are presented in Figure 5.9.

In some cases, both algorithms show a slight increase in end-point error for the complete KITTI data (*all*) when used with the SDC features. This is most likely due to the fact, that the KITTI *noc* data excludes the out-of-bounds motions only, not the real occlusions. A higher end-point error in the occluded areas is actually an advantage, because it makes outlier filtering during the consistency check easier. In fact, EPE and outliers are better for KITTI-*all-fg* for FF++ after filtering (see Table 5.4). Also, it is important to note that the filtered matches with SDC are denser for both algorithms (cf. Figure 5.9). Dense, well distributed matches make interpolation easier. This way, the feature descriptor supports the whole pipeline. Again, nothing but the descriptor is changed, not even the distance function that is used to compare the feature descriptors. CPM [HSL16] for example uses the sum of absolute difference as feature distance, while the SDC network was trained using the L2 distance. Overall, SDC features reduce the outliers during optical flow matching by up to 50 %.

**Matching-based Scene Flow Algorithms.** SFF [SWKB+18] is presented in Section 4.1, and can be considered the stereo extension of FlowFields [BTS15] to estimate 3D motion. The pipeline is comparable to FF++ except for the additional egomotion refinement step that is used in SFF. All intermediate results are evaluated and presented in Table 5.5.

Similar to the experiments on stereo and optical flow, the SDC features improve scene flow matching significantly, which results in almost half the percentage of outliers and end-point errors. This effect can be maintained throughout the whole pipeline for almost all image regions. As before, outlier filtering at the foreground regions (*fg*) of KITTI seems to be more difficult with SDC features, which potentially can be solved by adjusting the consistency threshold. The minor decrease in correctness of the filtered matches might again be acceptable when considering that SDC features increase the filtered density from 43.6 % to 67.0 % and from 36.4 % to 56.0 % in non-occluded (*noc*) and *all* image regions (cf. Figure 5.8). The SDC features improve scene flow matching over all image regions (including non-matchable, occluded areas) by more than 10 percentage points which corresponds to 25 % less outliers after matching.

### 5.1.5. Empirical Evaluation Study on Training of SDC

Though a principled way for learnable representations, classifiers, and regressors is endorsed, the current understanding of deep neural networks lags behind.

**Table 5.5.:** *Results for scene flow estimation. SFF [SWKB+18] with SIFTFlow [LYT11] features and SDC features are compared on the KITTI scene flow benchmark [MG15]. The densities after filtering increase from 43.6 % to 67.0 % in noc and from 36.4 % to 56.0 % in all regions when using SDC features.*

| Data | | | Matching | | | | Filtered | | | | Interpolated | | | | Ego-motion Refinement | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | SIFTFlow | | **SDC** | | SIFTFlow | | **SDC** | | SIFTFlow | | **SDC** | | SIFTFlow | | **SDC** | |
| | | | >3px | EPE | >3px | EPE | >3px | EPE | >3px | EPE | >3px | EPE | >3px | EPE | >3px | EPE | >3px | EPE |
| D1 | *noc* | *bg* | 9.84 | 2.00 | 4.69 | 1.10 | 2.29 | 0.85 | 1.39 | 0.68 | 4.94 | 1.04 | 4.26 | 0.93 | – | – | – | – |
| | | *fg* | 16.23 | 2.64 | 9.72 | 1.89 | 2.76 | 0.80 | 2.81 | 0.78 | 7.85 | 1.33 | 7.59 | 1.17 | – | – | – | – |
| | | *all* | 10.91 | 2.11 | 5.53 | 1.23 | 2.37 | 0.84 | 1.60 | 0.69 | 5.43 | 1.09 | 4.82 | 0.97 | – | – | – | – |
| | *all* | *bg* | 11.62 | 4.93 | 6.53 | 5.63 | 2.30 | 0.85 | 1.40 | 0.68 | 5.33 | 1.13 | 4.58 | 1.02 | – | – | – | – |
| | | *fg* | 20.64 | 15.64 | 14.51 | 10.08 | 2.76 | 0.80 | 2.82 | 0.78 | 7.78 | 1.33 | 8.20 | 1.26 | – | – | – | – |
| | | *all* | 12.99 | 6.55 | 7.59 | 6.31 | 2.38 | 0.84 | 1.61 | 0.69 | 5.71 | 1.16 | 5.13 | 1.06 | – | – | – | – |
| D2 | *noc* | *bg* | 17.49 | 2.82 | 10.49 | 1.80 | 2.74 | 0.92 | 1.94 | 0.80 | 12.05 | 2.24 | 7.64 | 1.35 | 6.89 | 1.47 | 6.12 | 1.17 |
| | | *fg* | 16.65 | 2.88 | 11.41 | 1.86 | 2.75 | 0.88 | 2.88 | 0.88 | 9.91 | 1.69 | 8.48 | 1.38 | 10.33 | 1.62 | 8.61 | 1.43 |
| | | *all* | 17.35 | 2.83 | 10.65 | 1.81 | 2.74 | 0.91 | 2.08 | 0.81 | 11.69 | 2.15 | 7.78 | 1.36 | 7.47 | 1.49 | 6.54 | 1.21 |
| | *all* | *bg* | 31.38 | 8.47 | 25.54 | 8.71 | 2.83 | 0.94 | 2.11 | 0.84 | 18.11 | 3.30 | 12.97 | 2.20 | 8.80 | 1.82 | 8.74 | 1.61 |
| | | *fg* | 20.80 | 5.20 | 15.91 | 5.54 | 2.75 | 0.88 | 2.88 | 0.88 | 9.85 | 1.68 | 10.70 | 1.51 | 10.24 | 1.61 | 10.82 | 1.57 |
| | | *all* | 29.61 | 7.97 | 24.09 | 8.23 | 2.82 | 0.93 | 2.23 | 0.85 | 16.86 | 3.06 | 12.63 | 2.09 | 9.02 | 1.79 | 9.05 | 1.60 |
| Fl | *noc* | *bg* | 22.95 | 9.07 | 13.25 | 5.10 | 2.34 | 0.85 | 2.55 | 0.95 | 17.77 | 6.65 | 10.18 | 2.52 | 9.42 | 2.27 | 8.10 | 2.02 |
| | | *fg* | 25.40 | 7.06 | 14.42 | 4.34 | 2.14 | 1.05 | 3.00 | 1.19 | 11.48 | 2.73 | 7.52 | 1.92 | 13.05 | 3.42 | 9.07 | 2.52 |
| | | *all* | 23.36 | 8.74 | 13.44 | 4.97 | 2.31 | 0.88 | 2.61 | 0.98 | 16.72 | 5.99 | 9.73 | 2.42 | 10.03 | 2.46 | 8.26 | 2.11 |
| | *all* | *bg* | 36.68 | 45.26 | 28.36 | 38.11 | 2.43 | 0.97 | 2.71 | 1.11 | 26.84 | 13.42 | 18.31 | 7.44 | 13.04 | 4.71 | 11.73 | 5.00 |
| | | *fg* | 29.70 | 17.64 | 17.60 | 8.20 | 2.14 | 1.05 | 3.00 | 1.19 | 12.52 | 2.83 | 8.88 | 2.12 | 13.97 | 3.47 | 10.31 | 2.68 |
| | | *all* | 35.47 | 41.08 | 26.73 | 33.58 | 2.38 | 0.99 | 2.76 | 1.12 | 24.68 | 11.82 | 16.89 | 6.64 | 13.18 | 4.52 | 11.51 | 4.65 |
| SF | *noc* | *bg* | 29.99 | – | 17.65 | – | 4.51 | – | 3.83 | – | 20.38 | – | 12.52 | – | 11.24 | – | 9.46 | – |
| | | *fg* | 34.97 | – | 22.03 | – | 5.00 | – | 5.39 | – | 16.25 | – | 13.72 | – | 17.48 | – | 15.03 | – |
| | | *all* | 30.82 | – | 17.65 | – | 4.59 | – | 3.83 | – | 19.69 | – | 12.72 | – | 12.28 | – | 10.39 | – |
| | *all* | *bg* | 42.68 | – | 31.19 | – | 4.61 | – | 3.73 | – | 26.84 | – | 20.46 | – | 14.68 | – | 12.97 | – |
| | | *fg* | 40.04 | – | 28.09 | – | 5.00 | – | 5.40 | – | 17.36 | – | 16.42 | – | 18.49 | – | 17.62 | – |
| | | *all* | 42.28 | – | 31.19 | – | 4.67 | – | 3.98 | – | 27.43 | – | 19.85 | – | 15.26 | – | 13.67 | – |

Networks are often handled as black boxes due to the stochastic and iterative nature of back-propagation, the uninterpretable interior of deep and wide architectures, and the increasing number of hyper-parameters.

These facts lead to a conflict for complex, yet safety-critical applications like autonomous driving. On the one hand, most recent achievements for core components of self-driving cars, like perception or action planning, are enabled by deep learning, including e.g. the feature representation for dense matching in this section. On the other hand, the robustness and reliability of these components remain unexplored, which introduces high risk since neither the probability nor the possible maximum harm of wrong decisions is known.

As a result, networks are required that are more interpretable, more robust, and less self-confident (i.e. providing a measure of certainty).

Moreover, part of the success of deep learning is driven by the availability of data. Astonishing results are often obtained only by increasing the amount of training data, using deeper architectures, and thus requiring even more data. Along with this, the computational effort for training increases likewise, introducing another limiting factor. While, in principle, there is nothing wrong with using more data, it has to be kept in mind that data (labeled or unlabeled) is differently scarce for different domains and applications. Thus, a working model for one domain might not be transferable to another. Further, an advanced usage of only very few data is essential to limit the expensive efforts for annotation. As a conclusion, the available data should be used as efficiently as possible to train more accurate and robust models in less time.

In this study, the focus is not on the selection of the architecture, but instead on the effects of training procedures and data in the hope to derive some heuristics that can guide others when training deep neural networks.

**Similar Studies for Learning of Optical Flow.** The importance of training data and schedules for end-to-end optical flow estimation is investigated in [MIFH+18; SYLK19]. In [MIFH+18], it is dealt with the usability of synthetic data for transfer learning (in the form of pre-training + fine-tuning). The authors conduct a series of experiments about lighting, data augmentation, displacement statistics, simulation of realistic noise when generating synthetic images, hyperparameter tuning, and the importance of the order when training with multiple data sets. The model under review is FlowNet [DFIH+15; IMSK+17]. Advanced training strategies for PWC-Net [SYLK18] are presented in [SYLK19]. Here, the focus is to adjust the training process to improve generalization of the network for the Robust Vision Challenge[3].

**Overview of the Matching Tasks.** Depending on the matching problems, the viewpoints between image pairs vary. For optical flow (*of*), images, taken with the same camera, are matched in the temporal domain. For stereo matching (*st*), two distinct rectified cameras capture images simultaneously. A combination of both (*mix*) is possible if a data set provides ground truth for

[3] www.robustvision.net/rvc2018.php

optical flow and stereo disparity. If the annotations further provide a measure for the change of depth, image correspondences between stereo cameras over time (*cross*, (*cr*)) can be established. A data set that contains labels for *st*, *of*, and *cr* is capable of training full scene flow (*sf*) matching. These matching tasks have quite different characteristics.

**Data Sets.** As explained in Chapter 2, data is of utmost importance for training. Increasing effort is spent on capturing, labeling, or the generation of large data sets for different domains to enable a training of deeper and larger models. The generalization to unseen samples – and even more to unseen domains – remains a challenging problem for neural networks. Yet, a tendency to overcome this issue by the extensive use of more and diverse data is evident in recent publications [BVS17; SYLK18].

For many applications it is hard or tedious to collect labeled training data, sometimes even impossible despite applying manual annotation. Therefore, synthetic data sets are often used for the training followed by fine-tuning on the target domain to transfer what has been learned before. Advantages of the generation of synthetic data include the possible, large scale, and dense, exact ground truth annotations. However, image appearance (even if photo-realistic) might not fit the realistic data, thus increasing the problems of generalization, overfitting, and domain adaption.

One large, synthetic data set that is relevant for this work is FlyingThings3D (FT3D) [MIHF+16] since it is, besides KITTI [GLU12; MG15], the only other data set providing full scene flow labels. Especially the *Driving* subset of FT3D is relevant, because it simulates a traffic scenario. MPI Sintel [BWSB12] is also considerably large and provides optical flow and stereo labels, making it a possible option for deep training.

Among realistic data sets, KITTI [MG15] is the natural choice since it provides scene flow ground truth (though sparse) in an automotive context. The data of HD1K [KNHK+16] is also captured from a stereo camera mounted on a driving vehicle, but it provides only annotations for optical flow correspondences. The original SDC network is additionally trained on the other data sets which are part of the Robust Vision Challenge[4] for stereo and optical flow, (Middlebury (MB) [BSLR+11; SS02] and ETH3D [SSGS+17]). However, the latter three are not suitable for training because they are very limited in size. An overview of all these data sets is given in Table 5.6.

**SDC Feature Analysis.** The original training strategy as described in Section 5.1.3 is used for a deeper analysis of the features. First, the learned kernels of the first SDC layer are visualized (see Figure 5.10). The first learned filters with a dilation rate of 1 show a high similarity to two-dimensional second order Gaussian kernels. For higher dilation rates, the kernels become less intuitive. There are also some filters that respond to a certain color.

Next, some of the normalized filter responses of the last SDC layer, i.e. the final feature representation, are presented for an exemplary image in

[4] www.robustvision.net/rvc2018.php

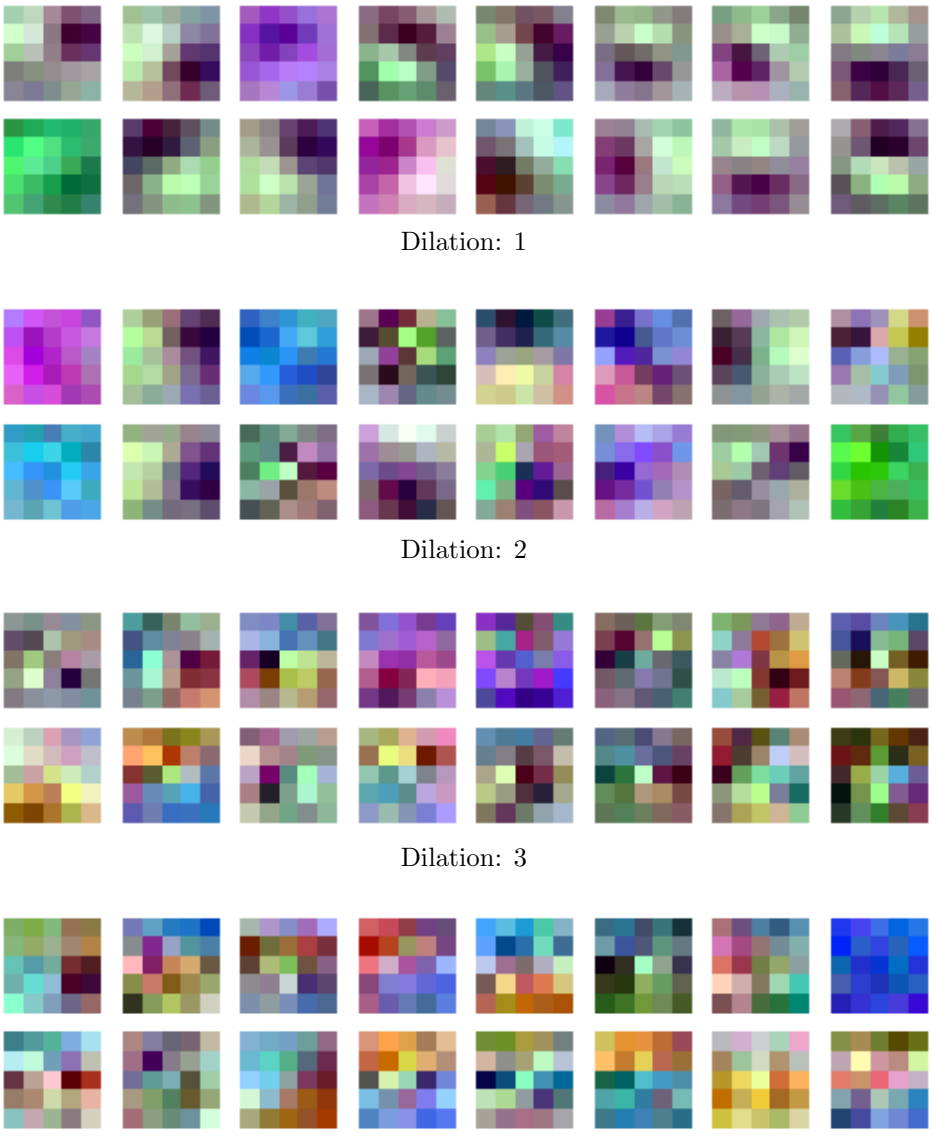
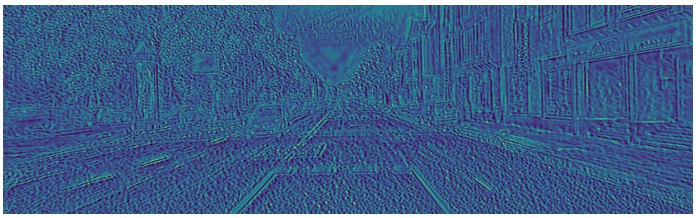
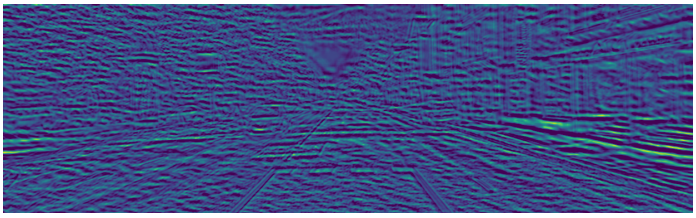



**Figure 5.10.:** *Convolution kernels for the first SDC layer of the SDC feature network [SWUS19]. The color gives the respective sensitivity to the RGB color channels of the input images.*

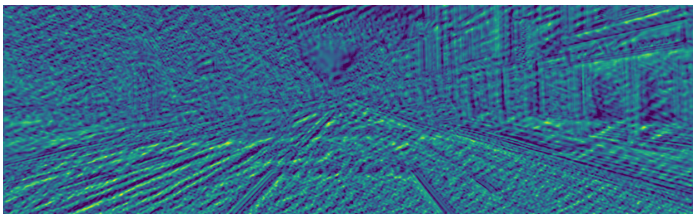
**(a)** *Input Image*



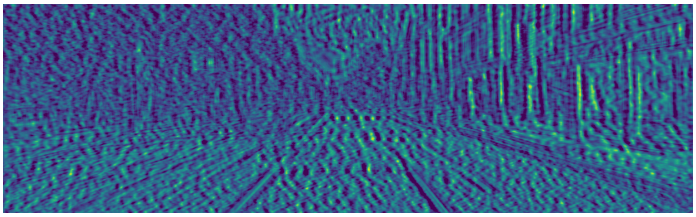
**(b)** *Channel 26*



**(c)** *Channel 53*



**(d)** *Channel 66*



**(e)** *Channel 103*

**Figure 5.11.:** *Some SDC feature channels for the given input image.*

Figure 5.11. Different channels for coarse and fine structures can be identified clearly. One special observation is, that one feature channel dominates the representation (not shown in the figure), i.e. all values are 1, the maximum. Further experiments show that this dimension is the same for all investigated images on all data sets. Also interesting is the fact that more than one third of all dimensions does not contribute to the description significantly, i.e. the features for these channels are all very close to zero for all data sets. The amount of "dead" channels decreases for increasing dilation rates (conv5-1: 18, conv5-2: 16, conv5-3: 12, conv5-4: 7). However, the remaining channels (not 0 and not 1) are all equally important for the description according to their variance.

**Failure Cases.** SDC is evaluated on a test set of patch triplets. A triplet is considered as misclassified, if the feature distance of the corresponding image patches is smaller than the feature distance of non-corresponding patches (cf. Section 5.1.5). Some representative, misclassified triplets from the KITTI test set are depicted in Figure 5.12. The failure cases can be clustered into the following categories where one triplet can belong to multiple classes: *Vegetation* (34 %), *dynamic objects* (29 %), *occlusions* (18 %), *boundary regions* (16 %), *homogeneous patches* (14 %). While homogeneous, untextured, and occluded regions can only be matched with a wider receptive field (i.e. changing the architecture to consider more context knowledge), the issues of dynamic foreground objects and vegetation can be tackled by changing the training schedule as it is done in the next sections. The only reliable way to handle image boundaries is to ignore them during feature computation and matching.

### Improved SDC Training

The experiments within this section are split into two groups. First, it is investigated how training can be improved in general. The second part focuses on data and topics related to training on multiple domains. Unless stated otherwise in the experiments, a model for a single data set is always trained on all available image pairs (e.g. KITTI uses the three image pairs of scene flow). The major evaluation criterion is the triplet accuracy (cf. Section 5.1.4).

**Hard Mining.** Hard mining is a well documented technique to speed up the training and increase the accuracy especially for difficult samples [SKP15]. It is also helpful when training with imbalanced data [DGZ17]. The idea is to ignore samples with a sufficiently accurate prediction during the training and focus more on samples with less accurate or wrong predictions. In this case, offline hard mining is implemented by ignoring triplets with zero loss, i.e. the positive distance is below the threshold and the negative distance is higher than the margin (cf. Equation 5.1). The expected behavior of training with hard mining is threefold. First of all, higher (average) losses are expected since zero losses are neglected. Secondly, training should be speeded up because higher losses lead to higher gradients in more relevant directions. Lastly, the

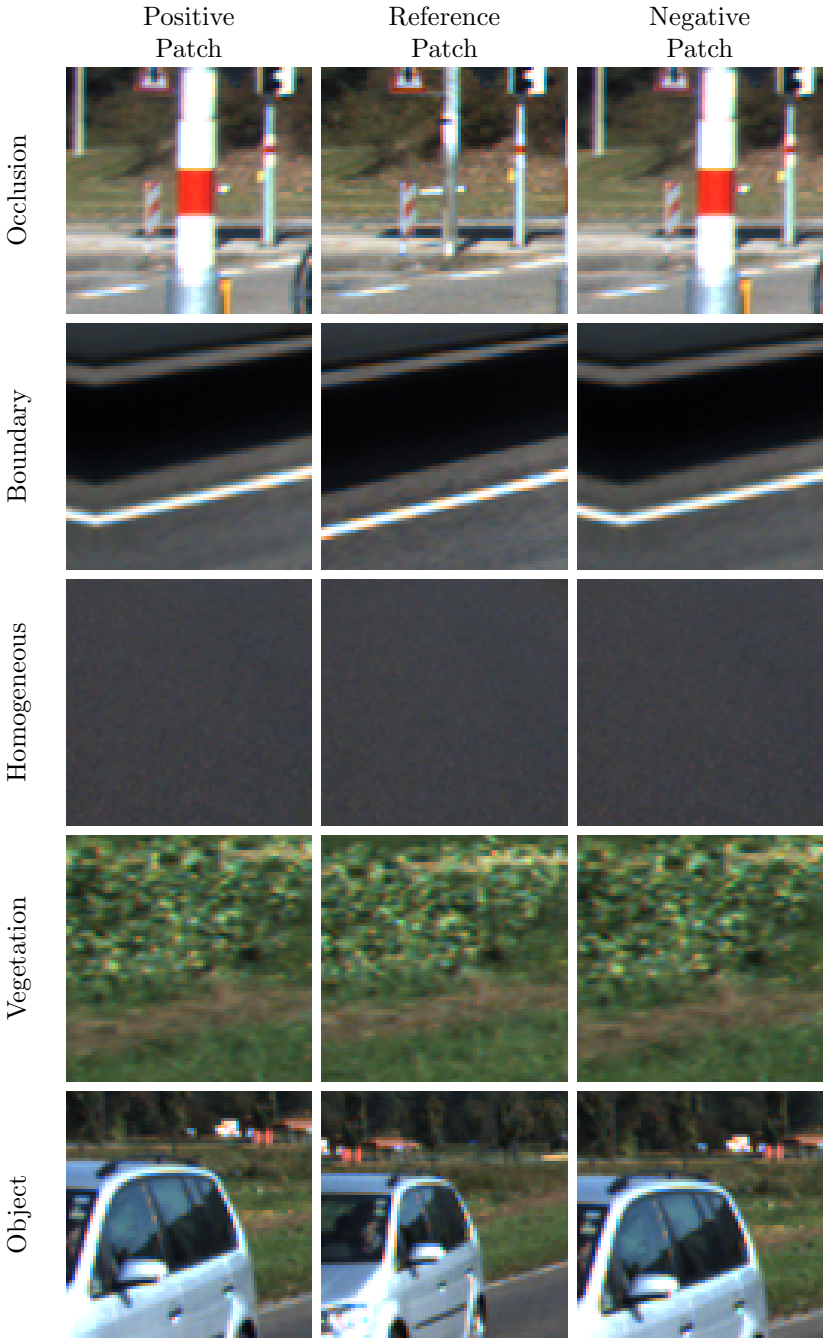
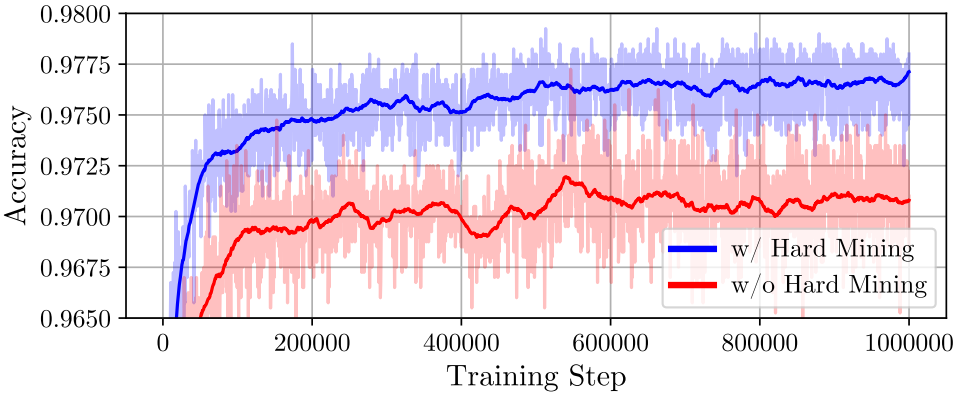
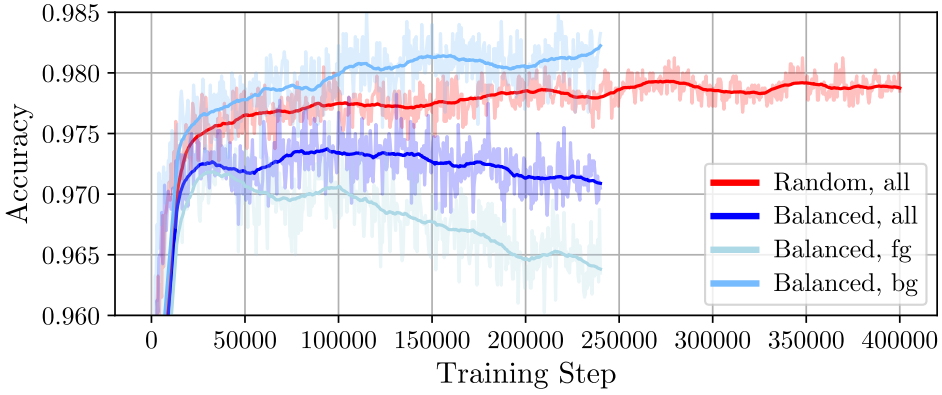


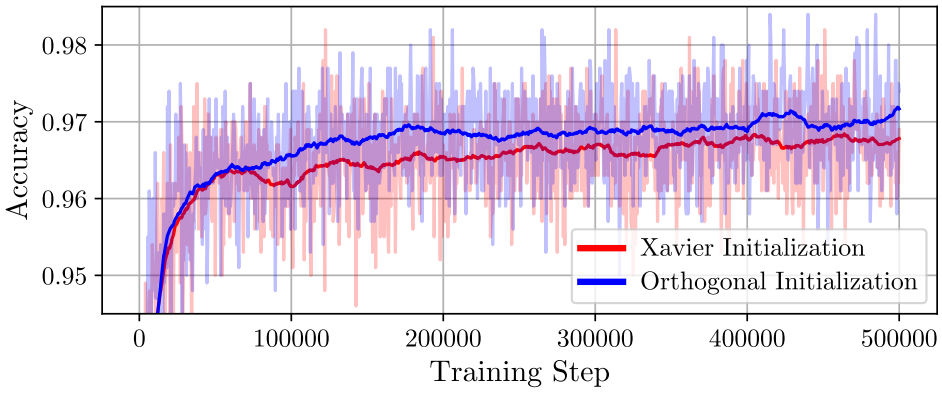

**Figure 5.12.:** *Misclassified triplets from the KITTI test split for the original SDC network.*




**(a)** *Hard Mining.*




**(b)** *Balanced Region Sampling.*




**(c)** *Weight Initialization.*

**Figure 5.13.:** *Comparison of different training setups.*

predictions for difficult samples should be more accurate. Figure 5.13a shows the validation accuracy during the training with and without hard mining. Not only is the training much faster, it also reaches a higher final accuracy.

**Region Sampling.** Foreground objects on KITTI are one of the identified failure categories. Previously it is argued that this is due to the under-representation of dynamic foreground in the KITTI data set (only about 15 % of the available ground truth). Apart from hard mining, this issue is tackled by manually balancing different image regions during patch sampling. Since ground truth object segmentation is available for KITTI training images, reference patches for training can be sampled equally often from foreground objects and static background regions. A comparison between balanced sampling and uniform random sampling is presented in Figure 5.13b by plotting the validation accuracy during training on KITTI optical flow data for different image regions (foreground (*fg*) / background (*bg*) / *all*). It is evident in this diagram that balanced sampling leads to very early over-fitting in foreground regions, thus hindering the convergence of the model. As a result, not even the foreground regions are similarly well described compared to using uniform random sampling.

**Initialization.** The high-dimensional, highly non-linear and non-convex functional together with a stochastic iterative optimization technique makes neural networks sensitive to initialization. Depending on the activation function, the works in [GB10; HZRS15] propose random initialization that considers the scale of the previous layer. Orthogonal initialization [SMG14] is proposed for the use in linear fully-connected layers. The authors also demonstrate positive effects with networks that use non-linear activation and convolutional layers. A state-of-the-art variance scaling initializer [GB10] is compared with orthogonal initialization [SMG14] for the SDC network in Figure 5.13c. Even for the shallow, fully-convolutional SDC Network with ELU activation [CUH16], orthogonal initialization speeds up the training by about a factor of 2. The final accuracy is also slightly higher.

**Learning Rate Disruption.** Initialization is important for stochastic processes and so is the learning rate for the optimizer. Progressively (either in steps or continuous) decreasing learning rates are a best practice to enable convergence to local optima. However, with monotonically decreasing learning rates, the optimizer can not escape local optima. A measure to encounter this is learning rate disruption, as used e.g. in [SYLK19]. The idea is to disrupt the learning rate schedule by increasing the learning rate significantly (e.g. to the initial value) and then continue with the progressive learning rate decay. This way, the optimizer can escape from a local optimum (though not necessarily in favor of a better optimum). This concept is tested for the training of the SDC network. Figure 5.14 shows three alternate learning rate schedules: The original monotonic decay as used before and two variants of learning rate disruption with different periods for recovery. Some signs of over-fitting could

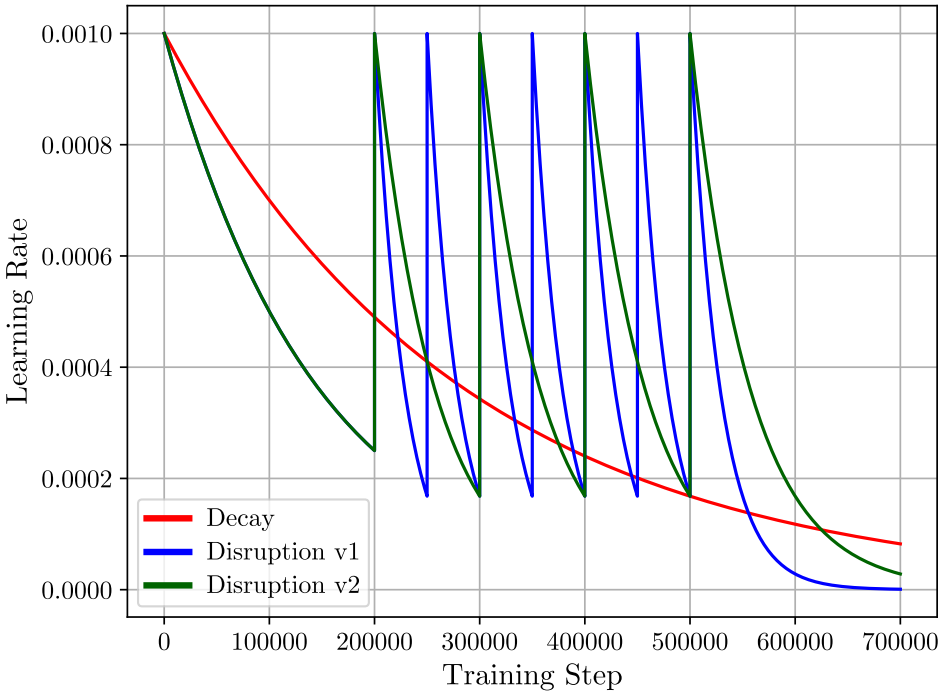



**Figure 5.14.:** *Monotonic decreasing learning rate schedule and two versions for learning rate disruption.*

be observed right after the disruption. However, the network did recover quickly but without any significant sign of changing the local optima (neither in a positive nor negative way).

### Multi Domain Training

**Domain Similarity.** As an approximation of the similarity of domains, mono-domain networks are trained on a single data set and cross-evaluated on all data sets. Table 5.7 shows the evaluation matrix for all trained models on all data sets. No model is trained on the Middlebury (MB) data sets [BSLR+11; SS02] or ETH3D [SSGS+17] because of their small size. For all other data sets, it is trained with the union of all available image correspondences (the three scene flow image pairs for KITTI, FT3D, and Driving, and optical flow and stereo correspondences for Sintel). Training all combinations of data sets and tasks would be infeasible.

It can be observed that domain differences are not necessarily symmetric. Moreover, the matching task (i.e. the type of image correspondences) has influence on the matching performance. Matching on the Driving data set is particularly difficult. On HD1K [KNHK+16], matching is extremely simple, probably because the ground truth does not contain any dynamic objects. The performance of all models is similarly high for the Middlebury Flow data [BSLR+11]. This is most likely due to the very small displacements. The model trained on Sintel [BWSB12] shows high compatibility with many diverse data sets.

The overall observation is that domain similarity for matching is mostly defined by the displacement characteristics and camera hardware, and less by the scenario or realism of the data. The Driving data set, for example, shows a big discrepancy to KITTI in the cross-evaluation, though both contain traffic scenarios. Reversely, Sintel shares neither the realism nor the automotive setting with KITTI, but still demonstrates high compatibility. This observation is in accordance with the results in [MIFH+18] on displacement statistics for optical flow. This can be further confirmed by an additional experiment: The Driving data set is generated for two virtual cameras with different focal lengths (15 and 35 mm). The two versions do not differ in anything else. Performing a cross-evaluation with models trained on KITTI and both versions of Driving, there is a significant loss in domain similarity when switching to the 35 mm focal length, which is also further away from the KITTI camera parameters. Moreover, the transfer between Driving with different focal length does also not work very well.

**Color.** Two questions of interest regarding color spaces are 1.) Which color space provides good generalization properties? and 2.) Does color influence domain adaption? The first question is investigated by training models on one data set with two different color spaces (RGB and YUV) and evaluating them on the other data sets (each in the respective color space). In the experiments, there is no clear sign that one of the two color spaces should be preferred over

**Table 5.6.:** *Characteristics of different data sets.*

| Data Set | Task | Number of Sequences | Frames per Sequence | Image Size [MP] | Color Space | Synthetic Real | Automotive Context |
|---|---|---|---|---|---|---|---|
| KITTI [MG15] | sf | 200 | 1 | 0.46 | RGB | R | yes |
| FlyingThings3D [MIHF+16] | sf | 2239 | 10 | 0.52 | RGB | S | no |
| Driving [MIHF+16] | sf | 1 | 800 | 0.52 | RGB | S | yes |
| Sintel [BWSB12] | mix | 23 | 46 | 0.45 | RGB | S | no |
| HD1K [KNHK+16] | of | 35 | 30 | 2.8 | Gray | R | yes |
| Middlebury Flow [BSLR+11] | of | 8 | 1 | 0.25 | RGB | both | no |
| Middlebury Stereo [SS02] | st | 15 | 1 | 1.1 - 17.4 | RGB | R | no |
| ETH3D [SSGS+17] | st | 16 | 1 | 0.31 / 0.46 | Gray | R | no |

**Table 5.7.:** *Cross evaluation for different domains represented by different data sets. For each evaluation set, the best model trained with a different data set is given in bold.*

| Train \ Eval | KITTI | | | | | FT3D | | | | | Driving | | | | | Sintel | | | HD1K | MB | | ETH3D |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | sf | mix | cr | fl | st | sf | mix | cr | fl | st | sf | mix | cr | fl | st | mix | fl | st | fl | fl | st | st |
| KITTI [MG15] | 97.2 | 97.7 | 97.8 | 97.9 | 96.2 | 91.4 | 91.5 | 90.1 | 93.5 | 90.1 | **68.6** | **70.7** | **64.5** | 66.7 | **75.7** | 90.0 | 89.3 | 90.7 | **98.5** | 98.8 | 90.3 | 95.5 |
| FT3D [MIHF+16] | 73.9 | 76.5 | 74.6 | 76.9 | 73.3 | 95.1 | 95.5 | 93.7 | 96.7 | 94.4 | 57.4 | 60.8 | 51.7 | 51.8 | 71.5 | **95.3** | **92.7** | **95.3** | 97.5 | 96.8 | 82.2 | **96.7** |
| Driving [MIHF+16] | 89.3 | 90.8 | 86.7 | 89.6 | 90.6 | 89.9 | 90.0 | 89.1 | 92.0 | 88.5 | 75.2 | 75.8 | 74.2 | 74.6 | 76.7 | 88.7 | 89.0 | 88.7 | 97.0 | 99.2 | 85.9 | 92.7 |
| Sintel [BWSB12] | **93.5** | **94.6** | **92.3** | **95.2** | **92.9** | **92.7** | **92.8** | **91.6** | **94.2** | **91.8** | 59.8 | 62.7 | 55.2 | 55.6 | 70.8 | 92.3 | 92.7 | 93.0 | 97.2 | **99.5** | **91.4** | 94.2 |
| HD1K [KNHK+16] | 91.0 | 92.0 | 90.6 | 92.8 | 90.8 | 91.6 | 91.7 | 90.7 | 93.7 | 90.3 | 66.2 | 68.2 | 64.0 | **67.0** | 69.7 | 88.7 | 88.0 | 88.0 | 99.5 | 99.0 | 89.3 | 95.9 |

the other in terms of generalization. Both models perform similarly on all test data sets. There is also no sign that YUV or RGB color promote the training process. To answer the second question, two models are trained on KITTI, one with the original RGB color and one with gray scale converted images to match the color space of HD1K. Intuitively, is is assumed that the gray scale model performs better when evaluated on a gray scale data set like HD1K. Contrary, the result of the experiment shows that the color model achieves a higher accuracy on HD1K data compared to the gray scale model. However, when swapping training and evaluation data, the model trained on HD1K performs better on KITTI if the test images are converted to gray scale.

**Scale.** In a similar fashion, the influence of scale spaces is studied. HD1K images have a much higher resolution compared to KITTI (cf. Table 5.6), thus the receptive field of the SDC network ($81 \times 81$ pixels) covers a much smaller part of the visible scene; even more so because the field of view (FOV) of the camera device is smaller (69 ° instead of 90 °). Again, the assumption is that shifting the scale for the training domain towards the scale of the target domain, improves the transfer. Once more, in contrast to the expectation, a model trained on down-scaled HD1K data does not perform better on KITTI compared to a model trained on full resolution images. Here, the inverse experiment (KITTI model evaluated on full resolution and down scaled HD1K data) indicates also that images should not be scaled to achieve better domain transfer. This might be due to artifacts introduced by the scaling.

Nonetheless, scale is important for detection and matching. The SDC network is specifically designed to deal with varying scales by the use of parallel convolutions with different dilation rates [SWUS19]. Table 5.8 shows some baseline descriptors, the original SDC network, and SDC trained on image patches extracted at multiple scales (*multi-scale*), all evaluated on multiple scales of the KITTI data. The heuristic descriptors (SIFT [Low99], DAISY [TLF10], BRIEF [CLSF10]) show an almost quadratic loss in performance when the image size decreases, even if they are supposed to be scale-invariant. For increased image scale, they perform better. It is presumed that this is because smaller patches show fewer deformations, or other variations between viewpoints. The implicit multi-scale design of SDC performs extremely well on different scales, with only a small drop in accuracy. For SDC, the performance drops also when the input is up-sampled. This is not surprising since the dilation rates can only simulate smaller scales. A model explicitly trained on multi-scale data amplifies the scale invariance even more, showing almost no degradation of the accuracy when the scale changes.

**Normalization.** Standardization of the input is useful to remove any bias from the data and to scale features into equal range, making them equally important for training. A common practice is to remove the mean pixel value and to scale them so that all channels have unit variance. Surprisingly, standardization is not crucial to train the SDC network. A model trained on normalized images performs as well as a model trained on the original image data.

**Table 5.8.:** *Multi-scale behavior for different descriptors.*

| Descriptor | ×2 | Original | ×0.5 | ×0.25 |
|---|---|---|---|---|
| Multi-scale | **96.60** | **97.30** | **96.85** | **96.60** |
| **SDC** [SWUS19] | 94.55 | 97.25 | 96.70 | 93.90 |
| BRIEF [CLSF10] | 95.00 | 94.00 | 90.50 | 82.15 |
| DAISY [TLF10] | 92.80 | 91.25 | 88.15 | 80.80 |
| SIFT [Low99] | 93.90 | 89.90 | 81.95 | 73.65 |

Anyway, normalization might also be useful to boost transfer learning by adjusting the pixel distribution to better fit the target domain. This, of course, is only possible if imagery for the target is available at the time of training. When training on a single domain, experiments showed that neither normalization nor a distribution shift help to better generalize to unseen domains. Yet, when training with a mixture of data (as done in the original SDC network), standardization for each training data set separately improves the performance on unseen domains if the test data is also standardized according to its own statistics. For training on multiple domains, a unified normalization based on the pixel distribution of the entire data works also very well and is favorable if a single, unified model for different domains is required.

### 5.1.6. Conclusion

Based on the observation that dilated convolution is related to sub-scale filtering, a novel neural network layer is designed by stacking multiple parallel dilated convolutions (SDC). These SDC layers are combined to a new architecture that can be used for image feature description. For all experiments, a single unified network for all data sets and algorithms is used. The SDC features outperform heuristic image descriptors like SIFT and other descriptor networks from previous works in terms of accuracy and robustness. In a second set of experiments, the SDC feature network is applied in different matching tasks on many diverse data sets and it is shown that the deep descriptor improves matching for stereo, optical flow, and scene flow drastically yielding a better final result in the majority of cases.

The analysis of the network and its feature representation brings insights into the weaknesses of SDC features, which motivate some adjustments of the training schedule. Proper weight initialization and hard mining during the computation of the loss improve the accuracy and speed up training by a factor of about 4. More balanced region sampling during the generation of training data or learning rate disruption do not improve the network's performance. The evaluation of similarity of different domains gives useful directions to improve the process of domain adaption and the training on multiple data sets. Also, the influence of color, scale, and normalization is investigated. The excellent scale invariance of SDC is boosted even more by dedicated multi-scale training.

For future work, it will be interesting to improve the feature description to make use of all feature dimensions and to explicitly model a measure of uncertainty or matching likelihood of image points.

## 5.2. Deep Scene Flow Interpolation

The problems of interpolation and extrapolation have a long history in mathematics and computer science. In this thesis, interpolation finds its application in 3D motion estimation [SWKB+18; SWUK+20].

As described in Chapter 4, in high-level computer vision there are also other problems that benefit from a formulation as a sparse-to-dense problem, e.g. optical flow estimation [BTS15; BTS19; GS03; HLS17; HSL16; LWAS+12; NM03; ORBH18; RWHS15; WZLY+19; ZW17]. Other tasks embody this characteristic naturally, e.g. depth completion where it is the goal to densify sparse LiDAR measurements with respect to a reference image [CWY18; JDWP+18; MCK19; TTFL+20; USSF+17].

The strategies of previous work are quite distinct for motion field interpolation and depth completion. While the first focuses on hand-crafted models and piece-wise patches extracted from edge information (cf. Chapter 4), the latter fully relies on deep neural networks often neglecting the information of the corresponding image, or not using its full potential. With the learning capabilities and inherent parallelism of the data-driven approach, this section is supposed to push the limits of motion field estimation towards higher accuracy and speed. At the same time, previous ideas from depth completion shall be extended and combined into a model that works equally well on different domains and applications. This exposes novel challenges like effective mechanisms for handling sparse data with different patterns or densities, efficient strategies for guidance from dense image information, or suitable fusion of heterogeneous data (e.g. image and depth feature representations).

To solve the aforementioned challenges, Sparse Spatial Guided Propagation (SSGP) [SWUS21] is proposed, which is the combination of efficient, spatially invariant, image-dependent *convolutional propagation* and *sparsity-aware convolution*. This key concept is used in a *generic* sparse-to-dense encoder-decoder with *full image guidance* at every stage. The overall contribution consists of the following:

- A unified architecture which performs sparse-to-dense interpolation in different domains, e.g. interpolation of optical flow, scene flow, or depth.
- A proper architectural design that leads to excellent robustness against noisy input or changes in the input density.
- Appropriate image guidance to resolve the dependency of previous flow interpolators on edge maps.
- A modification of existing spatial propagation that saves a vast amount of trainable parameters and improves generalization.
- Exhaustive experiments to validate all the above claims and to compare to state-of-the-art where in several cases SSGP produces top results.

### 5.2.1. Sparse-to-Dense Interpolation in the Literature

**Sparse-to-Dense Motion Estimation.** The interpolation of sparse points to a dense motion field dates back to at least [GS03; NM03]. A practical approach for large displacement optical flow is introduced by EPICFlow [RWHS15]. The authors make use of image edges computed with SED [ZL15] to find local edge-aware neighborhoods of previously computed, sparse flow values. Based on these neighborhoods, an affine 2D transformation is estimated to interpolate the gaps. Later, this concept is improved by RICFlow [HLS17] to be more robust by using small superpixels and RANSAC in the estimation of the transformation. In Chapter 4, both interpolators for optical flow are transferred to the higher dimensional scene flow setup. These are the interpolation modules of SFF [SWKB+18] and SceneFlowFields++ (SFF++) [SWUK+20], which are refered to as *EPIC3D* and *RIC3D* respectively. SemFlow [WZLY+19] extends the above concepts for interpolation of optical flow by the use of deeply regressed semantic segmentation maps. These maps replace the edge information used in EPIC or RIC to improve the measure of similarity within connected neighborhoods of input matches. However, this approach is heavily dependent on semantic segmentation algorithms and thus not suitable for all domains and data sets. Lastly, InterpoNet [ZW17] is another recent approach that considers deep neural networks for the actual interpolation task. Yet, InterpoNet still requires an explicit edge map as input.

In contrast to all interpolation modules mentioned, SSGP performs dense interpolation at full resolution for a multitude of problems (i.e. it is not restricted to optical flow or scene flow) and utilizes a trainable deep model (i.e. it is not subjected to hand-crafted rules or assumptions and provides significantly better run times). Additionally, the existing approaches highly depend on an intermediate representation of the image (edges, semantics). SSGP operates on the input image directly and resolves this dependency.

**Depth Completion.** Most recent related work (especially in the area of deep learning) is concerned with depth completion. In this field, literature differentiates between unguided and guided depth completion. The latter utilizes the reference image for guidance. In the setup of guided depth completion, novel questions arise which are also highly relevant for this work, e.g. how to deal with sparse information in neural networks or how to combine heterogeneous feature domains. SparseConvNet [USSF+17] introduces sparsity-invariant CNNs by normalizing regular convolutions according to a sparsity mask. This work also introduces the Depth Completion Benchmark to the KITTI Vision Benchmark Suite [GLU12]. Later, confidence convolution [EFK19] is presented, another strategy for the handling of sparsity. In this case, the authors replace the binary sparsity mask with a continuous confidence volume that is used to normalize features after convolution.

Another promising strategy is the use of spatially variant and content dependent kernels in convolutional networks [LHAY16; WZZH18]. This idea is successfully used by [LDGZ+17] for semantic segmentation and later by CSPN [CWY18] for the refinement of already densified depth maps. Most recently,

GuideNet [TTFL+20] has applied the same idea for the densification of sparse depth maps itself. In all cases, the idea is to predict per-pixel propagation kernels based on the image (or a feature map) directly instead of learning a spatially invariant set of kernels that is likewise applied to every pixel of the input.

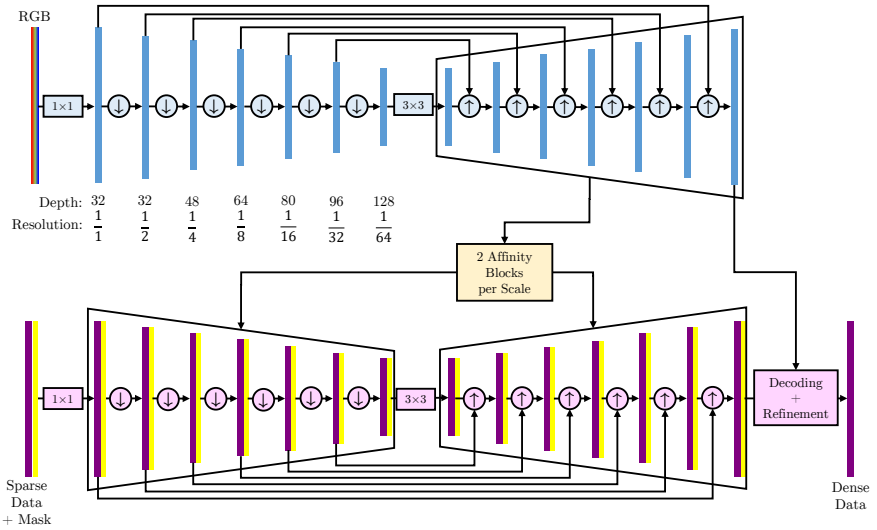
The proposed approach makes use of the two latterly presented concepts, namely awareness and explicit handling of sparsity as well as learning of spatially variant and image-dependent convolutions. Both ideas are combined in the novel, sparsity-aware, image-guided interpolation network that uses a new Sparse Spatial Guided Propagation module.

**Other Interpolation Tasks.** Lastly, there are more computer vision problems that are remotely related, e.g. image inpainting which is also a problem of interpolation. However, for image inpainting the challenge usually lies within the reconstruction of the texture. For the interpolation of geometry or motion, the expected result is piece-wise smooth and thus the problem is rather to find semantically coherent regions. Still, related ideas can also be found in the field of image inpainting, where e.g. in [LRSW+18] partial convolutions are used, which is the same idea for the handling of sparsity as in [USSF+17]. Similarly, the task of super-resolution can also be posed as an interpolation problem with a regular pattern of sparse input. Though theoretically, SSGP is directly applicable to this family of problems, super-resolution goes beyond the scope of this thesis and might be easier to be solved with other approaches.

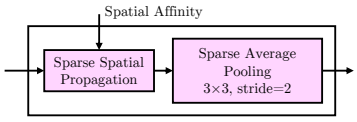
### 5.2.2. Interpolation Network

As motivated earlier, a deep neural network is used for the task of sparse-to-dense interpolation. The network has to be equipped with an appropriate mechanism to handle sparsity, otherwise the considerably large gaps in the used sparse-to-dense motion estimation pipelines can lead to significantly deteriorated feature representation in these regions. For the same reason of large gaps in motion fields (contrary to e.g. depth completion where LiDAR measurements follow a predictable pattern of rotated scan lines), the network architecture has U-Net [RFB15] structure. This way, even large gaps are effectively closed after a few layers of the encoder, leading to a dense representation at the bottleneck. Additionally, to inject a maximal amount of guidance through the entire sparse-to-dense codec, image information is used to compute spatially variant propagation kernels that are applied for densification by convolutional propagation in the sparse encoder, and for guided up-sampling in the dense decoder. These guidance kernels are computed from the RGB image within a feature pyramid network with skip connections, for high expressiveness and accurate localization.

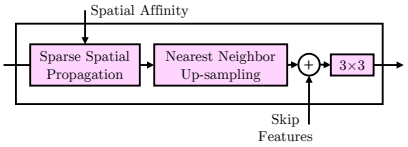
In summary, the interpolation network consists of four components. Firstly, the RGB codec for computation of image-dependent and spatially variant propagation kernels. Secondly, a sparse, spatial propagation module that is likewise used within the encoder and decoder of the sparse-to-dense codec. Thirdly,



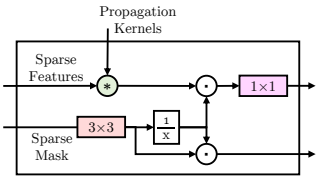

**(a)** *Overall network architecture showing the RGB and the sparse-to-dense codec.*



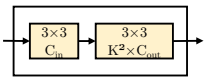

**(b)** *Sparse down-sampling block.*



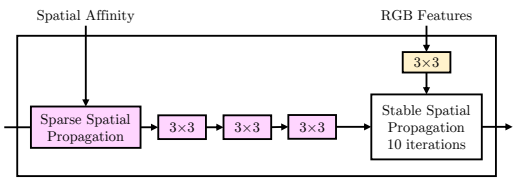

**(c)** *Sparse up-sampling block.*




**(d)** *The proposed sparse spatial propagation module.*




**(e)** *Affinity module that predicts spatially variant kernels.*




**(f)** *Decoding and refinement.*

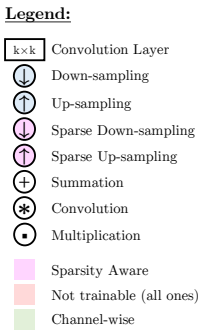



**Figure 5.15.:** *An overview of the network architecture (a) as well as a close-up view on various modules.*

the u-shaped sparse-to-dense network that applies the propagation module for guidance and considers sparsity throughout. Lastly, a dense refinement module to further improve the dense result. The combination of all elements – the sparse-to-dense interpolation network – is visualized in Figure 5.15.

### RGB Codec

The purpose of the RGB codec is to provide a well-shaped feature representation of the image that fits the according level of the sparse codec. Therefore it mimics the shape of the sparse codec and has the same number of levels $l$ in the encoder and decoder as the interpolator. The image are pre-processed by a regular $1 \times 1$ convolution and is then passed through $l$ down-sampling blocks. Each consists of four $3 \times 3$ convolutions where the third convolution applies a stride of 2 to sub-sample the representation. After one additional convolution at the bottleneck, the representation of the lowest resolution is passed through $l$ up-sampling blocks. Again, each of these blocks consists of four $3 \times 3$ convolutions, but this time the second one is a transposed convolution with a stride of 2 for up-sampling. After up-sampling, the intermediate feature representation gets concatenated with the next higher resolved level of the encoder, i.e. regular skip connections to reintroduce localization into the feature maps. In this architecture, the number of output channels is gradually increased as the spatial resolution is reduced, which is a common practice for low resolution feature embeddings. The setup uses $l = 6$ pyramid levels with fully symmetric feature depth of 32, 32, 48, 64, 80, 96, and 128. An overview of the RGB codec is shown in Figure 5.15a.

Finally, two affinity blocks are branched from each level of the decoder to predict the spatially variant, content-dependent kernels for each scale. One affinity block consists of two convolutional layers. One layer is used for pre-transformation, and one to predict a single $K \times K$ kernel per pixel for propagation in the sparse-to-dense codec. Please note that different sets of propagation kernels are predicted for the encoder and the decoder of the sparse codec, i.e. weights are not shared for the two affinity blocks at each level of the RGB decoder. For reasons of memory consumption and computational efficiency, the propagation kernels have a size of $K = 3$. Contrary to existing work [CWY18], SSGP uses a single, flat affinity map independent of the number of feature channels to propagate. This reduces the total number of parameters significantly and effectively diminishes over-fitting during fine-tuning on small data sets.

### Sparse Spatial Propagation

The previously computed multi-scale feature maps, affinity maps, and propagation kernels are now used within the sparse spatial propagation module. Consider an arbitrarily shaped $H \times W \times C$ feature representation $\mathcal{S}$ of the sparse input along with a binary sparsity mask $\mathcal{M}$ of shape $H \times W \times 1$ and a feature representation $\mathcal{F}$ of the guidance image of the same spatial size (and potentially a different number of feature channels). The affinity block of the

previous section transforms the image features $\mathcal{F}$ into a set of propagation kernels $\mathcal{K}$ of the shape $H \times W \times 1 \times K^2$. For the sake of affinity and propagation, the center pixel of the propagation kernels is fixed to 1, i.e. isolated sparse points are not altered. These kernels are then applied in a channel-wise $K \times K$ convolution with the sparse representation $\mathcal{S}$ to spread the information into the neighborhood according to the image features. In GuideNet [TTFL+20] one set of kernels is predicted for each feature channel of the sparse input, which leads to the necessity of depth-wise separable convolutions [Cho17]. Other than that, here a single affinity map is predicted, which results in the natural use of depth-wise convolution for practicability and efficiency. After channel-wise spatial propagation, a $1 \times 1$ convolution is performed to mix the propagated input dimension and expand (or compress) the representation to a new feature depth. Further and in contrast to existing methods using convolutional spatial propagation, sparsity-awareness is explicitly modeled in the propagation module. Towards this end, the idea of sparse convolution from [USSF+17] is adopted and the sparsity mask $\mathcal{M}$ is utilized to normalize the propagated features. By that, only valid information is spread according to the guidance image to fill in the gaps. Formally, the output of the sparse spatial convolution of $\mathcal{S}$ with $\mathcal{K}$ for a single pixel and a single channel $c$ is

$$\tilde{\mathcal{S}}_c = \frac{\sum_{i,j \in \mathcal{W}} \mathcal{S}_{c,i,j} \cdot \mathcal{K}_{i,j}}{\sum_{i,j \in \mathcal{W}} \mathcal{M}_{i,j}}, \tag{5.2}$$

whereas $\mathcal{W}$ is the $k \times k$ window around the pixel under consideration. The normalization and the propagation kernel are independent of the feature channel, i.e. there is only a single 1-channel mask $\mathcal{M}$ and a single set of kernels $\mathcal{K}$ for the entire feature volume. This relationship is also visualized in Figure 5.15d. The entire concept expands directly to arbitrary batch sizes.

#### Image-guided Sparse-to-Dense Codec

The RGB codec and the sparse spatial propagation module enable an efficient way to introduce image guidance to the interpolation network. All convolutions of the sparse-to-dense codec make use of the sparse convolution as presented by [USSF+17]. Sparsity masks are used throughout the entire sparse codec which makes it easy to verify that full density is reached by the end of the decoder by the latest (usually already at the bottleneck), i.e. all pixels are filled with information from the initially valid points. As with the RGB codec, the sparse input is pre-processed with a sparse $1 \times 1$ convolution. Then, $l$ sparse down-sampling blocks are applied. These blocks consist of the sparse spatial propagation module that applies the spatial guidance kernels from the RGB decoder, followed by a $1 \times 1$ convolution to complete the depth-wise separation of the spatially variant guidance. The last step within this block is a sparse average pooling layer with a kernel size of $3 \times 3$ and a stride of 2 to perform the sparse sub-sampling. Again, a single $3 \times 3$ convolution is applied at the bottleneck. Starting at lowest resolution from the bottleneck, $l$ guided up-sampling blocks are passed through. As with the down-sampling, the first

part of these blocks is the depth-wise separated sparse spatial propagation. Then, the feature representation along with its validity mask are up-sampled using nearest-neighbor interpolation to avoid mixture with invalid pixels in case some are still remaining. Lastly, skip connections are established from the next higher resolution of the sparse encoder. The skipped encoder features are summed up with the decoder features to avoid reintroduction of sparsity into the feature representation and merged in another $3 \times 3$ convolution.

At full input resolution of the decoder pyramid, one additional sparse spatial guided propagation is performed, followed by three more convolutions for final decoding. The first two of these are of size $3 \times 3$, the other's size is $1 \times 1$. The last two have linear activation to allow a final prediction of negative motions. Theoretically, the two linear activated convolutions could be folded into a single one. However, the explicit separation leads to a faster convergence initially, probably due to better initialization by separation. Another advantage of using sparse convolution is that (especially during the decoding) no negative boundary effects are introduced, because the sparsity mechanism can treat padded areas as invalid.

#### Dense Refinement

At the end of the sparse-to-dense codec, a dense result in the respective target domain is already obtained. However, following the idea of CSPN [CWY18], the result is further refined using spatial propagation for filtering. Since the RGB codec already provides a strong feature representation, these features can be transformed into affinity maps for each output channel using a single $3 \times 3$ convolution. The kernels extracted from the affinity maps are further transformed to introduce stability as in CSPN [CWY18]. The dense results are then refined during 10 iterations of spatial propagation.

### 5.2.3. Data, Training, and Implementation Details

**Data Sets.** The KITTI data set [MG15] only provides 200 annotated images for scene flow and optical flow. To overcome this issue, synthetically generated data is used, i.e. the FT3D data set [MIHF+16]. This data is shown to be irreplaceable for pre-training [IMSK+17; MIHF+16; SSWS19; SYLK18]. Next to KITTI and FT3D, Sintel [BWSB12] provides a trade-off between realism and size, though only for optical flow. Additionally, HD1K [KNHK+16] is used for extended experiments with interpolation of optical flow. For depth completion, the KITTI Benchmark Suite [GLU12; USSF+17] offers a larger and yet more realistic data set that provides labels for about 45000 stereo image pairs.

For all results in Section 5.2.4, the experiments are performed on a randomly selected validation split which is not used for training. In particular, these sets are the 20 sequences 4, 42, 46, 65, 92, 94, 98, 106, 115, 119, 121, 124, 146, 173, 174, 181, 184, 186, 190, 193 on KITTI, the original *val_selection_cropped* split from the KITTI depth completion data, the sequences *alley_2*, *ambush_4*,

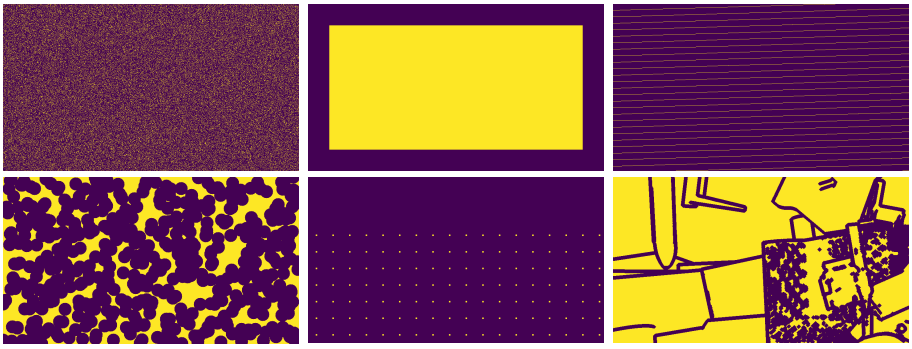


**Figure 5.16.:** *Different patterns for random sparsification. Yellow indicates selected pixel, remove pixels are purple.*

*bamboo_2*, *cave_4*, *market_5* for Sintel that sum up to 223 frames, and the sequences 0, 5, 15, 16, 18, 19, 27, 31 for HD1K.

**Automatic Sparsification.** For large size data sets like FT3D, it is infeasible to compute the actual sparse input of existing sparse-to-dense pipelines, due to the high run times of several seconds up to one minute per frame. Instead and because FT3D is only used for pre-training, a randomized sparsification process is introduced to simulate the sparse or non-dense input for interpolation. Various strategies are implemented to remove parts of the dense ground truth, of which one is randomly selected during training as part of the augmentation.

- Removing a uniformly random sampled ratio between 50 an 99 % of the dense input. This simulates a sparse pattern (isolated points) at different levels of sparsity.
- Removing a boundary frame at the edges of the ground truth with a random thickness of 10 to 25 % of the smaller spatial dimension. This simulates the common pattern of out-of-bounds motion that originates from fast ego-motions.
- Removing randomly oriented stripes with randomized thickness.
- Removing randomly sized circles at random (possibly overlapping) positions until 5 to 50 % are left. Both previous strategies provide a trade-off between large gaps and large valid information that do not align with the image content.
- Sampling sparse points in a quadratic grid with a random number of 4 to 64 horizontal lines and points per line. This roughly mimics the pattern of a very sparse LiDAR sensor.
- Removing randomly dilated regions around the motion discontinuities as provided by FT3D. This forces the network explicitly to learn interpolation boundaries based on the image information.
- Removing object regions from a randomly sampled sequence and frame of FT3D, to have irregular shaped large gaps.

Some examples of the automatic sparsification are given in Figure 5.16.

Additionally, random Gaussian noise ($\sigma$ = 2 px) is added to all remaining valid

pixels to simulate inaccuracies of a real matching process. For the experiments on optical flow and scene flow interpolation, SSGP is first trained on FT3D [MIHF+16]. The KITTI depth completion data set is sufficiently large to train on it directly. Pre-training is performed for one million iterations which corresponds to approximately 64 epochs. Afterwards, training is continued on the respective target domain and task with the pre-trained weights for initialization. For pre-training, photometric image augmentation is applied as in [DFIH+15].

**Loss Function.** The objective for training depends on the specific interpolation problem at hand. For motion fields, the average Euclidean distance between predicted $\hat{\boldsymbol{s}}$ and ground truth $\boldsymbol{s}$ motion vectors is minimized.

$$\mathcal{L}_{flow} = \frac{1}{N_{GT}} \cdot \sum_{\boldsymbol{s} \in GT} \|\hat{\boldsymbol{s}} - \boldsymbol{s}\| \tag{5.3}$$

This loss function is equally used for optical flow $\hat{\mathbf{u}}$ and scene flow $\hat{\mathbf{s}}$. For single valued depth, the mean squared error between ground truth $D$ and prediction $\hat{D}$ is optimized.

$$\mathcal{L}_{depth} = \frac{1}{N_{GT}} \cdot \sum_{D \in GT} \left(\hat{D} - D\right)^2 \tag{5.4}$$

**More Details.** Except for the two final linearly activated layers, ReLU activation [GBB11] is used for all convolutional layers. Adam [KB15] with an initial learning rate of $10^{-4}$ is used. The learning rate is continuously reduced with an exponential decay rate of 0.8 after every 10 % of the total number of steps. Due to hardware constraints, the training is limited to a batch size of 1 for all experiments. For training stability and improved generalization, all input of the network is normalized according to the statistics of the respective image and sparse data to zero mean and unit variance.

### 5.2.4. Experiments and Results

Three sets of experiments are presented. The first one is an ablation study on the different components of the architecture to clarify the contributions and validate the impact. Then, the robustness of SSGP is demonstrated in terms of noisy input, wrong input, changes of density of the input, and padding artifacts. Lastly, SSGP is compared to state-of-the-art on various data sets and interpolation tasks.

For flow interpolation, the metrics under considerations are the ones described in Section 2.3.2. For depth completion, the default mean absolute error (MAE) and the root mean squared error (RMSE) are measured.

To obtain the sparse input for the experiments with optical flow, the prominent FlowFields (FF) [BTS15] or its extension FlowFields+ (FF+) [BTS19] are used along with their competitor CPM [HSL16]. There is also a longer history of sparse matching techniques in optical flow [HS12; WRHS13]. However latest interpolation approaches [HLS17; RWHS15] show that these are superseded by

**Table 5.9.:** *Ablation study. Different concepts for sparse-to-dense interpolation of LiDAR measurements are compared on the validation split of KITTI. Mean absolute error (MAE) [mm], root mean squared error (RMSE) [mm], number of parameters (Params, $\times 10^6$) and floating point operations (FLOPs, $\times 10^9$) are presented.*

| Guide | Sparse | Flat | Refine | MAE | RMSE | Params | FLOPs |
|---|---|---|---|---|---|---|---|
| **none** | yes | yes | no | 356 | 1171 | **0.93** | **41.2** |
| **enc** | yes | yes | no | 312 | 1013 | 4.32 | 148.5 |
| **dec** | yes | yes | no | 289 | **953** | 4.47 | 149.5 |
| **full** | yes | yes | no | **288** | 957 | 4.61 | 156.9 |
| enc | no | no | no | 280 | 929 | 6.49 | 250.1 |
| full | **yes** | **no** | no | 276 | 915 | 10.14 | 382.4 |
| full | **no** | **no** | no | 270 | 910 | 10.14 | 381.3 |
| full | **yes** | **yes** | no | 288 | 957 | **4.61** | 156.9 |
| full | **no** | **yes** | no | **267** | **908** | **4.61** | **155.8** |
| full | yes | no | **yes** | 260 | 892 | 10.15 | 384.7 |
| full | no | no | **yes** | 251 | 881 | 10.15 | 383.6 |
| full | yes | yes | **yes** | 260 | 910 | **4.61** | 159.2 |
| full | no | yes | **yes** | **248** | **877** | **4.61** | **158.1** |

the FlowFields family or CPM. The similar matching concepts for scene flow correspondences of Chapter 4 are used as input for the experiments on scene flow. To the best of the author's knowledge, these are the only approaches which have tested the sparse-to-dense approach for scene flow. For the problem of depth completion, the sparse input is obtained directly from a LiDAR sensor.

### Ablation Study

Part of the contributions is the combination of sparsity-awareness and efficient spatial propagation for full guidance into an end-to-end interpolation network. Therefore in this section, SSGP is compared to equivalent networks that differ only conceptually from its design. All the results of the ablation study are reported in Table 5.9. As a first step, it is validated that the fusion of image data into the sparse target domain (image guidance) is beneficial, especially when image data is available anyways. Towards this goal, an *unguided* version of the sparse-to-dense codec is evaluated, i.e. the input image is not used at all and the RGB branch is removed. Whenever the ablation removes the Sparse Spatial Guided Propagation, it is replaced by a spatially invariant $3 \times 3$ convolution. Different variants of guidance are tested as well. Guidance is removed from either the encoder or decoder of the sparse-to-dense codec and the results are compared to the fully guided approach. It is obvious that guidance improves the results significantly. Furthermore, guidance in the encoder alone (*enc*) performs not as good as in later stages of the network (*dec*), or during all stages (*full*). The latter two variants perform on a par, but it is observed that

full guidance improves the results in difficult scenarios without much additional computational effort.

Besides networks which use regular convolution wherever SSGP uses sparse convolution (*sparse*), also networks which compute either a full affinity volume for guidance or a single affinity map (*flat*), are compared. Because LiDAR measurements have a quite regular pattern across all samples, the network variants without sparse convolution perform slightly better in general than the versions with sparse convolution. Anyways, in the next section it is shown that sparse convolution introduces higher robustness in case this property is not fulfilled. The flat versions reduce the network size and computational complexity by more than 50 % without much loss of accuracy. In fact, the version with flat guidance and regular convolutions performs the best. In later experiments with smaller data sets, the impact of flat guidance is found to be even more effective in reducing over-fitting. Lastly, dense *refinement* improves the results for all variants with very little increase in the number of parameters or Floating Point Operations (FLOPs).

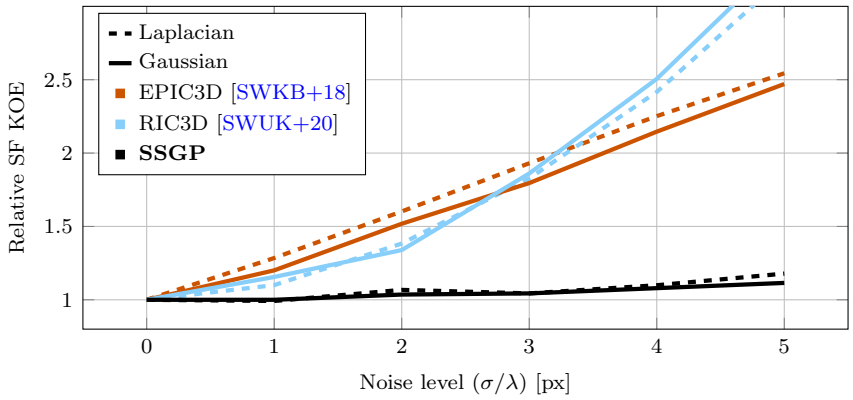
The fifth row in Table 5.9 represents a setup which is conceptually comparable to GuideNet [TTFL+20], i.e. guidance is only used in the encoder, the network is not sparsity-aware, and guided propagation uses the full affinity volume. This setup is called *GuideNet-like* in the comparisons.

#### Robustness

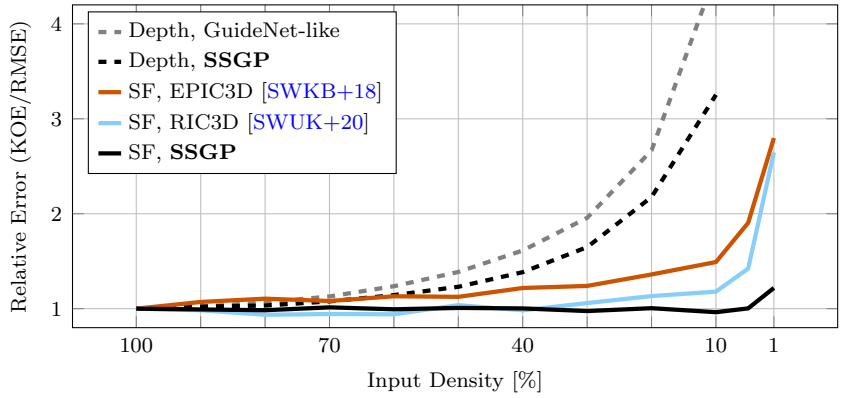
In this section, the robustness of SSGP is demonstrated. SSGP is evaluated when the input is deteriorated with random noise, and when the density is reduced by random sampling. Both results are presented in Figure 5.17. For the experiment with noisy input, random Gaussian or Laplacian noise with zero mean and different values of standard deviation $\sigma$ or exponential decay $\lambda$, are added to all valid points of the sparse input. Then, the relative increase of outliers for different levels of noise and different interpolation approaches with respect to the unaltered input of the interpolated scene flow are compared. Figure 5.17a clearly shows, that SSGP is extremely robust even to very noisy input. The outlier rate is maintained almost constant, while the competing methods perform considerably worse even for small amounts of additive noise.

In a second experiment, it is validated that the contribution of sparse convolution during guided propagation and the rest of the sparse-to-dense codec introduces higher invariance to the level of sparsity. Towards this end, depth completion and scene flow interpolation are performed with randomly sparsified input. Results are presented in Figure 5.17b. The increase of errors for the sparsity-aware model is about 50 % less when considering very sparse depth measurements. For SSGP on scene flow data (*SF*), the impact of the sparisfication is neglectable until exceeding less than 5 % of the original density. Note that all models are trained on the full input density. This improved robustness also applies to changes in the pattern of the input, e.g. when the LiDAR measurements are sparsified non-uniformly by one of the patterns in Figure 5.16.

An additional indicator for the robustness of SSGP is the *outlier rejection*




**(a)** *Results for scene flow interpolation when the input is superposed with different types and levels of random noise.*




**(b)** *Relative increase of errors when the input for different interpolation tasks is uniformly sparsified.*

**Figure 5.17.:** *Experiments on the robustness of SSGP. The input is altered with additive Gaussian and Laplacian noise (a) or random sparsification for depth completion and interpolation of scene flow (b). The proposed architecture is most robust to any type or level of degradation.*

**Table 5.10.:** *Evaluation of scene flow interpolation on the validation split of the KITTI scene flow data set. KITTI outliers (KOE) [%], end-point error (EPE) [px], and run time [s] are reported.*

| Input | Method | D0 | | D1 | | OF | | SF | | Run |
|---|---|---|---|---|---|---|---|---|---|---|
| | | KOE | EPE | KOE | EPE | KOE | EPE | KOE | ΣEPE | time |
| SFF | EPIC3D [SWKB+18] | 12.83 | 1.88 | 17.80 | 11.49 | 29.62 | 112.1 | 31.72 | 125.4 | 1.0 |
| | RIC3D [SWUK+20] | 9.88 | 1.92 | 13.94 | 2.79 | **15.44** | 8.42 | **17.45** | 13.10 | 3.8 |
| | **SSGP** | **9.06** | **1.33** | **13.93** | **1.83** | 20.67 | **5.04** | 25.19 | **8.20** | **0.19** |
| SFF++ + SDC | EPIC3D [SWKB+18] | 6.74 | 1.30 | 10.83 | 1.96 | 15.65 | 6.23 | 17.91 | 9.49 | 1.0 |
| | RIC3D [SWUK+20] | 5.91 | 1.29 | **7.24** | 1.53 | **9.80** | 3.33 | **11.50** | 6.15 | 3.8 |
| | **SSGP** | **5.71** | **1.04** | 9.89 | **1.45** | 12.39 | **3.00** | 16.61 | **5.50** | **0.19** |

*rate* (ORR), i.e. the percentage of input that is classified as scene flow outlier before interpolation, but is corrected during interpolation. Considering input from SFF and SFF++, EPIC3D achieves ORRs of 51.2 % and 40.3 %, RIC3D achieves 64.2 % and 55.7 %, and SSGP yields ORRs of **67.6 %** and **56.7 %**.

To show the robustness of sparse convolution to padding, the errors at boundary regions of the image are investigated. While the *GuideNet-like* variant obtains an MAE and RMSE of 186 and 505 mm in regions which are less than 10 px away from the image boundary, the full setup of SSGP achieves **140** and **448 mm**.

### Interpolation

**Scene Flow.** As first application to the interpolation network, the sparse matches from Chapter 4 of SFF and SFF++ (together with the SDC feature descriptor from Section 5.1) are used for the interpolation into dense scene flow. The results are computed on the KITTI data set [MG15] and are compared to EPIC3D [SWKB+18] and RIC3D [SWUK+20] which are the heuristic two-stage interpolators of SFF and SFF++, respectively. Both use additional edge information of the scene. Results are given in Table 5.10.

SSGP achieves competitive performance to previous methods, though being significantly faster. In the interpolation of initial disparity (*D0*), SSGP outperforms the baselines. Further, SSGP performs comparatively well in the EPE metric, which was also the objective function during training.

**Optical Flow.** For the experiments related to optical flow, multiple data sets are used for the evaluation, namely KITTI [MG15], HD1K [KNHK+16], and Sintel [BWSB12]. The method and state-of-the-art are evaluated for two kinds of input matches generated from FF+ [BTS19] and CPM [HSL16]. The proposed approach is compared to EPICFlow [RWHS15], RICFlow [HLS17], and InterpoNet [ZW17]. Note, that all three methods use additional explicit boundary information, while SSGP operates on the raw image. A visual comparison for a cropped frame of KITTI is presented in Figure 5.18. In this example, SSGP presents a globally consistent result, even in the static part of the scene, where small deviations have most impact in the visualization. SSGP shows the most accurate and sharp object contours, even though it is not

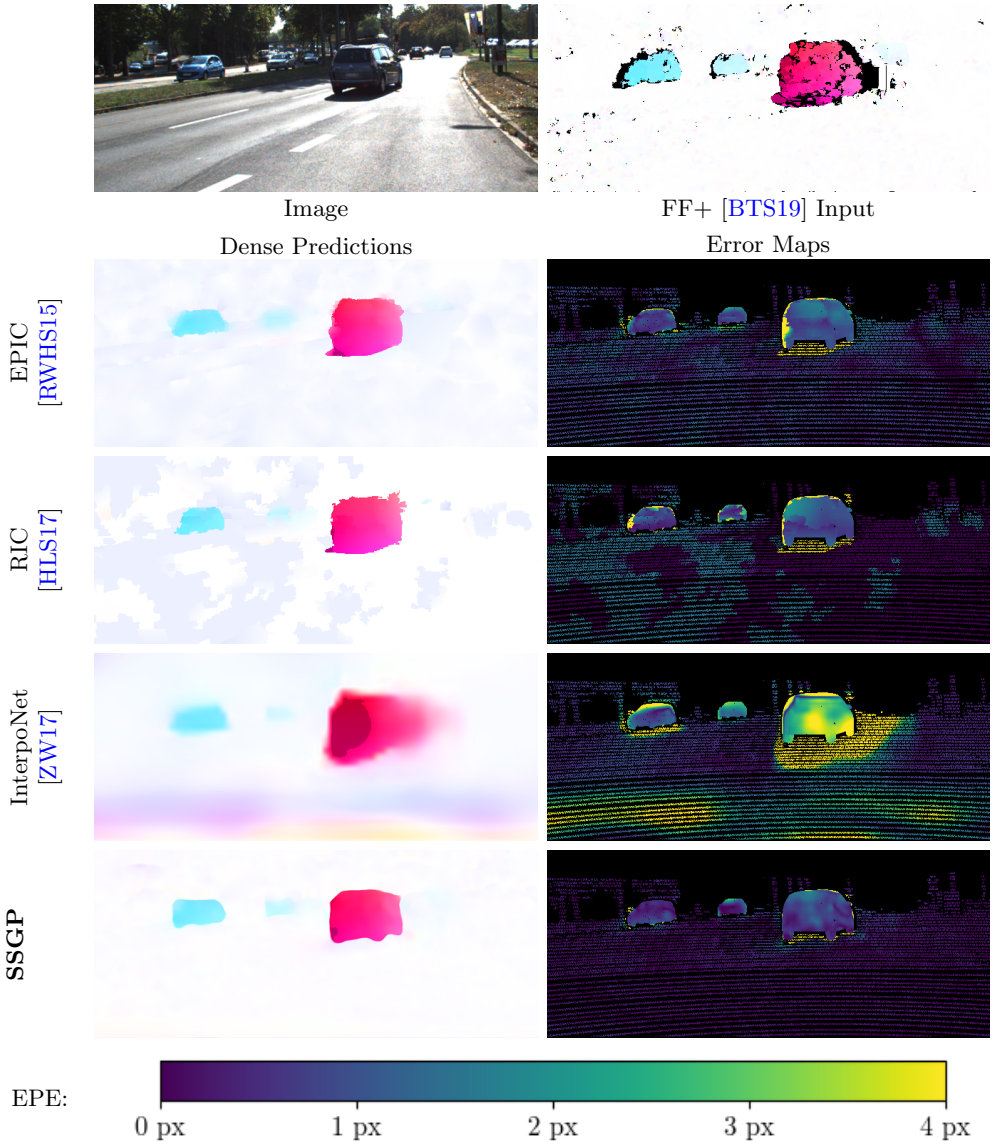



**Figure 5.18.:** *Visual comparison of interpolated optical flow on the KITTI data set.*

**Table 5.11.:** *Evaluation of interpolated optical flow. Tests are on the validation splits of the KITTI, HD1K, and Sintel data sets. Outlier rates (KOE) [%], end-point error (EPE) [px], and run time [s] are reported.*

| Input | Method | KITTI | | HD1K | | Sintel clean | | Sintel final | | Run time |
|---|---|---|---|---|---|---|---|---|---|---|
| | | KOE | EPE | KOE | EPE | KOE | EPE | KOE | EPE | |
| CPM [HSL16] | EPICFlow [RWHS15] | 24.39 | 10.04 | 5.43 | 1.11 | 9.98 | **3.84** | 13.94 | **5.76** | 0.4 |
| | RICFlow [HLS17] | 21.98 | 9.91 | 5.02 | 1.09 | **9.17** | 4.05 | **13.60** | 5.88 | 2.8 |
| | InterpoNet [ZW17] | 40.38 | 12.81 | 12.3 | 2.36 | 14.94 | 4.75 | 18.09 | 6.24 | 0.3 |
| | **SSGP** | **20.26** | **5.02** | **4.32** | **0.83** | 14.97 | 5.63 | 20.33 | 7.27 | **0.16** |
| FF+ [BTS19] | EPICFlow [RWHS15] | 23.97 | 11.34 | 5.55 | 1.21 | 11.25 | **5.05** | 15.99 | **7.26** | 0.4 |
| | RICFlow [HLS17] | 20.46 | 10.17 | 4.88 | 1.07 | **10.59** | 5.59 | **15.82** | 8.19 | 2.8 |
| | InterpoNet [ZW17] | 37.08 | 11.34 | 13.1 | 2.35 | 16.49 | 5.7 | 20.51 | 7.64 | 0.3 |
| | **SSGP** | **20.34** | **5.21** | **4.54** | **0.85** | 16.53 | 6.55 | 22.20 | 8.43 | **0.16** |

provided with pre-computed edge information. This highlights the capabilities of the full guidance strategy. In fact, SSGP is even able to reject wrong matches in shadows of the vehicles during interpolation.

Table 5.11 compares quantitative results for the entire validation sets. It is to highlight that SSGP cuts the end-point error on KITTI by about half in the comparison. On KITTI also, the outlier rates of SSGP beat all previous work. For completeness and fairness, it needs to be mentioned that InterpoNet [ZW17] is evaluated using the publicly available pre-trained weights that have been fine-tuned on Sintel with input from DF [MHG15] and on KITTI with matches from FlowFields [BTS15]. However, this indicates that InterpoNet is not very robust to changes of the input. On Sintel, SSGP is on par with InterpoNet, but lags behind the other methods. This is likely due to the limited variance between scenes which makes it hard to train a deep model for a complex task on Sintel and the more complex motions included in these scenes. Yet on HD1K, SSGP outperforms state-of-the-art in all metrics while also being faster.

**Depth Completion.** SSGP can also be used for the completion of sparse LiDAR measurements. The entire architecture is trained from scratch on the KITTI depth completion data set [USSF+17] and the results are compared to state-of-the-art in Table 5.12. The network again achieves a competitive result on yet another challenge, indicating its broad applicability. A visual example of an interpolated depth map is given in Figure 5.19. It is further noticed that RIC3D [SWUK+20] (Section 4.2.2), a top-performing method for interpolation of scene flow, performs considerably worse than any other approach. This shows, that even though RIC3D is not a learning-based method, it has a strong dependency on properly selected hyper-parameters.

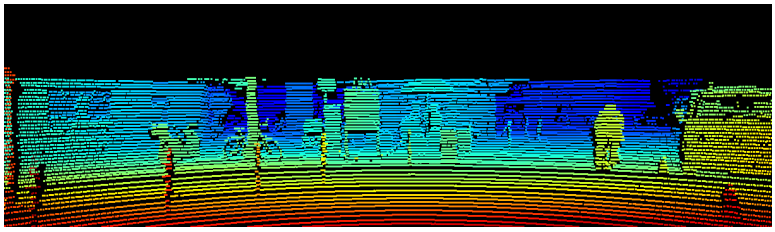
### 5.2.5. Conclusion

SSGP successfully combines sparsity-aware convolution and an efficient version of spatially variant propagation for fully image-guided interpolation. The network design is applicable to diverse sparse-to-dense problems and achieves

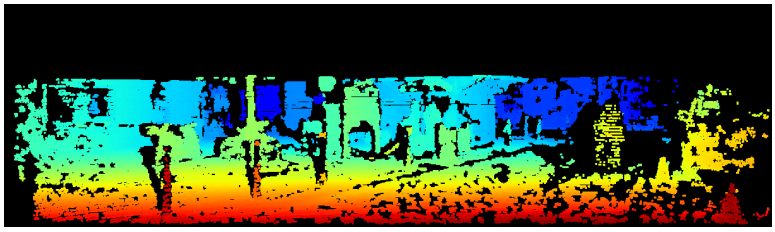
**Table 5.12.:** *Comparison of methods for depth completion on the KITTI benchmark [USSF+17]. The mean average error (MAE [mm]), root mean squared error (RMSE [mm]), and run time [ms] are reported for the best performing, published methods using image guidance out of more than 90 total submissions. Values in gray are computed on the validation split.*

| Method | MAE | RMSE | Run time |
|---|---|---|---|
| GuideNet [TTFL+20] | 219 | **736** | 140 |
| CSPN++ [CWGY20] | 209 | 744 | 200 |
| FuseNet [CYLU19] | 221 | 753 | 90 |
| DeepLiDAR [QCZZ+19] | 227 | 758 | 70 |
| MSG-CHN [LYLC+20] | 220 | 762 | **10** |
| Guide&Certainty [VNDV19] | 215 | 773 | 20 |
| PwP [XZSZ+19] | 235 | 777 | 100 |
| CrossGuidance [LLKK20] | 254 | 807 | 200 |
| Sparse-to-Dense [MCK19] | 250 | 815 | 80 |
| NConv-CNN [EFK19] | 233 | 830 | 20 |
| DDP [YWS19] | **204** | 833 | 80 |
| **SSGP** [SWUS21] | 245 | 838 | 140 |
| Spade [JDWP+18] | 235 | 918 | 70 |
| DFineNet [ZNMC+19] | 304 | 945 | 20 |
| CSPN [CWY18] | 279 | 1020 | 1000 |
| RIC3D [SWUK+20] | 588 | 2477 | 1400 |

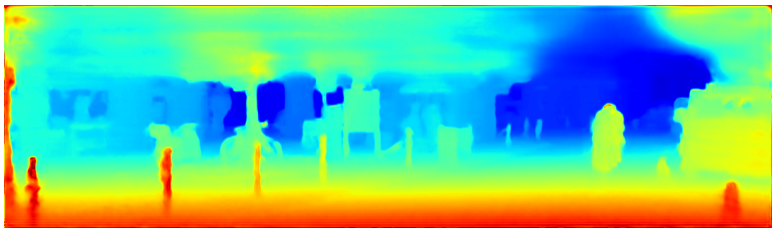

**(a)** *Input image*



**(b)** *LiDAR measurements (visually enhanced)*



**(c)** *Ground truth (visually enhanced)*



**(d)** *Densified depth with SSGP*

**Figure 5.19.:** *An example of depth completion on KITTI [GLU12] data with SSGP. Even very small details are well preserved.*

competitive performance throughout most experiments, beating state-of-the-art in interpolation of optical flow in two out of the three data sets and in terms of EPE for scene flow interpolation. A flat affinity map can be used for spatial guidance equally well as a full affinity volume, drastically reducing the overall network size. This strategy for guidance resolves the dependency on explicitly pre-computed edge information resulting in even more accurate interpolation boundaries with a globally consistent output that preserves fine details. SSGP is especially robust against variations of the level of sparsity and sparsity pattern, or noise in the input.

## 5.3. Summary

This chapter has introduced two deep modules which are applied in the sparse-to-dense pipeline. The first module tackles the problem of feature representation of corresponding image points under prevalent changes of appearance. The second module deals with the problem of occlusion in conformity with the sparse-to-dense principle. Both solve the problem of data scarcity by relating the respective problem to other domains and similar tasks. By that, a broader range of data from diverse data sets can be used during training. As a result, higher robustness is achieved, which is further amplified by a task-specific architectural design. Furthermore, the parallelization of deep neural networks boost the run time of the replaced modules.

Chapter 6

# Sparse-to-Dense Combination Approach

*Divide et impera.*

— Philip II of Macedon

## Contents

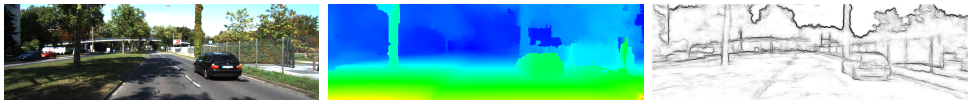


This chapter is backed by the following publications:

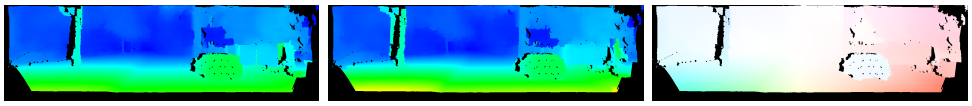
**René Schuster**, Christian Bailer, Oliver Wasenmüller, and Didier Stricker. "Combining Stereo Disparity and Optical Flow for Basic Scene Flow." In: *Commercial Vehicle Technology Symposium (CVT)*. 2018.

**René Schuster**, Oliver Wasenmüller, and Didier Stricker. "Dense Scene Flow from Stereo Disparity and Optical Flow." Extended Abstract for the Computer Science in Cars Symposium (CSCS). 2018.

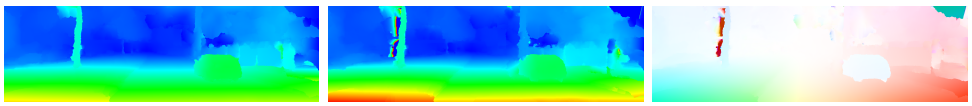
**René Schuster**, Christian Unger, and Didier Stricker. "MonoComb: A Sparse-to-Dense Combination Approach for Monocular Scene Flow." In: *Computer Science in Cars Symposium (CSCS)*. 2020.



**(a)** *Reference image at time t.* **(b)** *Estimated disparity at time t + 1.* **(c)** *Edges used for interpolation.*



**(d)** *Sparse disparity at time t.* **(e)** *Warped, sparse disparity of (b).* **(f)** *Sparse optical flow from t to t + 1.*



**(g)** *Interpolated disparity at time t.* **(h)** *Interpolated disparity of (e).* **(i)** *Interpolated optical flow.*

**Figure 6.1.:** *Optical flow and stereo disparity are combined to sparse scene flow by warping. Edge-aware interpolation is used to reconstruct a dense scene flow field.*

The sparse-to-dense concept separates scene flow estimation into independent steps to allow for easier individual solutions. Analogously, the combination approach splits the difficult problem of scene flow estimation into several smaller auxiliary tasks. This way, well-studied existing solutions can be reused in a modular fashion to obtain scene flow. This strategy is presented and investigated in this chapter. It is important to note that the sparse-to-dense concept of this thesis enables a dense estimation of scene flow together with the combination approach. In general, this was not possible before. Also, one claim is that the interpolation unifies the auxiliary results to overcome inconsistencies. Section 6.2 exploits the separation further to reduce the sensor suite from a stereo camera to a monocular setting.

## 6.1. Scene Flow from Stereo Disparity and Optical Flow

Because of the complexity of the scene flow problem there exists no applicable variant for real-time scene flow estimation in an automotive context that is sufficiently robust and accurate. The combination of top-performing state-of-the-art optical flow and stereo disparity algorithms is used to achieve a basic, non-dense scene flow. The sub-tasks can be considered computationally less expensive and the combination itself is negligible in terms of overhead. In a first step, a non-dense scene flow result is obtained by a combination of disparity and optical flow. Its reasonable accuracy and computational efficiency is demonstrated on the KITTI scene flow benchmark [MG15], where many previous, dedicated scene flow algorithms are outperformed. Together with a

considerably fast and accurate interpolation it is possible to reconstruct dense scene flow.

### 6.1.1. Similar Approaches and Auxiliary Estimators

In the literature, there are attempts to compute scene flow from a combination of stereo depth and 2D optical flow, but back then optical flow estimation was not as advanced as it is now. The pre-computation of depth to generate input for methods that require RGB-D images for scene flow estimation can be considered an alternate form of the proposed combination approach, e.g. as in [HFR14; JSGC15].

**Auxiliary Disparity Estimation.** Two very popular stereo algorithms are Semi-Global Matching (SGM) [Hir08] and Slanted Plane Stereo (SPS) [YMU14] because they achieve reasonable accuracy and speed at the same time. That is why they are often used as standalone algorithms or for initialization purposes. Stereo matching in SGM is not using a local nor global neighborhood for regularization. Instead a semi-global energy formulation which uses eight different paths radiating from each target pixel location is used, enabling sharp boundaries, accurate depth estimation and good run time. Because of occlusions by the camera frustum, it is hard to recover depth for all image points. Yet, it is possible to detect potentially wrong estimates through consistency checks. SGM uses a left-right consistency check for two computed depth maps to localize and remove wrongly estimated values. Thus, SGM yields a non-dense disparity map. SPS applies the slanted plane model of [VSR13] to estimate piece-wise planar surfaces of the scene. Given the strong regularization of planar patches, the method is able to handle occlusions to some extend, yielding a dense result.

**Auxiliary Optical Flow.** FlowFields [BTS15] tackles the problem of optical flow estimation without any regularization, which makes it versatile. It is tailored to find pixel correspondences for optical flow estimation by propagation and random search with multiple stages of outlier filtering followed by interpolation with EpicFlow [RWHS15] to reconstruct an accurate, dense optical flow field. FlowFields+ [BTS19] is the extension of FlowFields that uses an enhanced matching term. Similar to the stereo algorithms, the full resolution matching result is filtered to remove outliers. EpicFlow [RWHS15] is used to fill up the gaps. The latest improvement of FlowFields+ combines it with the robust interpolation of RICFlow [SBWS18b]. Another version uses deep learning to match correspondences across the images [BVS17]. These methods are noteworthy because they were among the first to achieve top performance across different data sets, which makes them very versatile.

### 6.1.2. Combination Approach

The standard dual-frame stereo camera setup is used for the recombination, i.e. two rectified, temporally adjacent frame pairs (see Figure 6.2). These four

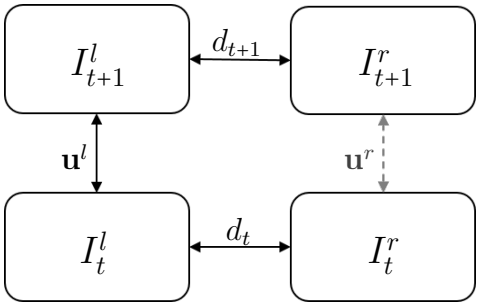



**Figure 6.2.:** *Relation of two stereo image pairs. Superscripts denote the viewpoint (left/right) and subscripts the time step. The left image at time $t = 0$ is considered the reference frame. Each stereo image pair is related by the according disparity map, while the temporal image pairs are related by 2D optical flow.*

images provide sufficient information to estimate 3D scene flow.

To avoid greater overhead in computation and still estimate a full scene flow representation in 3D, depth information and 2D flow are combined. Depth is estimated by stereo disparity using only a single image pair at a time. Motion information is estimated by optical flow using only consecutive images from the left camera over time. Both complement each other regarding scene flow. Optical flow is lacking depth information, and stereo depth can be generalized to change over time.

**Sparse Combination by Warping.** Given the camera intrinsics and extrinsics, full scene flow can be described in image space by optical flow, disparity, and change of disparity. Consequently, direct computation of optical flow and disparity solves two sub-tasks of scene flow estimation. What is missing is the change of disparity $\Delta d_{t\to t+1}$ that together with the disparity $d_t$ yields the disparity at the next time step $d_t + \Delta d_{t\to t+1}$ for each pixel of the reference time step. However, a disparity map $d_{t+1}$ with reference to the next time step can be computed directly and together with the optical flow, that relates corresponding pixels between both time steps. This disparity map can be warped back to the reference frame.

$$\mathrm{d}_{t,w}(x,y) = \mathrm{d}_{t+1}(x+u, y+v). \tag{6.1}$$

$\boldsymbol{u}(\mathbf{p}) = (u,v)^T$ are the optical flow components at pixel $\mathbf{p} = (x,y)^T$. Bi-linear interpolation is used to warp disparity values from sub-pixel positions:

$$\begin{aligned}\mathrm{d}_{t,w}(x,y) = {} & \mathrm{d}_{t+1}(\lfloor x' \rfloor, \lfloor y' \rfloor) \cdot (1 - \{x'\}) \cdot (1 - \{y'\}) \\ & + \mathrm{d}_{t+1}(\lfloor x' \rfloor + 1, \lfloor y' \rfloor) \cdot \{x'\} \cdot (1 - \{y'\}) \\ & + \mathrm{d}_{t+1}(\lfloor x' \rfloor, \lfloor y' \rfloor + 1) \cdot (1 - \{x'\}) \cdot \{y'\} \\ & + \mathrm{d}_{t+1}(\lfloor x' \rfloor + 1, \lfloor y' \rfloor + 1) \cdot \{x'\} \cdot \{y'\}\end{aligned} \tag{6.2}$$

whereas $\mathbf{p}' = (x', y')^T$ with $\mathbf{p}' = \mathbf{p} + \boldsymbol{u}$ is the target position of pixel $\mathbf{p}$, and $\{x\} = x - \lfloor x \rfloor$ denotes the fractional part of number $x$.

Thus, (non-dense) scene flow can be reconstructed from optical flow and two disparity maps. Formally, the reconstruction can be described as

$$\mathbf{s}(\mathbf{p}) = \begin{cases} (\mathbf{u}(\mathbf{p})^T, \mathrm{d}_t(\mathbf{p}), \mathrm{d}_{t,w}(\mathbf{p}))^T, & \text{if } \mathbf{p} + \mathbf{u}(\mathbf{p}) \in \Omega \text{ and} \\ & \mathrm{d}_t(\mathbf{p}), \mathrm{d}_{t,w}(\mathbf{p}) \text{ valid} \\ \text{undefined}, & \text{otherwise} \end{cases} \tag{6.3}$$

The remaining problem is that the reconstruction fails if the optical flow leaves the image boundaries, or where the depth information contains gaps. As a result, the reconstruction approach produces a non-dense scene flow field (cf. Figures 6.1d to 6.1f). This is also reflected in the results shown in Table 6.2 where the evaluation results for the originally estimated sparse scene flow (*Est*) and for a dense version (*All*) that was interpolated by KITTI during evaluation are presented.

**Occlusion Estimation.** Furthermore, the warping introduces errors where the scene is occluded. To overcome this, occluded regions need to be masked. Since the disparity maps provide depth information, occlusions can be estimated explicitly. This is done, by comparing depth values of pixels that have the same target position according to the optical flow, and masking all but the closest point. It is formalized in Equation 6.4 by

$$occ(\mathbf{p}) = \left[ d_t(\mathbf{p}) < d_t(\mathbf{p}') \forall \mathbf{p}' \in \Omega \mid \mathbf{p}' + \mathbf{u}(\mathbf{p}') \approx \mathbf{p} + \mathbf{u}(\mathbf{p}) \right] \tag{6.4}$$

to obtain a binary occlusion mask *occ* for each pixel in the image domain $\Omega$, where $[\bullet]$ denotes the Iverson bracket, and sub-pixel optical flow is handled by rounding. After initial occlusion mask estimation, discretization and rounding errors are corrected by applying two iterations of morphological closing and opening. Exemplary results of a warped disparity map are shown in Figures 6.1e and 6.4.

**Interpolation.** To fill in all these gaps, the interpolation method of SceneFlowFields (SFF) [SWKB+18] (Section 4.1) can be used. Depending on the auxiliary methods that are used for optical flow and disparity estimation, the results for these tasks are already dense. Only the warped disparity map $d_{t,w}$ is non-dense. This leaves several options for the interpolation which are all compared in Table 6.1:

- Using the default interpolation algorithm of the KITTI submission system (*kitti*).
- Interpolating all sub-tasks where $\mathrm{d}_{t,w}$ has gaps (*full*).
- Interpolating the 3D motion only (flow + disparity change) according to local affine 3D transformation models (*motion*).
- Interpolating the warped disparity only, using the affine 3D transformations (*disp-affine*).
- Interpolating the warped disparity map only, using a local plane model (*disp-plane*).

### 6.1.3. Experiments and Results

Two combinations of auxiliary methods are evaluated. The first is SGM+FF+, the second is SPS+FF++. For both combinations the non-dense and the interpolated results are computed and presented in Tables 6.1 and 6.2. Visual examples of the results compared to other methods are given in Appendix A in Figures A.1 and A.2. These figures show the color encoded disparity maps and the optical flow along with the respective error maps. Figure A.2 illustrates that most errors in the proposed approach are introduced in areas where the sparse recombination method can not reconstruct scene flow, because motion leaves the image boundaries (shaded regions in the error maps).

Two major characteristics of the approach are discussed in more detail. Firstly, as explained before, there is a trade-off between density and accuracy. Secondly, the combination of stereo disparity and optical flow is fast compared to previous scene flow algorithms.

**Accuracy and Density.** Sparsity of course is not a desired result, yet the non-dense nature of the method leads to accurate results. This can be seen in Table 6.2 where the sparse results (*Est*) outperform other methods which combine stereo and optical flow as well as many of the dedicated scene flow algorithms. The interpolated results (*All*) are still better in comparison to SGM+C+NL [Hir08; SRB14] and SGM+LDOF [BM11; Hir08], two methods that also combine depth and optical flow to obtain scene flow in a similar way. This is an expected result because the used optical flow algorithm is ranked higher in the respective KITTI benchmark [GLU12]. However, interpolation of sparse scene flow as it is done by KITTI decreases the accuracy significantly. Other interpolation methods lead to a higher accuracy for dense results as shown in Table 6.1. Nevertheless, many of the dedicated scene flow algorithms like e.g. the variational approach of [HD07] are outperformed. In the end, the achieved density of about 81 % (cf. Table 6.2) is rather high considering that KITTI's data is recorded with a frame rate of 10 which means that large parts of the visible scene leave the image boundaries at the next time step, even for slow ego-velocities. In KITTI, density is given by the amount of available ground truth pixels that are covered by the results. For the non-occluded areas, i.e. areas that are also visible in the next frame, a density of 92.42 % is achieved. Of course there exist methods that are ranked higher in the KITTI scene flow benchmark. Most of these methods make further assumptions on the observed scene, which makes them less versatile. Furthermore, these methods solve scene flow estimation as a single task where geometry and 3D motion are estimated jointly. This allows for strong regularization mechanisms, e.g. the piece-wise rigid scene model that is used by [BJMA+17; LBAL+16; MG15; VSR13]. But typically, the problem formulation in these methods results in a complex energy term that requires a lot of computational effort to minimize. This is reflected by the considerably long run times of the top performing methods on KITTI.

**Table 6.1.:** *Comparison of different interpolation schemes. Percentage of outliers on KITTI training data. The sparse result is obtained by combining SPS+FF++.*

| Interpolation | D1 | D2 | Fl | SF | Density |
|---|---|---|---|---|---|
| *sparse* | 4.4 | 8.3 | 9.0 | 12.7 | 84.23 % |
| *kitti* | 11.0 | 17.5 | 19.9 | 23.8 | 100.0 % |
| *full* | 6.5 | **12.6** | 16.3 | **19.6** | 100.0 % |
| *motion* | **4.9** | 13.2 | 16.7 | 20.4 | 100.0 % |
| *disp-affine* | **4.9** | 13.2 | **15.5** | 20.8 | 100.0 % |
| *disp-plane* | **4.9** | 13.5 | **15.5** | 21.2 | 100.0 % |

**Dense Results.** The different options for interpolation are evaluated in Table 6.1. Remarkable about the different concepts is that the joint interpolation (an exemplary result of this variant is given in Figures 6.1g to 6.1i) produces the overall best scene flow estimate, though the different sub-results are less accurate than for some other interpolation strategies (see Table 6.1). This supports the general paradigm that scene flow should be estimated jointly. Further, it is to highlight that the sparse combination results are already very accurate. With the steady improvement of methods for the auxiliary tasks, scene flow estimation by recombination enhances also. Here, SPS [YMU14] is used for disparity estimation and FlowFields++ [SBWS18b] for the optical flow tasks. Both are ranked higher than the respective auxiliary methods (SGM [Hir08] and FF+ [BTS19]) that are used in [SBWS18a]. Due to this and because the interpolation algorithm of Section 4.1 is more sophisticated than the automatic interpolation of the benchmark, the dense scene flow estimate from stereo disparity and optical flow is also ranked higher (see Table 6.2).

**Run time.** The second important aspect of the combination approach is the fast run time. It can be concluded that the overall run time of any recombination method for scene flow is determined by the time for the computation of depth and optical flow. In the first case, the 29 seconds originate from 28 seconds computation time for FlowFields+ and 1 second for disparity computation for both time steps using SGM. The time of combination can be neglected. This means that sparse scene flow can be computed in real-time if real-time algorithms for the stereo and optical flow tasks are used. Even though the two subsidiary methods are not the fastest in their respective field, the computation of scene flow from stereo and optical flow is at least two times faster than most methods and about one order of magnitude faster than the top performing method (cf. Table 6.2). The run time of the approach using sparse-to-dense interpolation consists of 29 seconds for FlowFields++ [SBWS18b], 2 seconds for each disparity map computed with SPS [YMU14], and 3 seconds for dense interpolation with EPIC3D [SWKB+18].

**Table 6.2.:** *Results of the public KITTI scene flow benchmark [MG15]. Dual frame methods, i.e. methods that use only two consecutive frame pairs for computation are compared. Results are given as the average percentage of outliers according to the KITTI metric. Results of the non-dense combination approach are displayed for the originally submitted scene flow (Est) and for the automatic dense interpolation (All) of the submission system.*

| Method | D1 | D2 | Fl | SF | Density | Run time |
|---|---|---|---|---|---|---|
| ISF [BJMA+17] | 4.5 | **6.0** | **6.2** | **8.1** | 100.00 % | 600 s |
| OSF18 [MHG18] | 5.3 | 7.1 | 7.4 | 9.7 | 100.00 % | 390 s |
| SSF [RSKS17] | **4.4** | 7.0 | 7.1 | 10.1 | 100.00 % | 300 s |
| OSF [MG15] | 5.8 | 7.8 | 7.8 | 10.2 | 100.00 % | 3000 s |
| CSF [LBAL+16] | 6.0 | 10.1 | 13.0 | 15.7 | 100.00 % | 80 s |
| SFF [SWKB+18] | 6.6 | 10.7 | 12.9 | 15.8 | 100.00 % | 65 s |
| PRSF [VSR13] | 6.2 | 12.7 | 13.8 | 16.4 | 100.00 % | 150 s |
| **SGM + FF+ (Est)** | 4.7 | 10.6 | 11.8 | 19.8 | 81.24 % | **29 s** |
| **SPS + FF++ (+EPIC3D)** | 6.6 | 14.4 | 16.6 | 20.7 | 100.00 % | 36 s |
| SGM + SF [HFR14; Hir08] | 6.8 | 15.6 | 21.7 | 25.0 | 100.00 % | 2700 s |
| PCOF + LDOF [DPSL16] | 8.5 | 21.0 | 18.3 | 29.3 | 100.00 % | 50 s |
| **SGM + FF+ (All)** | 13.4 | 27.8 | 22.8 | 33.6 | 100.00 % | **29 s** |
| SGM + C+NL [Hir08; SRB14] | 6.8 | 28.25 | 35.6 | 40.3 | 100.00 % | 270 s |
| SGM + LDOF [BM11; Hir08] | 6.8 | 28.6 | 39.3 | 43.7 | 100.00 % | 86 s |

## 6.2. Dense Monocular Scene Flow from Single Image Depth and Optical Flow

One current trend regarding many applications in assisted or autonomous driving is the utilization and fusion of as many sensors as available. As a result, certain approaches have increasing requirements on the hardware in a product. A different strategy is to solve the same problems based on input from less sensors, which allows to have the same functionality at lower cost. This becomes possible by adding constraints or assumptions to the formulation of the problem, by technical progress, or by relying on more or other visual cues.

Following the strategy to reduce the sensors, a method to estimate scene flow in the purely monocular case is proposed. While a single RGB camera does not provide a geometric cue as in stereo cameras and is also unable to measure depth directly as a LiDAR or RGB-D camera, this setup poses scene flow estimation to be a much more difficult problem. To the rescue, latest developments in single image depth estimation prove that absolute depth can be reconstructed from a single viewpoint [FGWB+18; GMB17; GMFB19; LHKS19; OKM19]. This is possible by relying on depth cues like the monocular motion parallax, defocus blur by the field of depth, perspective transformation of parallel lines, texture gradients, or the relative size of known objects.

The progress in the field of single image depth estimation is exploited within the previously introduced combination approach to estimate scene flow in a monocular setting. Towards that end, 1.) depth is estimated from two

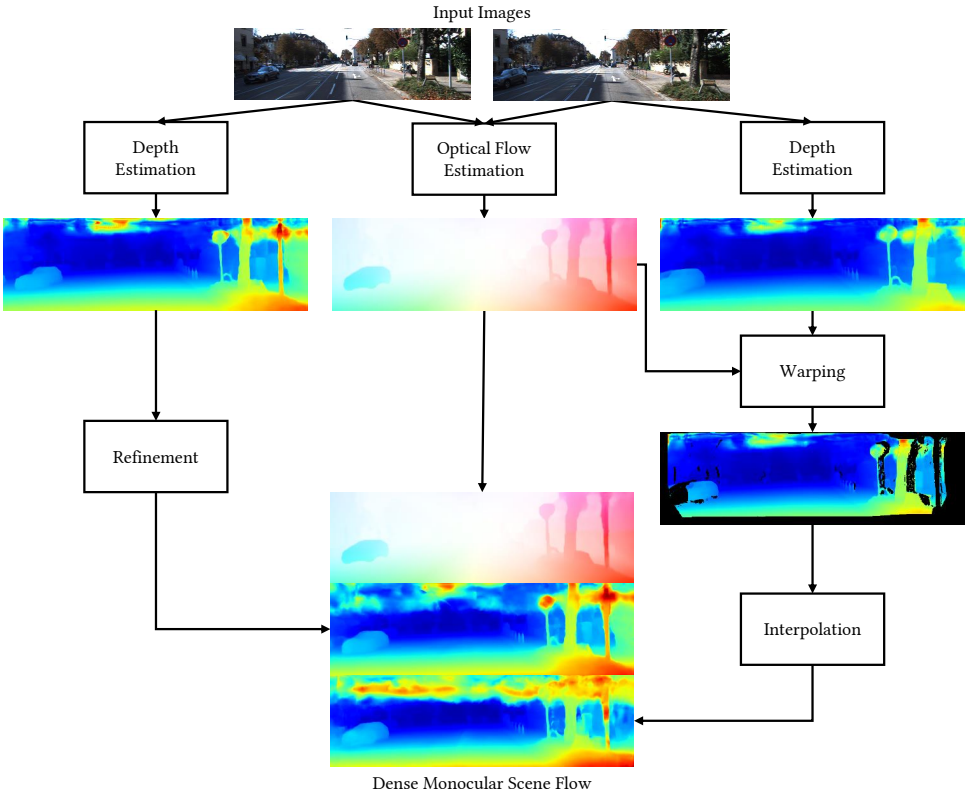



**Figure 6.3.:** *Overview of the pipeline for monocular dense scene flow estimation.*

single images, 2.) optical flow between these two images is estimated, 3.) corresponding 3D points are correlated to estimate 3D scene flow, and 4.) gaps due to occlusions are interpolated to obtain a dense result. The overview of these steps is given in Figure 6.3. Since the combination approach operates in a monocular camera setting, the proposed method is termed *MonoComb* [SUS20].

### 6.2.1. Monocular Scene Flow in the Literature

In the early beginning of scene flow estimation the problem has already been formulated for a monocular camera [VRCK05], however with a strong dependency on multiple views. There are a few more recent approaches considering the same setup. In Mono-SF [BAM19], pixel-wise depth distributions are estimated for both images and then used in a probabilistic optimization framework to estimate rigidly moving, planar segments for the scene. Though the superpixel segmentation provides a strong regularization and high accuracy, the assumptions of rigidity and planarity introduce errors depending on the granularity of the segmentation. Also, the optimization is computationally heavy. Self-Mono-SF [HR20] uses an adaptation of PWC-Net [SYLK18] that is trained in a self-supervised manner to jointly estimate 3D position and 3D flow at a quarter resolution. This method is able to achieve competitive results after supervised fine-tuning, but the purely unsupervised version lags behind. Lastly, Optical Expansion (OE) [YR20] exploits the assumption that most of the change of the projected size of objects is due to the change in distance to the observer. Therefore the authors argue that the relative motion-in-depth can be estimated directly together with optical flow. In conjunction with an estimate for the depth in one frame, the depth at the second frame can be reconstructed to obtain full scene flow.

Since most of the previous work in monocular scene flow estimation – as well as the proposed approach – rely on single image depth estimation, this area is discussed as well. LRC [GMB17] is a self-supervised approach that uses an encoder-decoder network architecture and trains with a photometric loss and consistency between the left and right view of a stereo camera. The approach is refined in MonoDepth2 [GMFB19]. DORN [FGWB+18] proposes to use a ordinal regression loss instead of a regular regression loss to train a network for single image depth estimation. This formulation is closer to monocular depth cues, where it is often easier to order the depth of regions instead of regressing the absolute depth of each point. BTS [LHKS19] represents state-of-the-art across different data sets. The idea of BTS is to fuse depth estimates from multiple scales at full resolution using local planar guidance for up-sampling.

### 6.2.2. Monocular Combination Approach

The following method is proposed to obtain dense scene flow from a single monocular image pair $I_t$ and $I_{t+1}$ at time steps $t$ and $t+1$. First, an off-the-shelf single image depth estimator is used to predict a dense depth map for each of the images, $\mathrm{D}_t$ and $\mathrm{D}_{t+1}$. Also, dense optical flow $\mathbf{u} = (u, v)^T$ from the first

to the second image is predicted with an auxiliary optical flow estimator. As before, the optical flow result is directly used within the combined scene flow, and further to estimate the (non-dense) change in depth, by warping $\mathrm{D}_{t+1}$ towards the reference frame at time $t$. At this point, a non-dense scene flow is obtained. Lastly, the gaps which originate during warping are interpolated to reconstruct the dense scene flow.

**Auxiliary Depth and Optical Flow Estimation.** In principle, any methods for depth and optical flow estimation can be used. The performance of the auxiliary methods directly influence the quality of the final scene flow result. Therefore in the experiments, state-of-the-art approaches are used. For optical flow this is VCN [YR19] and HD3 [YDY19], and BTS [LHKS19] for single image depth estimation.

While for optical flow the publicly available pre-trained weights are used, BTS is re-trained on the complete KITTI depth data set [GLU12; USSF+17] with a depth cap of 100 meters. Since the KITTI scene flow data set [GLU12; MG15] provides scene flow labels for the stereo setup in image space, i.e. disparity and optical flow displacements, all estimated depth values for a pixel $\mathbf{p}$ are further transformed into (virtual) disparity displacements using the available focal length $f$ and baseline $B$ of the stereo camera according to Equation 6.5.

$$\mathrm{d}_i(\mathbf{p}) = \frac{f \cdot B}{\mathrm{D}_i(\mathbf{p})} \tag{6.5}$$

Assuming calibrated cameras, both domains are interchangeable since both provide the necessary geometric 3D information.

**Warping and Occlusion Estimation.** The auxiliary estimators provide the basis for the combination approach. However, warping is the actual core and biggest challenge for the combination method. It is needed to correlate the 3D information of different pixels to find the shift over time. According to the previous section, warping is defined as in Equation 6.1.

During warping, target pixels outside of the image domain are ignored (cf. Equation 6.3) and bilinear interpolation is used to account for sub-pixel displacements in the estimated optical flow (cf. Equation 6.2). These out-of-bound regions lead to a non-dense warped disparity. Additionally, some points move in front or behind others so that not all points visible at time $t$ are also visible at time $t+1$. These occlusions need to be filtered, because they produce ghosting effects, i.e. duplicated objects, after warping. This is again done by applying Equation 6.4 with the same post-processing as in Section 6.1.2. An exemplary result of a warped disparity map and the corresponding occlusion mask is given in Figure 6.4.

By combining the initially estimated optical flow $\mathbf{u}$, the geometric information of $d_t$, and the warped disparity $d_{t,w}$, a non-dense scene flow can be obtained. This intermediate result is analyzed in Section 6.2.3.

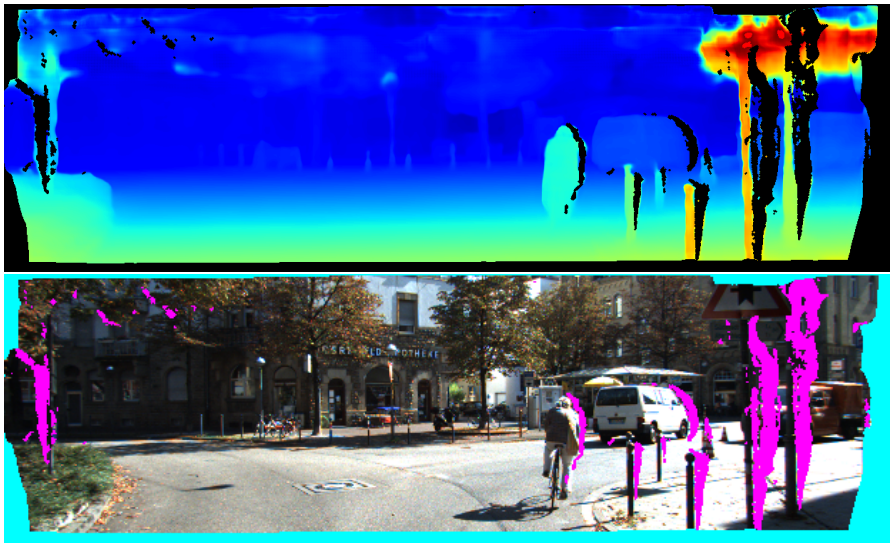


**Figure 6.4.:** *Illustration of the warping process with occlusion handling. The top images visualizes the geometry (virtual disparity) at time $t+1$ after warping towards the reference frame at time $t$. The bottom image shows the two types of obstruction, geometric occlusion (magenta) and out-of-view motion (cyan), which can be considered as occlusion by the camera frustum.*

**Interpolation and Refinement.** To recover full density, the gaps of the warped disparity $d_{t,w}$ have to be filled. The method of choice in the monocular case is Sparse Spatial Guided Propagation (SSGP) [SWUS21] from Section 5.2. It can be applied for the interpolation of optical flow, scene flow, or depth maps and is used to interpolate the gaps in the warped virtual disparity map. The interpolated output is named $d_{t,i}$. Because of the inverse target domain (disparity instead of depth), SSGP is re-trained on the KITTI depth data in the disparity space. This is achieved by the conversion of all predicted values and ground truth depth labels with Equation 6.5 during loss computation. An example for warped, sparse geometry and the interpolated result is given in Figure 6.5. Note how in this example even the fully occluded bush on the right side is reconstructed reliably by the image guidance of SSGP.

In the experiments in Section 6.2.3, it is shown that the interpolation is able to reconstruct full density (with respect to the image resolution) and additionally to improve the predicted geometric information compared to the warped, sparse result. This is mostly attributed to a correction of the absolute scale of depth (virtual disparity) and is similar to the two-stage depth estimation in Mono-SF [BAM19] where a dedicated network for re-calibration refines the initial depth estimates.

However, the observation that SSGP improves the results beyond interpolation suggests that this step might also improve already dense input, i.e. the (virtual) disparity estimate at time $t$. For this reason, the full model processes the dense estimate $d_t$ with SSGP before combining it into dense scene flow (cf.

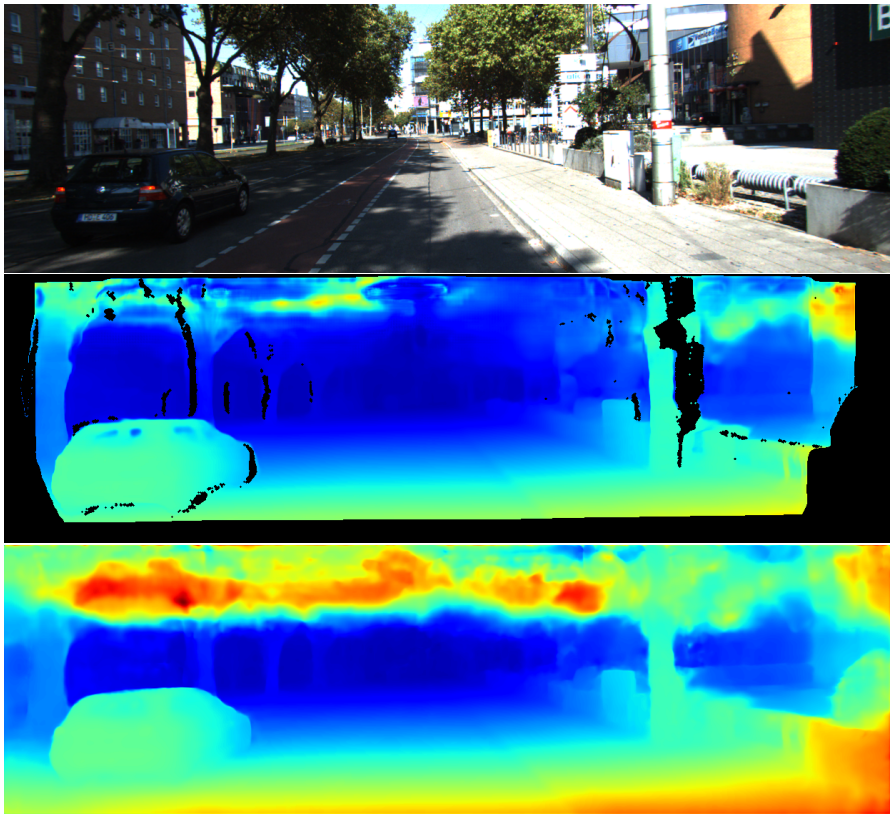


**Figure 6.5.:** *Visualization of a depth (virtual disparity) estimate of the validation split before and after interpolation with SSGP [SWUS21] (Section 5.2).*

Figure 6.3). The refined disparity map is termed $\mathrm{d}_{t,r}$.

Lastly, all separate results $(\mathbf{u}, \mathrm{d}_{t,r}, \mathrm{d}_{t,i})$ are combined to form dense scene flow (in image space).

### 6.2.3. Experiments and Results

In the experiments, the monocular combination approach is evaluated step-by-step – starting from the initial estimates until dense scene flow – and then compared to state-of-the-art on the KITTI scene flow benchmark [GLU12; MG15]. The auxiliary model for monocular depth estimation is trained with the KITTI depth data set, which is much larger. These two data sets are related by an intersection of 142 sequences of the labeled training sets. The remaining 58 sequences are defined as the validation split for the experiments. The full list of these sequences is available in the official development kit of the KITTI scene flow data set.

#### Ablation Study

In the ablation study, separate parts and intermediate results of the approach are evaluated individually. The results are presented in Table 6.3. In detail, the re-trained BTS [LHKS19] with disparity transformation is tested on the validation set and a small improvement over the officially provided pre-trained weights can be noticed. Further, the two considered estimators for optical flow, HD3 [YDY19] and VCN [YR19], are evaluated. It is important to note, that these networks are not re-trained and thus the validation split is entirely used during the training of both, HD3 and VCN. This is indicated by parentheses in Table 6.3. That said, the generalization of the models to unseen data is proper as it is shown by the results on the KITTI benchmark (cf. Table 6.4).

The important next step of the pipeline is the warping. The warped (virtual) disparities are also evaluated together with the sparse scene flow that is created by warping. Similar performance for both optical flow estimators can be observed, reaching densities of 81.9 % and 81.7 %. It is further evident that the warping introduces some artifacts which reduce the accuracy compared to the disparity maps being directly estimated in the respective reference frame. At the same time, the masking of occlusions and out-of-bounds motions also removes some outliers in the disparity at time $t$ and the optical flow. This reveals that HD3 has more difficulties handling occlusions compared to VCN. Overall, as seen in Section 6.1, the sparse scene flow obtained by warping is comparatively accurate, yet non-dense.

To recover full density, $d_{t,w}$ is interpolated with SSGP. Interestingly, the interpolation does not only fill in the gaps, but further improves the dense result over the sparse one. For that reason, dense refinement is finally applied to $d_t$ using SSGP. This results in a significant reduction of errors for *D1* and an even bigger improvement for the overall scene flow estimates. At the same time, the qualitative impressions in Figures 6.5 and A.3e reveal that the interpolation and the dense refinement with SSGP introduce artifacts in the sky regions

**Table 6.3.:** *Evaluation of different components and steps of MonoComb on the validation split of the KITTI scene flow training data. End-point error (EPE, [px]) and KITTI outlier error (KOE, [%]) are given. Numbers in parentheses indicate that the respective model is (partially) trained on the validation data.*

| Method | D1 | | D2 | | OF | | SF | | Density |
|---|---|---|---|---|---|---|---|---|---|
| | EPE | KOE | EPE | KOE | EPE | KOE | ΣEPE | KOE | |
| BTS [LHKS19] (original) | 3.60 | 24.51 | – | – | – | – | – | – | 100 % |
| BTS (re-trained) | 3.19 | 23.86 | – | – | – | – | – | – | 100 % |
| HD3 [YDY19] | – | – | – | – | (1.74) | (5.13) | – | – | 100 % |
| VCN [YR19] | – | – | – | – | (1.41) | (5.00) | – | – | 100 % |
| BTS + HD3 | 2.99 | 20.18 | 3.42 | 26.28 | (0.95) | (2.39) | 7.35 | 29.04 | 81.89 % |
| BTS + VCN | 2.98 | 20.15 | 3.36 | 25.44 | (0.76) | (2.70) | 7.10 | 28.07 | 81.70 % |
| BTS + HD3 + SSGP [SWUS21] (D2) | 3.19 | 23.86 | 3.21 | 24.01 | (1.74) | (5.13) | 8.14 | 35.08 | 100 % |
| BTS + VCN + SSGP (D2) | 3.19 | 23.86 | 3.24 | 24.46 | (1.41) | (5.00) | 7.84 | 35.22 | 100 % |
| BTS + HD3 + SSGP (D1+D2) | 2.76 | 20.30 | 3.21 | 24.01 | (1.74) | (5.13) | 7.70 | 29.50 | 100 % |
| BTS + VCN + SSGP (D1+D2) | 2.76 | 20.30 | 3.24 | 24.46 | (1.41) | (5.00) | 7.41 | 29.34 | 100 % |
| BTS + SSGP (D1) + OE [YR20] | 2.76 | 20.30 | (3.36) | (23.63) | (2.02) | (7.01 ) | 8.14 | 26.53 | 100 % |
| BTS + OE | 3.19 | 23.86 | (3.90) | (27.87) | (2.02) | (7.01 ) | 9.10 | 30.60 | 100 % |
| MonoDepth2 [GMFB19] + OE | 2.95 | 25.37 | (3.54) | (28.47) | (2.02) | (7.01) | 8.50 | 30.90 | 100 % |

**Table 6.4.:** *Evaluation results from the KITTI benchmark for all submitted monocular scene flow approaches. It is distinguished between supervised and purely unsupervised methods. Best (lowest) numbers in bold.*

| Method | D1 [%] | | | D2 [%] | | | OF [%] | | | SF [%] | | | Run |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | bg | fg | all | bg | fg | all | bg | fg | all | bg | fg | all | time |
| Mono-SF [BAM19] | **14.21** | 26.94 | **16.32** | **16.89** | 33.07 | **19.59** | 11.40 | 19.64 | 12.77 | **19.79** | 39.57 | **23.08** | 41 s |
| **MonoComb** | 17.89 | **21.16** | 18.44 | 22.34 | **25.85** | 22.93 | 5.84 | 8.67 | 6.31 | 27.06 | **33.55** | 28.14 | 0.58 s |
| MonoExpansion [YR20] | 24.85 | 27.90 | 25.36 | 27.69 | 31.59 | 28.34 | **5.83** | **8.66** | **6.30** | 29.82 | 36.67 | 30.96 | 0.25 s |
| Self-Mono-SF-ft [HR20] | 20.72 | 29.41 | 22.16 | 23.83 | 32.29 | 25.24 | 15.51 | 17.96 | 15.91 | 31.51 | 45.77 | 33.88 | **0.09 s** |
| Self-Mono-SF [HR20] | 31.22 | 48.04 | 34.02 | 34.89 | 43.59 | 36.34 | 23.26 | 24.93 | 23.54 | 46.68 | 63.82 | 49.54 | **0.09 s** |

(upper third of the image) where no supervision signal is available in the data set.

Lastly, numbers for the competing method OE [YR20] are presented. This method uses every fifth frame (starting with sequence 0) for validation and the rest for training. As a consequence, part of the validation set is used to train OE. The reference depth estimate for OE is computed with MonoDepth2 [GMFB19]. Numbers when OE is used in conjunction with BTS and the refined disparity estimate $d_{t,r}$ are also reported. This comparison is made to validate that the proposed monocular approach does not perform better only because of the better depth estimator (BTS over MonoDepth2), but because of the way corresponding depth values are correlated over time. This is validated by the relatively little improvement of scene flow outliers (*SF KOE*) compared to the outliers of *D1* in the last two rows of Table 6.3. The third last row validates that the contribution of dense refinement is also beneficial for other approaches.

### Comparison to State-of-the-Art

In this experiment, the dense monocular scene flow approach is compared to state-of-the-art in this field. This is done by submitting to the online KITTI scene flow benchmark. For this step, SSGP for interpolation and dense refinement is re-trained on all 200 sequences of the KITTI training split. This turned out especially useful since the validation split is comparatively large (> 25%). Results for all published monocular approaches and the proposed method are shown in Table 6.4. The results of the ablation study (Table 6.3) are transferred within reasonable deviation and some further improvements due to the additional training data.

The monocular combination approach (MonoComb) pushes state-of-art in various ways. It achieves the second best result overall and the best result among all methods with sub-second run time. Further, it achieves the lowest error rate for the important foreground regions (*fg*) of dynamic objects with a margin of more than 3 percentage points.

A qualitative comparison is provided in Figure A.3 in Appendix A where the result of the first test frame for all monocular methods is visualized along with the corresponding error maps. This particular sample reveals the major issues of each approach. For Mono-SF [BAM19] the biggest challenge is the correct estimation of dynamic objects. This is also reflected by the quantitative results in Table 6.4. The monocular version of OE [YR20] (MonoExpansion), depends a lot on the estimate of the initial depth/disparity. Further, since the relative change in depth is tightly coupled and estimated together with optical flow, the estimated second disparity suffers from the same limitations as optical flow, i.e. occlusions, visual perturbation by large geometric deformations over time, etc. The (fine-tuned) self-supervised approach [HR20] is mainly restricted by the lower level of details due to the reduced output resolution. The proposed approach predicts scene flow at full resolution, handles occlusion explicitly and solves this issue by interpolation, and provides top performance for dynamic objects. However, the error maps in Figure A.3e indicate that the absolute

scale of depth is not recovered sufficiently well. Further, the major limitation of MonoComb is the inconsistency of the prediction due to the separation. This results in a small overlap of correct and erroneous regions across the separate tasks, and ultimately in a high outlier rate in the scene flow metric. In fact in Table 6.4, MonoComb has a much larger margin between the highest outlier rate of *D1*, *D2*, or *OF* and *SF*, compared to e.g. OE [YR20].

### Run Time

One advantage of the combination approach is the modularity. As part of that, the overall run time is defined by the sum of the auxiliary run times as given by Table 6.5. In the given case this leads to an approximate average run time per frame of 0.58 seconds. This sums up over two runs of the single image depth estimator, one forward pass of the optical flow estimator, and two runs of SSGP for refinement and interpolation. The time for warping and combination of the separate results is neglectable small.

**Table 6.5.:** *Breakdown of the run time for MonoComb and its modules.*

| Module | Calls | Run time |
|---|---|---|
| HD3 [YDY19] / VCN [YR19] | 1 | 0.1 / 0.18 s |
| BTS [LHKS19] | 2 | 0.06 s |
| SSGP [SWUS21] | 2 | 0.14 s |
| Total | – | 0.5 / 0.58 s |

## 6.3. Summary

In summary, a straightforward approach to compute scene flow from auxiliary results is presented. Accuracy and run time only depend on the algorithms that are used to compute depth and optical flow and for the interpolation. Thus, scene flow estimation in real time is possible in theory. It is demonstrated that even the basic combination of optical flow and disparity leads to competitive results. The sparse and accurate results can be interpolated to a dense scene flow field with competitive performance. Improvements in stereo algorithms, optical flow estimation, and scene flow interpolation improve the combination approach as presented, immediately.

The sparse-to-dense recombination approach for scene flow estimation is further successfully transferred to the monocular camera setup. This is achieved by the latest success in single image depth estimation and robust sparse-to-dense interpolation. Together with state-of-the-art auxiliary estimators, the proposed concept achieves competitive results at reasonable speed. The biggest limitation of MonoComb (Section 6.2) is the inconsistency of the separate results. A proposition to overcome this in the future is to perform interpolation and dense refinement jointly, which might also introduce mutual advantages for both tasks. Additionally, the monocular camera setup for scene flow estimation

does not restrict the depth estimation to use a single image. It should be investigated whether the joint estimation of depth over time (e.g. two-view depth estimation) can improve the results further. In this relatively new discipline of monocular scene flow estimation, the monocular combination approach can be seen as a strong baseline for future developments.

Chapter 7

# End-to-End Scene Flow Estimation with Deep Neural Networks

*"Now these points of data make a beautiful line."*
— Jonathan Coulton, *Still Alive*

## Contents



This chapter is backed by the following publications:

Rohan Saxena, **René Schuster**, Oliver Wasenmüller, and Didier Stricker. "PWOC-3D: Deep Occlusion-Aware End-to-End Scene Flow Estimation." In: *Intelligent Vehicles Symposium (IV)*. 2019. **Oral**.

Rishav*, **René Schuster***, Ramy Battrawy, Oliver Wasenmüller, and Didier Stricker. "ResFPN: Residual Skip Connections in Multi-Resolution Feature Pyramid Networks for

Accurate Dense Pixel Matching." In: *International Conference on Pattern Recognition (ICPR)*. 2021. *Equal contribution. **Oral**.

**René Schuster**, Christian Unger, and Didier Stricker. "A Deep Temporal Fusion Framework for Scene Flow Using a Learnable Motion Model and Occlusions." In: *Winter Conference on Applications of Computer Vision (WACV)*. 2021.

---

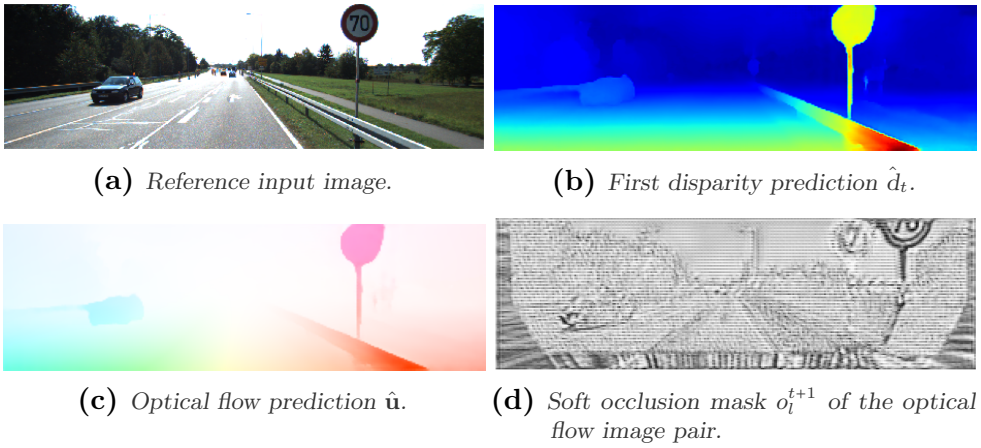
In the last few years, Deep Neural Networks (DNNs) have demonstrated increasing success at learning many computer vision tasks including dense matching problems such as optical flow and stereo matching. The inherent, massive parallelization on Graphics Processing Units (GPUs) yields run times close to real time in many cases. The first end-to-end dense matching network for optical flow has been introduced by FlowNet [DFIH+15]. At this time, the feasibility was proven, but the overall accuracy was lagging behind conventional approaches. With the introduction of FlowNet2 [IMSK+17], deep neural networks started to take over the field of optical flow estimation. This motivates to also investigate deep neural networks for scene flow estimation in more detail in this chapter. At the same time, an evaluation of DNNs against state-of-the-art is necessary to validate how limiting the lack of annotated data is and what can be done to overcome this issue.

Apart from presenting one of the first end-to-end trainable deep neural networks for scene flow estimation (Section 7.1), this chapter deals with the frequent motives of occlusion and feature representation. Section 7.2 introduces a universal improvement on previous feature extracting blocks for end-to-end networks that operate on multiple scales, comparable to the standalone feature representation of SDC in Section 5.1. As the final contribution of this thesis, a deep framework for generic multi-frame extension is presented in Section 7.3 that implicitly models the issue of occlusion.

## 7.1. Pyramids, Warping, Occlusion Estimation, and Cost Volume Correlation in 3D Scene Flow Estimation

The work presented here overcomes some drawbacks in terms of speed and accuracy by proposing Pyramid, Warping, Occlusions, and Cost Correlation for 3D Scene Flow (PWOC-3D) [SSWS19], a compact Convolutional Neural Network (CNN) architecture to predict scene flow from stereo image sequences in an end-to-end supervised setting. Further, large motion and occlusions are well-known problems in scene flow estimation. PWOC-3D employs specialized design decisions to explicitly model these challenges. In this regard, a novel self-supervised strategy to predict occlusions from images (learned without any labeled occlusion data) is proposed. Leveraging several such constructs, the network achieves competitive results on the KITTI benchmark and the challenging FlyingThings3D (FT3D) data set.

Solving the problem of scene flow estimation with a deep neural network also offers the advantages of omitting prior assumptions and speed over earlier



**(a)** *Reference input image.*

**(b)** *First disparity prediction* $\hat{d}_t$.

**(c)** *Optical flow prediction* $\hat{\mathbf{u}}$.

**(d)** *Soft occlusion mask* $o_l^{t+1}$ *of the optical flow image pair.*

**Figure 7.1.:** *Example predictions of the PWOC-3D network on the KITTI benchmark. PWOC-3D uses CNNs to predict scene flow in an efficient end-to-end fashion. The soft occlusion map (d) is predicted by the novel self-supervised occlusion reasoning mechanism, which is leveraged to improve scene flow estimates.*

approaches, both of which are critical for deployment in intelligent vehicles. Autonomous driving systems and embedded devices require real-time estimation of scene flow. Traditional approaches like the ones presented in Chapter 4 are iterative and can take between 1 to 50 minutes to process a set of images. In comparison, PWOC-3D can perform the same task with a single forward pass of the network in less than 0.2 seconds.

PWOC-3D also contains specialized constructs to handle specific challenges in a typical scene flow pipeline. Firstly, inspired by the work of PWC-Net [SYLK18] for optical flow, a coarse-to-fine estimation scheme is employed by using a spatial pyramid and warping image features at each pyramid level using the intermediate flow estimate from the previous level to handle large motion. Secondly, a novel self-supervised strategy to predict dense occlusion maps from images is proposed and used to improve scene flow estimates (cf. Figure 7.1). To the best of the author's knowledge, this is the first method to reason about occlusion using a single flow prediction and without any occlusion ground truth. Some previous methods [HR17; MHR18; WYYZ+18] require at least bidirectional flow (forward and backward preditions) to model occlusion, thus the network reduces the effort by half compared to these methods.

The PWOC-3D design demonstrates that embedding vision techniques, which leverage the underlying domain-knowledge of the problem and geometry of the scene, within differentiable CNNs produce better results than either approach is able to single-handedly achieve. The code base associated with PWOC-3D is publicly available[5].

[5] https://github.com/dfki-av/pwoc-3d

### 7.1.1. End-to-End Networks for Dense Matching

**End-to-end CNNs for Optical Flow.** The incorporation of a spatial pyramid and warping at different pyramid levels in an end-to-end CNN for optical flow estimation is first introduced in SPyNet [RB17]. PWC-Net [SYLK18] builds upon SPyNet's pipeline by replacing the latter's image pyramid with a feature pyramid. A cost volume is used to predict optical flow instead of plain features, and an additional network is used to refine predictions from the last pyramid level. PWOC-3D uses the PWC-Net architecture as a skeleton, but differs from it in several ways. Firstly, PWOC-3D reasons about the full 3D motion of objects rather than just 2D optical flow. This is made possible by several key design decisions: Four image pyramids (one for each image in the stereo sequence) are constructed, 1D (for disparity) and 2D (for optical flow) versions of the warping and cost volume operations are defined in the network. The 1D operations leverage the epipolar constraint for rectified stereo images to limit computation. Secondly, a Feature Pyramid Network (FPN) [LDGH+17] is employed to construct the feature pyramids instead of PWC-Net's generic CNN feature extractor (as illustrated in Figure 7.2), with significant improvement in results. Thirdly, PWOC-3D explicitly reasons about occlusion via a novel self-supervised method of predicting occlusion directly from images (without any occlusion ground truth), and exploits this understanding to improve scene flow predictions. The entire PWOC-3D pipeline is described in detail in Section 7.1.2.

**End-to-End CNNs for Scene Flow.** At the time of publication, there have been only a couple of other end-to-end CNN architectures published for scene flow. The first has been proposed alongside the FT3D data set [MIHF+16] primarily as a proof-of-concept of the utility of the data set. This network contains roughly three times the number of trainable parameters of FlowNet [DFIH+15]. In contrast, PWOC-3D outperforms it while being smaller than a single FlowNet model. The second end-to-end CNN is presented in [ISKB18]. This network uses three separate processing pipelines to predict optical flow, initial and final disparities, respectively. It is able to demonstrate competitive performance on the KITTI benchmark, though with the number of parameters of ten FlowNet models. PWOC-3D is a fast and compact network with 48 times fewer parameters and follows closely in terms of accuracy.

### 7.1.2. Network Architecture

The pipeline of PWOC-3D involves extracting a feature pyramid for each of the four images $I_t^l$, $I_t^r$, $I_{t+1}^l$, $I_{t+1}^r$. The features of $I_t^r$, $I_{t+1}^l$, $I_{t+1}^r$ at a particular pyramid level (except the top, i.e. lowest resolution) are warped towards the features of $I_t^l$ using the flow estimates from the upper pyramid level. Based on the warped features, occlusion maps are predicted for $I_t^r$, $I_{t+1}^l$, $I_{t+1}^r$. A cost volume is then constructed using the features of $I_t^l$ and each of the warped features of $I_t^r$, $I_{t+1}^l$, $I_{t+1}^r$ considering the predicted occlusions. Afterwards, a scene flow estimator is used to predict scene flow using these cost volumes.

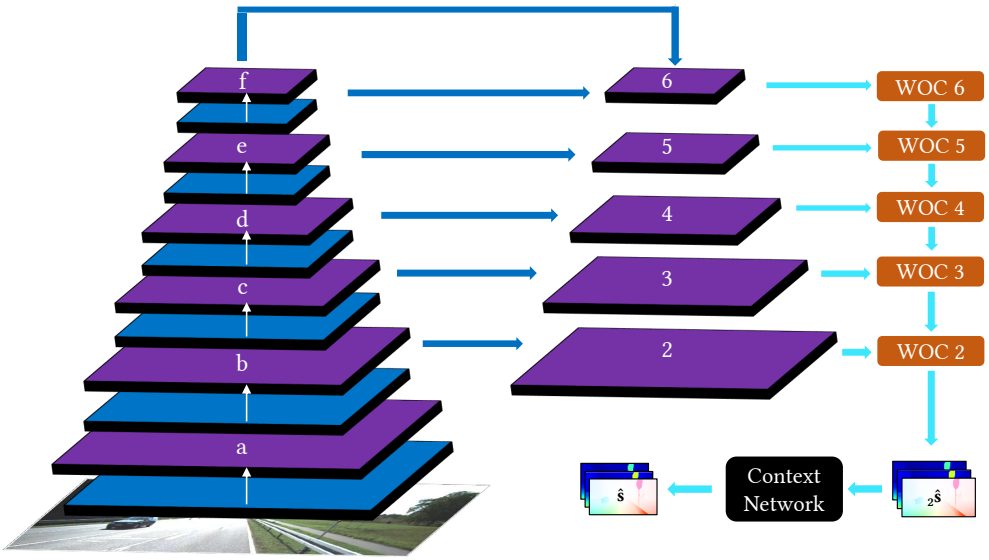



**Figure 7.2.:** *Visualization of the flow of information across pyramid levels in the entire PWOC-3D pipeline. PWC-Net [SYLK18] uses only the levels $b, c, d, e, f$ as a feature pyramid, while PWOC-3D uses the levels $2, 3, 4, 5, 6$. The orange boxes (WOC) represent warping, occlusion estimation, cost volume correlation, and scene flow prediction for one level of the pyramid and is detailed in Figure 7.3.*

Finally, a context network with dilated convolutions is used to refine the scene flow estimates. The complete overview of the end-to-end architecture is given in Figure 7.2. Figure 7.3 shows a detailed view of a particular pyramid level $l$.

**Feature Pyramids.** In PWC-Net [SYLK18], a simple feedforward strided CNN is used to construct feature pyramids for the input images. However, using the different feature maps of a generic CNN in this manner is not an optimal strategy. This is because the high-resolution feature maps from the first few layers of the network contain well-localized, but semantically weak features; while low-resolution maps from deeper layers contain processed and semantically strong features which are not well-localized with respect to the original image due to strided sub-sampling of the convolution operation [SWUS19].

FPN [LDGH+17] proposes to overcome this problem by incorporating additional connections in the network as shown in Figure 7.2. In addition to the backbone bottom-up pathway (which computes a feature hierarchy as in a standard CNN), top-down pathways and lateral residual connections are introduced. The top-down pathway produces higher resolved, semantically stronger feature maps, while the lateral skip connections (from lower layers in the pipeline) reinforce the localization of features with respect to the input images. Combined, this mechanism leads to the feature map at each pyramid level being well-localized and semantically strong.

Since the architecture uses a coarse-to-fine estimation approach, especially

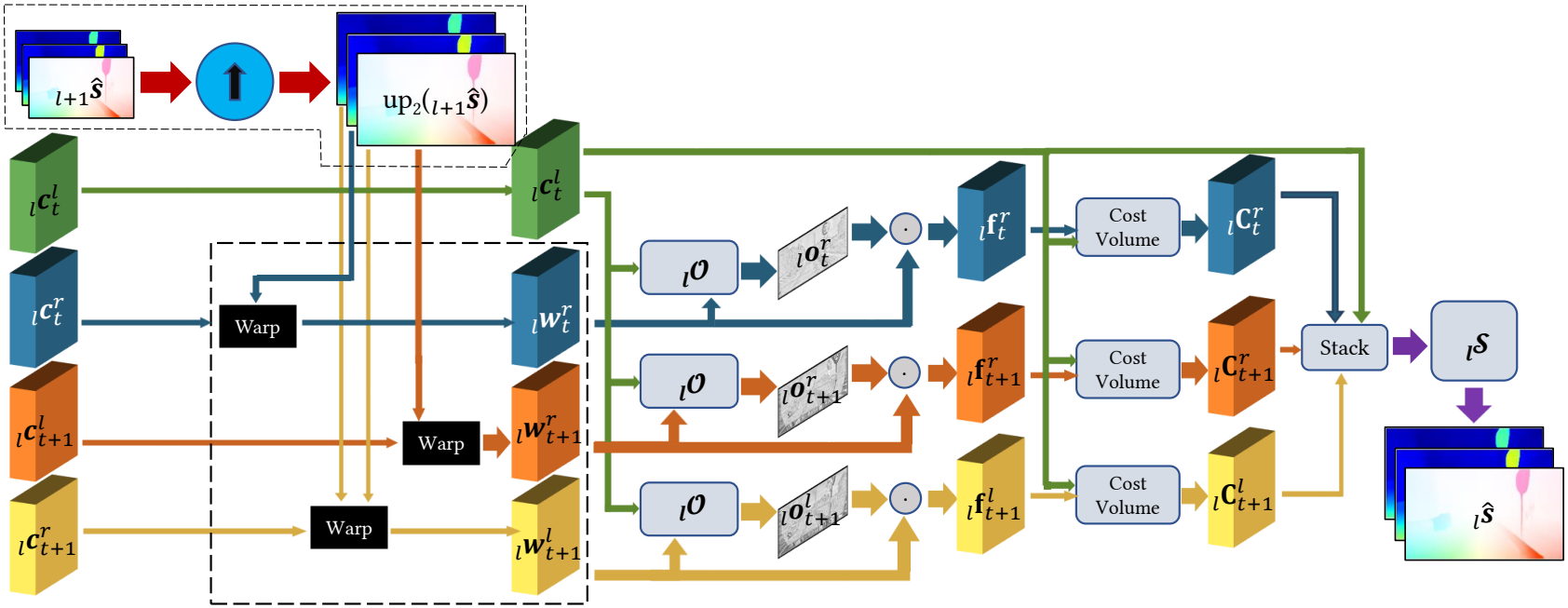



**Figure 7.3.:** *An overview of the inference pipeline of PWOC-3D at pyramid level $l$. The operations within the dashed boundary denote the warping operations, which are present at every pyramid level except the topmost.*

with predictions from further up in the pyramid being used in lower levels, the consistency of semantic strength and the spatial localization of the features across levels becomes particularly important. Thus, this work explores the role of FPN-like connections in the PWOC-3D pipeline.

For each of the four input images $I_t^l$, $I_t^r$, $I_{t+1}^l$, $I_{t+1}^r$, the same network (see Figure 7.2) is used to construct four six-layered feature pyramids (denoted as $\mathbf{c}_t^l$, $\mathbf{c}_t^r$, $\mathbf{c}_{t+1}^l$, $\mathbf{c}_{t+1}^r$ respectively), with each subsequent pyramid level having half the resolution (in each spatial dimension) of its predecessor, so that the sub-sampling factor at layer $l$ is $2^l$ for each dimension. The processing is started from the topmost level and continued in a coarse-to-fine estimation scheme until pyramid level 2. Consequently, PWOC-3D produces final scene flow estimates at 1/4th of the input resolution in each dimension. The final prediction is up-sampled using bilinear interpolation to obtain full-scale scene flow.

**Warping.** At every pyramid level $l$, the feature maps of $I_t^r$, $I_{t+1}^l$, $I_{t+1}^r$ (denoted as ${}_l\mathbf{c}_t^r$, ${}_l\mathbf{c}_{t+1}^l$, ${}_l\mathbf{c}_{t+1}^r$ respectively) are warped towards the reference image $I_t^l$. Let the scene flow estimate for pixel $\mathbf{p} = (x, y)^T$ at level $l$ be denoted as ${}_l\hat{\mathbf{s}}(\mathbf{p}) = ({}_l\hat{u}, {}_l\hat{v}, {}_l\hat{d}_0, {}_l\hat{d}_1)^T$, the images are warped as follows:

- ${}_l\mathbf{c}_t^r$ is warped towards $I_t^l$ using the disparity ${}_{l+1}\hat{d}_0$, a 1D warping:

$$ {}_l\mathbf{w}_t^r(\mathbf{p}) = {}_l\mathbf{c}_t^r\left(\left(x - \mathrm{up}_2\left({}_{l+1}\hat{d}_0\right)(\mathbf{p}), y\right)^\top\right), \tag{7.1} $$

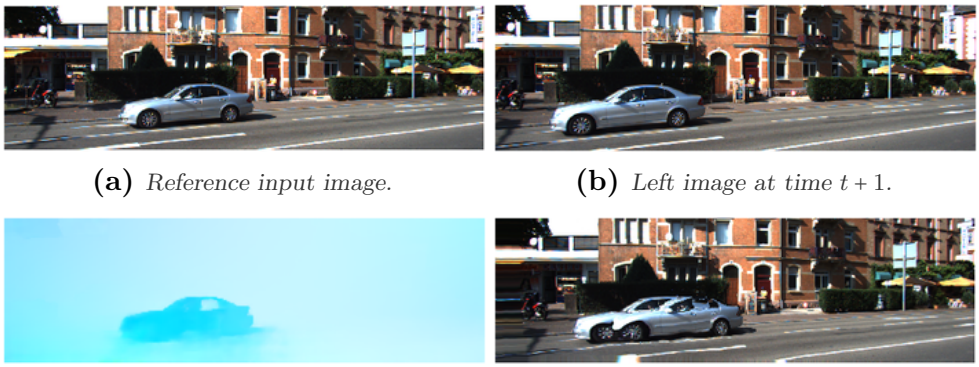
  whereas $\mathrm{up}_2({}_{l+1}\hat{d}_0)$ denotes the predicted disparity map from level $l+1$ which is up-sampled by a factor of 2 using bilinear interpolation.
- ${}_l\mathbf{c}_{t+1}^l$ is warped towards $I_t^l$ using optical flow ${}_{l+1}\hat{\mathbf{u}} = ({}_{l+1}\hat{\mathrm{u}}, {}_{l+1}\hat{\mathrm{v}})^T$, a 2D warping:

$$ {}_l\mathbf{w}_{t+1}^l(\mathbf{p}) = {}_l\mathbf{c}_{t+1}^r\left(\mathbf{p} + \mathrm{up}_2\left({}_{l+1}\hat{\mathbf{u}}\right)(\mathbf{p})\right). \tag{7.2} $$

- ${}_l\mathbf{c}_{t+1}^r$ is warped towards $I_t^l$ using optical flow ${}_{l+1}\hat{\mathbf{u}} = ({}_{l+1}\hat{\mathrm{u}}, {}_{l+1}\hat{\mathrm{v}})^T$ and disparity ${}_{l+1}\hat{\mathrm{d}}_1$, a modified 2D warping:

$$ \begin{aligned} &{}_l\mathbf{w}_{t+1}^r(\mathbf{p}) = \\ &{}_l\mathbf{c}_{t+1}^r\Big(\Big(x - \mathrm{up}_2\left({}_{l+1}\hat{\mathrm{d}}_1\right)(\mathbf{p}) + \mathrm{up}_2\left({}_{l+1}\hat{\mathrm{u}}\right)(\mathbf{p}), \\ &\quad y + \mathrm{up}_2\left({}_{l+1}\hat{\mathrm{v}}\right)(\mathbf{p})\Big)^\top\Big). \end{aligned} \tag{7.3} $$

**Occlusion Handling.** Occlusions are omnipresent in realistic, dynamic scenes and play an important role in the estimation of scene flow. Firstly, it leads to incorrect matching costs being computed since the object of interest is occluded from view. Secondly, the lack of information about the occluded area can throw off a naïve method because tracking the occluded pixels directly is impossible, and it must leverage other information to estimate this occluded motion. Thirdly, a considerable area of the reference image is occluded from view when it moves out of the camera's field of view due to motion of the camera itself. This ego-motion is an inherent characteristic of autonomous driving systems. Thus, failing to account for occlusion has significant drawbacks.



**(a)** *Reference input image.* **(b)** *Left image at time $t+1$.*

**(c)** *Predicted optical flow from $I_t^l$ to $I_{t+1}^l$.* **(d)** *$I_{t+1}^l$ warped towards $I_t^l$ using the predicted optical flow.*

**Figure 7.4.:** *The adverse effect of occlusion on the warping operation. In (d), there are two cars visible: The car on the right is the* true *car. The part of the road which is visible in $I_t^l$ and occluded by the car in $I_{t+1}^l$ is static, due to which the occluding area of the car is reproduced incorrectly from $I_t^l$. In other words, multiple flow vectors in (c) are pointing to the same target position in $I_{t+1}^l$.*

Occlusion also has an adverse effect on the (1D and 2D) warping operation, as illustrated in Figure 7.4.

MirrorFlow [HR17] predicts bidirectional flow using variational methods and uses it to warp both images towards each other. A forward-backward consistency check is imposed on these warps. Areas which do not pass this check are considered occluded. This leads to consistent occlusion maps in both directions. UnFlow [MHR18] uses a very similar formulation of the problem: Bidirectional flow estimation, forward-backward consistency check, occlusion estimation. The difference here is that FlowNet [DFIH+15] is used to predict flow instead of variational methods, and occluded areas are masked from contributing to the loss function. The authors of [WYYZ+18] use the same basic pipeline as UnFlow. They propose a different method of predicting occlusion maps based on warping a constant grid using the predicted flow. All previous methods predicted occlusion using bidirectional flow. In contrast, PWOC-3D estimates occlusions in three images $I_t^r$, $I_{t+1}^l$, $I_{t+1}^r$ without any labeled occlusion data while computing only the forward direction flow.

PWOC-3D employs a novel strategy to handle occlusion by learning an occlusion model conditioned on the input images. The occlusion mechanism is explained using $I_t^r$, but also applies in an analogous manner to the images $I_{t+1}^l$ and $I_{t+1}^r$. Specifically, occlusion in the image $I_t^r$ with respect to the reference image $I_t^l$ is modeled at each pyramid level $l$ as an occlusion map ${}_lo_t^r(\mathbf{p})$ where ${}_lo_t^r : {}_l\Omega \mapsto [0,1]$ and ${}_l\Omega$ denotes the image plane at scale $l$. Here 0 corresponds to occluded pixels while 1 corresponds to visible pixels. Since each pixel value is continuous, this is a soft occlusion map from which a hard occlusion map can be obtained by thresholding it appropriately to have binary values of 0 and 1.

This soft occlusion map is incorporated into PWOC-3D by multiplying it pixel-wise (by broadcasting along the channel dimension) with the corresponding warped features to result in masked features ${}_{l}\mathbf{f}_t^r$:

$$ {}_{l}\mathbf{f}_t^r(\mathbf{p}) = {}_{l}\mathbf{c}_t^r(\mathbf{p}) \cdot {}_{l}o_t^r(\mathbf{p}). \tag{7.4} $$

This has the effect of masking out occluded pixels from ${}_{l}\mathbf{w}_t^r$, leaving only the non-occluded areas. These masked, warped features are then used to construct the cost volume. Having the occluded pixels masked to zero, results in the correlation of the cost volume for these pixels to also be zero. In the cost volume correlation, a higher value means a higher degree of matching between two pixels. Thus, a cost of zero as computed for the occluded areas reflects no matching at all, which is semantically correct, since a pixel occluded in one image must not match with any other pixel in another image.

Due to the coarse-to-fine estimation approach adopted by PWOC-3D, the occlusion model is also a multi-scale mechanism. A separate occlusion map is predicted at each pyramid level $l$ for each of the warped image features (${}_{l}\mathbf{w}_t^r$, ${}_{l}\mathbf{w}_{t+1}^l$, ${}_{l}\mathbf{w}_{t+1}^r$), which masks occluded areas before computing the respective cost volume at this level.

**Learning Occlusions.** For predicting occlusion, a separate sub-network ${}_{l}\mathcal{O}$ at each scale $l$ is established, which maps the depth-wise stacked feature maps ${}_{l}\mathbf{c}_t^l$ and ${}_{l}\mathbf{w}_t^r$ to the soft occlusion map ${}_{l}o_t^r$. These stacked feature maps are used as input to the network because they provide sufficient information to predict occlusion: Regions without occlusion have similar features in the feature map ${}_{l}\mathbf{c}_t^l$ and the warped features ${}_{l}\mathbf{w}_t^R$; whereas pixels which are visible in ${}_{l}\mathbf{c}_t^l$ but occluded in ${}_{l}\mathbf{c}_t^r$ have dissimilarities in the features in ${}_{l}\mathbf{w}_t^r$, which can enable the network to predict such pixels as occluded.

The design of the ${}_{l}\mathcal{O}$ network consists of six layers of convolution, all of which employ a kernel of size $3 \times 3$, a stride of 1, and a padding of 1. The channel dimensions of the occlusion estimator network in each layer are 128, 96, 64, 32, 16 and 1 respectively. The last layer uses sigmoid as an activation function to ensure that the occlusion predictions are in the range $[0, 1]$. All other layers use the LeakyReLU activation function [MHN13].

This sub-network is inserted in the PWOC-3D pipeline after the warping operation and before the cost volume construction stage at each pyramid level. After warping the three feature maps towards the reference view, the reference features are stacked with each of the warped feature maps as $[{}_{l}\mathbf{c}_t^l, {}_{l}\mathbf{w}_t^r]$, $[{}_{l}\mathbf{c}_t^l, {}_{l}\mathbf{w}_{t+1}^l]$, $[{}_{l}\mathbf{c}_t^l, {}_{l}\mathbf{w}_{t+1}^r]$ and sequentially fed as input to the occlusion estimator network ${}_{l}\mathcal{O}$ of this corresponding pyramid level to obtain the occlusion maps ${}_{l}o_t^r$, ${}_{l}o_{t+1}^l$, ${}_{l}o_{t+1}^r$ respectively. Since the estimation of occlusion for each of the three images requires to learn the same underlying task (learning to match features in the stacked feature maps), a single occlusion estimator network ${}_{l}\mathcal{O}$ is used at each pyramid level to predict occlusions for each of the three images separately.

To maintain consistency of occlusion predictions across scales, every occlusion network ${}_{l}\mathcal{O}$ (except the network at the highest pyramid level) receives

an additional input from the network ${}_{l+1}\mathcal{O}$ of the next upper pyramid level. This additional input are the output features ${}_{l+1}\mathbf{g}$ of the penultimate convolutional layer of ${}_{l+1}\mathcal{O}$ and the corresponding predicted occlusion map ${}_{l+1}o$, i.e. $({}_{l+1}\mathbf{g}_t^r,\ {}_{l+1}o_t^r)$, $({}_{l+1}\mathbf{g}_{t+1}^l,\ {}_{l+1}o_{t+1}^l)$ and $({}_{l+1}\mathbf{g}_{t+1}^r,\ {}_{l+1}o_{t+1}^r)$ for the prediction of ${}_{l}o_t^r$, ${}_{l}o_{t+1}^l$ and ${}_{l}o_{t+1}^r$.

The occlusion estimator networks can learn to predict the occlusion weights based on the degree of similarity between the reference image features and the warped features. These weights are used to mask the incorrect matching costs in the cost volume, which results in more robust scene flow estimates. Thus, the network supervises itself while learning to estimate occlusions, with the goal of improving scene flow estimates (minimizing the error on scene flow predictions) without any labeled occlusion data. Note that the training of PWOC-3D requires ground truth scene flow data; only the estimation of the occlusion maps is self-supervised.

**Cost Volume.** Three cost volumes are computed using the reference image features $\mathbf{c}_t^l$ and the masked warped features ${}_{l}\mathbf{f}_t^r$, ${}_{l}\mathbf{f}_{t+1}^l$ and ${}_{l}\mathbf{f}_{t+1}^r$. In contrast, PWC-Net computed a cost volume using simply the warped features ${}_{l}\mathbf{w}_{t+1}^l$, which made its predictions susceptible to problems caused by occlusions.

Only a partial cost volume is constructed by limiting the search range to a maximum displacement of $d_{\max}$ pixels around each pixel. For the 1D cost volume operation (between $({}_{l}\mathbf{c}_t^l,\ {}_{l}\mathbf{f}_t^r)$), matches are searched for only in the horizontal dimension along the epipolar line, while for the 2D cost volume (between $({}_{l}\mathbf{c}_t^l,\ {}_{l}\mathbf{f}_{t+1}^l)$ and $({}_{l}\mathbf{c}_t^l,\ {}_{l}\mathbf{f}_{t+1}^r)$) the search is performed in 2D space. Then, the resulting cost volumes are organized as 3D arrays of dimension $H \times W \times C$ and $H \times W \times C^2$ for the 1D and 2D cost volumes, with $H$ and $W$ being the height and width of the feature maps, and $C = 2d_{\max} + 1$.

The matching cost in the cost volume is computed as the correlation between the feature vectors of two pixels. Consider ${}_{l}\mathbf{c}_t^l,\ {}_{l}\mathbf{f}_t^r : {}_{l}\Omega \mapsto I\!R^c$, where $\Omega$ is the image domain of the respective scale and $c$ is the number of channels of the feature maps. Then the correlation between two patches centered at pixels $\mathbf{p}$ and $\mathbf{p}'$ is computed as a vector of dimensionality $C^2$ where each individual pixel cost is given by:

$$ {}_{l}\mathbf{C}_t^r(\mathbf{p}, \mathbf{p}'|\mathbf{q}) = \frac{({}_{l}\mathbf{c}_t^l(\mathbf{p}))^\top {}_{l}\mathbf{f}_t^r(\mathbf{p}' + \mathbf{q})}{c}, \tag{7.5} $$

whereas $\mathbf{q} \in \{(\Delta x, \Delta y)^\top : \Delta x, \Delta y \in [-d_{\max}, d_{\max}]\}$.

**Scene Flow Prediction.** At every pyramid level $l$, a CNN ${}_{l}\mathcal{S}$ is trained to estimate scene flow using the masked cost volume described above. The primary input to this network consists of the three warped and masked cost volumes corresponding to ${}_{l}\mathbf{f}_t^r$, ${}_{l}\mathbf{f}_{t+1}^l$ and ${}_{l}\mathbf{f}_{t+1}^r$ stacked one over the other along the correlation dimension. The architecture of this network is similar to the occlusion estimator network: It consists of six layers of convolution filters with a kernel size of $3 \times 3$, a stride of 1, and a padding of 1. The channel dimensions of each layer are 128, 128, 96, 64, 32 and 4 respectively. Each layer again

employs the LeakyReLU activation, except the last layer which does not use any activation function to facilitate the prediction of continuous and negative scene flow values.

Similar to the occlusion estimator network, additional input is passed to ${}_l\mathcal{S}$ (at each pyramid level $l$ except the top), the features ${}_{l+1}\mathbf{h}$ of the penultimate convolutional layer of the network ${}_{l+1}\mathcal{S}$ (of the upper pyramid level) as well as its corresponding scene flow prediction ${}_{l+1}\hat{\mathbf{s}}$. This maintains consistency across the pyramid levels and allows the entire framework to perform multi-scale reasoning.

**Context Network.** As in PWC-Net [SYLK18], PWOC-3D employs a CNN with dilated convolutional layers to refine the prediction of the final pyramid level. The input to this network comprises the flow estimate ${}_2\hat{\mathbf{s}}$ and the last feature map ${}_2\mathbf{h}$ of the scene flow estimator ${}_2\mathcal{S}$ of the lowest pyramid level. This context network consists of seven convolutional layers with $3 \times 3$ kernels and a stride and padding of 1. The number of the filters at each layer are 128, 128, 128, 96, 64, 32, and 4. The dilation parameters used at each layer are 1, 2, 4, 8, 16, 1, and 1 respectively. All layers use the LeakyReLU activation, except the last layer which does not employ any activation.

This network outputs a residual flow ${}_2\Delta\hat{\mathbf{s}}$ which is added to the scene flow prediction ${}_2\hat{\mathbf{s}}$ to obtain the final prediction $\hat{\mathbf{s}}(\mathbf{p}) = {}_2\hat{\mathbf{s}}(\mathbf{p}) + {}_2\Delta\mathbf{s}(\mathbf{p})$.

**Loss Function.** To start the coarse-to-fine estimation scheme of PWOC-3D, the prediction is initialized with zeros at a hypothetical level 7 ${}_7\hat{\mathbf{s}}$. This has the effect that the features at the topmost level ($l = 6$) of the pyramid are not warped at all. Thus, the cost volume and occlusions are computed directly using the original feature maps. The rest of the pipeline progresses as described in the previous sections.

A multi-scale weighted loss function with intermediate supervision is employed, which penalizes errors at each level of the pyramid. Let $\Theta$ denote the set of all trainable parameters in the entire network and let ${}_l\mathbf{s}$ be the ground truth scene flow field sub-sampled to the resolution of pyramid level $l$, leading to the loss function

$$\begin{aligned}\mathcal{L}(\Theta) = &\underbrace{\sum_{l=3}^{6} \alpha_l \sum_{\mathbf{p} \in {}_l\Omega_{GT}} |{}_l\hat{\mathbf{s}}(\mathbf{p}) - {}_l\mathbf{s}(\mathbf{p})|_2}_{\text{pyramid levels except the last}} \\ &+ \underbrace{\alpha_2 \sum_{\mathbf{p} \in {}_2\Omega_{GT}} |\hat{\mathbf{s}}(\mathbf{p}) - {}_2\mathbf{s}(\mathbf{p})|_2}_{\text{bottom pyramid level 2}},\end{aligned} \tag{7.6}$$

in which $|\cdot|_2$ denotes the L2-norm of a vector.

This enables the entire PWOC-3D model: The feature pyramid network, the occlusion mechanism with its occlusion estimator networks at different pyramid levels, the scene flow estimator networks at each pyramid level, and the context network to be trained in an end-to-end manner.

### 7.1.3. Experiments and Results

**Training Details.** The primary focus of this work is to estimate scene flow for automotive applications, therefore the KITTI data set [MG15] is a natural choice. However, KITTI provides only 200 training sequences (with sparse ground truth only), which is not sufficient to train a deep neural network on such a complex task. To overcome this problem, PWOC-3D is first trained on the synthetic (but large) FT3D data set [MIHF+16], and then fine-tuned on KITTI. This transfer learning approach helps avoiding the network from being overfit on KITTI. PWOC-3D is trained for 760 epochs on FT3D and for 125 epochs on KITTI. The photometric data augmentation strategy of FlowNet [DFIH+15] is used, combined with random vertical flipping (the latter only for FT3D, and not for KITTI). Since FT3D also provides bidirectional scene flow annotations, these are utilized by random temporal flipping of the training sample from $(I_t^l, I_t^r, I_{t+1}^l, I_{t+1}^r)$ to $(I_{t+1}^l, I_{t+1}^r, I_t^l, I_t^r)$. Geometric transformations such as rotation, translation, etc. are not used, since they harm the epipolar constraint for disparity estimation across the rectified stereo pairs.

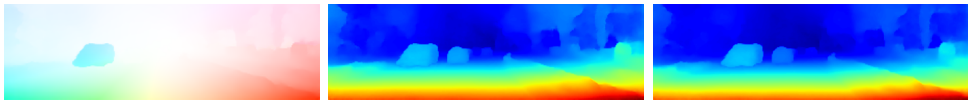
The hyperparameter $d_{\max}$ in the cost volume layer is set to 4. The weights used in the loss function $\alpha_2, \alpha_3, \ldots, \alpha_6$ are 0.32, 0.08, 0.02, 0.01 and 0.005 respectively as in [SYLK18]. Further, the ground truth scene flow is down-scaled by factor 20 as in [DFIH+15], and is down-sampled to different resolutions to compute the training signal at different scales. During inference, all scene flow predictions are made at the full input resolution by up-sampling $\hat{\mathbf{s}}$. The Adam optimizer [KB15] is used to train PWOC-3D with the default setting of hyperparameters as recommended. A constant learning rate of $\lambda = 10^{-4}$ is used.

**Quantitative Analysis.** As evident from Table 7.1, the PWOC-3D model with the occlusion mechanism and the improved feature pyramid is the best performing network among all the variants. This is due to its well-localized and semantically strong features, and its ability to mask the incorrect matching costs from the cost volume, thus preventing them from having adverse effects on the reasoning of the network. Table 7.1 also shows the comparison to two other end-to-end scene flow networks. PWOC-3D outperforms [MIHF+16] on FT3D in terms of end-point error and is 48 times smaller than the network from [ISKB18] (PWOC-3D has only ~8 million trainable weights).

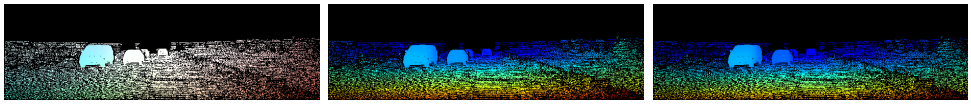
As depicted in Figure 7.5, the occlusion maps contain clear signs of the masking effect. Specifically, the occlusion map for $\mathrm{d}_0$ shown in Figure 7.5d contains areas occluded only along the horizontal epipolar line, while the maps for optical flow and $\mathrm{d}_1$ respectively model the occlusion caused by the ego-motion of the camera in addition to the occlusion arising because of motion. This demonstrates the significant impact of the occlusion mechanism in autonomous driving scenarios.

The network with the FPN shows improvement in performance over the network without. The end-point error of all the networks on the test split of FT3D [MIHF+16] is higher than that on the validation split as visible in Table 7.1. This is because in the test set, FT3D contains a set of objects and

**(a)** *The four input images $I_t^l$, $I_t^r$, $I_{t+1}^l$, $I_{t+1}^r$.*

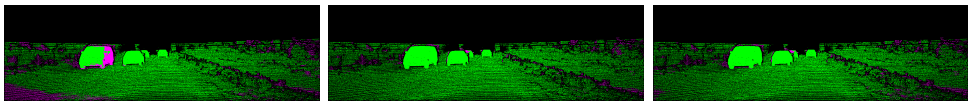


**(b)** *Scene flow prediction $\hat{\mathbf{s}}$ visualized by optical flow $\hat{\mathbf{u}}$, disparity $\hat{d}_0$, and disparity $\hat{d}_1$.*



**(c)** *Sparse scene flow ground truth $\mathbf{s}$.*

**(d)** *Occlusion masks for optical flow $\mathbf{o}_{t+1}^l$, first disparity $\mathbf{o}_t^r$, and second disparity $\mathbf{o}_{t+1}^r$.*



**(e)** *Binary error maps for optical flow and the two disparity estimates.*

**Figure 7.5.:** *Visualization of predictions of PWOC-3D with occlusion masks and ground truth on a validation sample from KITTI. In the error maps (e), pixels which contribute to the KITTI outlier error (KOE) are colored magenta, while those that do not are colored green.*

**Table 7.1.:** *Experimental results of PWOC-3D. End-point error (EPE [px]) and KITTI outlier error (KOE [%]) are shown on training, validation, and test set for KITTI and FlyingThings3D. Different components of the contribution are evaluated and compared to end-to-end scene flow networks from previous work.*

| | **FlyingThings3D** | | | | | | **KITTI** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Training** | | **Validation** | | **Testing** | | **Training** | | **Validation** | | **Testing** | |
| **Architecture** | EPE | KOE | EPE | KOE | EPE | KOE | EPE | KOE | EPE | KOE | EPE | KOE |
| PWOC-3D (*basic*) | 8.25 | 25.15 | 9.79 | 25.01 | 23.38 | 26.01 | 1.97 | 6.09 | 3.71 | 13.6 | – | – |
| PWOC-3D + FPN | 6.17 | 19.89 | 8.40 | 20.35 | **21.86** | 21.22 | **1.76** | **5.32** | 3.39 | 13.97 | – | – |
| PWOC-3D + FPN + Occ | **5.86** | **18.30** | **8.06** | **18.93** | 22.01 | **19.90** | 1.85 | 5.69 | **3.22** | **12.55** | – | **15.69** |
| SceneFlowNet [MIHF+16] | – | – | 11.24 | – | – | – | – | – | – | – | – | – |
| Occ-SceneFlow [ISKB18] | – | – | – | – | – | – | – | – | – | – | – | 11.34 |

**Table 7.2.:** *A snapshot of the KITTI scene flow benchmark's leaderboard at the time of publication. PWOC-3D is the most efficient in terms of run time.*

| **Method** | **D1-bg** | **D1-fg** | **D1-all** | **D2-bg** | **D2-fg** | **D2-all** | **Fl-bg** | **Fl-fg** | **Fl-all** | **SF-bg** | **SF-fg** | **SF-all** | **Run time** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ISF [BJMA+17] | 4.12 | **6.17** | 4.46 | **4.88** | **11.34** | **5.95** | 5.40 | **10.29** | **6.22** | **6.58** | **15.63** | **8.08** | 600 s |
| PRSM [VSR15] | **3.02** | 10.52 | **4.27** | 5.13 | 15.11 | 6.79 | **5.33** | 13.40 | 6.68 | 6.61 | 20.79 | 8.97 | 300 s |
| OSF+TC [NŠ17] | 4.11 | 9.64 | 5.03 | 5.18 | 15.12 | 6.84 | 5.76 | 13.31 | 7.02 | 7.08 | 20.03 | 9.23 | 3000 s |
| OSF 2018 [MHG18] | 4.11 | 11.12 | 5.28 | 5.01 | 17.28 | 7.06 | 5.38 | 17.61 | 7.41 | 6.68 | 24.59 | 9.66 | 390 s |
| SSF [RSKS17] | 3.55 | 8.75 | 4.42 | 4.94 | 17.48 | 7.02 | 5.63 | 14.71 | 7.14 | 7.18 | 24.58 | 10.07 | 300 s |
| OSF [MG15] | 4.54 | 12.03 | 5.79 | 5.45 | 19.41 | 7.77 | 5.62 | 18.92 | 7.83 | 7.01 | 26.34 | 10.23 | 3000 s |
| FSF+MS [TSS17] | 5.72 | 11.84 | 6.74 | 7.57 | 21.28 | 9.85 | 8.48 | 25.43 | 11.30 | 11.17 | 33.91 | 14.96 | 2.7 s |
| **PWOC-3D** | 4.19 | 9.82 | 5.13 | 7.21 | 14.73 | 8.46 | 12.40 | 15.78 | 12.96 | 14.30 | 22.66 | 15.69 | **0.13 s** |
| CSF [LBAL+16] | 4.57 | 13.04 | 5.98 | 7.92 | 20.76 | 10.06 | 10.40 | 25.78 | 12.96 | 12.21 | 33.21 | 15.71 | 80 s |
| SFF [SWKB+18] | 5.12 | 13.83 | 6.57 | 8.47 | 21.83 | 10.69 | 10.58 | 24.41 | 12.88 | 12.48 | 32.28 | 15.78 | 65 s |
| PR-Sceneflow [VSR13] | 4.74 | 13.74 | 6.24 | 11.14 | 20.47 | 12.69 | 11.73 | 24.33 | 13.83 | 13.49 | 31.22 | 16.44 | 150 s |

backgrounds which are disjoint from the training set. Thus, the samples are considerably different from those that the networks are trained on.

Another interesting result is that among all variants in Table 7.1, the difference between the validation and training errors is the lowest for PWOC-3D with the feature pyramid connections and the occlusion mechanism, particularly on KITTI where the training data is very limited. Thus, the occlusion reasoning scheme also helps to reduce overfitting.

Table 7.2 shows the ranking of the proposed method among the top ten published methods on the KITTI scene flow benchmark's leaderboard at the time of publication. As visible, PWOC-3D has a significantly lower run time (0.13 s per frame on a GeForce GTX 1080 Ti) than all other methods, thus making it suitable for real-time applications. Among the listed methods, PWOC-3D is the only approach with this property. Further, the approach is especially accurate in the important foreground regions of moving objects. In summary, PWOC-3D has a unique mixture of characteristics with competitive accuracy, small network size, and low run time.

## 7.2. Residual Skip Connections in Multi-Resolution Feature Pyramid Networks for Accurate Dense Pixel Matching

Dense pixel matching is the underlying problem of scene flow estimation. In recent years, especially CNN-based approaches, which are trained end-to-end, have achieved remarkable results for dense pixel matching [CC18; HTC18; SSWS19; SYLK18]. Within this category of algorithms, the feature representation turns out to be an essential factor for accurate matching [BVS17]. Section 5.1 demonstrates that CNNs are capable of learning a suitable representation of pixels for dense matching tasks. The previous Section 7.1 further shows, that FPNs are also a suitable feature extractor for dense matching on multiple scales. The representation must be as characteristic as possible in order to be distinguishable. In addition, it must be as localizable as possible to allow for accurate matching and avoid small displacement mismatches. In state-of-the-art, FPNs [LDGH+17] seem to fulfill these properties best. FPNs are originally proposed in the field of object detection, for which the localization is completely sufficient. FPN generates well-localized and semantically strong features at multiple scales. However, the generic FPN does not utilize its full potential, due to its reasonable but limited localization accuracy. That is why Residual Feature Pyramid Network (ResFPN) [RSBW+21] presents a multi-resolution feature pyramid network with multiple residual skip connections, where at any scale, the information from higher resolution maps are leveraged for stronger and better localized features. This is supposed to reintroduce details for better localization in the final feature representation. Further, the residual skip connections can reduce the length of gradient paths during back-propagation to improve convergence [ZDMD+18]. ResFPN is reviewed in a comprehensive ablation study by validating each individual de-

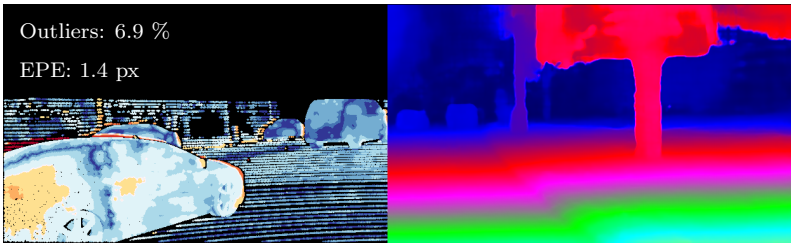
sign decision in detail. In addition, ResFPN is put into application for the dense pixel matching tasks of optical flow, scene flow and disparity estimation. For these experiments, state-of-the-art algorithms are utilized and nothing but the feature description is changed. The superior accuracy of ResFPN is confirmed across different matching algorithms as well as data sets, such as KITTI [MG15], Sintel [BWSB12] and FT3D [MIHF+16].

### 7.2.1. Related Concepts in the Literature

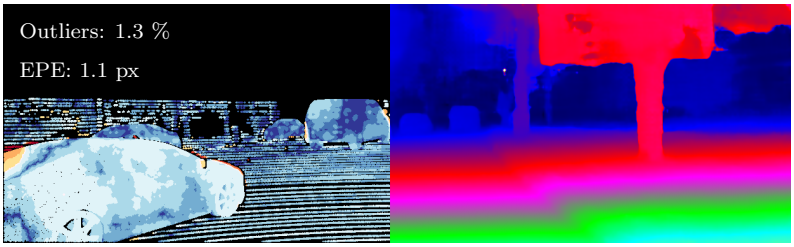
**Representations and Image Pyramids.** Feature maps (i.e. dense descriptors) are the basic cues for many computer vision tasks. A large number of methods show that a proper design of feature maps improves the results especially for dense pixel-wise matching in terms of geometric reconstruction and motion estimation. Many approaches employ handcrafted designs like SIFT [Low99], HOG [DT05] or DAISY [TLF09] features using image pyramid structure for seeking dense motion matches [BTS15; HSL16; XJM11] or for scene flow estimation [SWKB+18]. Pyramid feature representations use information from multiple scales for more improvement in terms of estimating correspondences. However, the advances of CNNs improve the robustness of feature maps against ill-conditioned environments, light or geometric changes compared to conventional solutions. In this context, many approaches aim to learn features [CGSC16; SWUS19] for dense matching. These methods replace the conventional descriptors, but they are not proven in end-to-end networks for dense matching. In contrast, ResFPN is a flexible, modular network that can be plugged in as feature backbone for end-to-end matching networks.

**End-to-End Solutions using Feature Pyramids.** Early end-to-end learning solutions yield impressive results based on encoder-decoder architectures, e.g. FlowNet [DFIH+15; IMSK+17] for optical flow estimation. DispNet [MIHF+16] extends the idea of FlowNet to disparity and scene flow estimation. The main idea of the encoder-decoder network is to aggregate the information from coarse-to-fine predictions, which is useful for large displacement predictions. However, it is a memory consuming approach and its computation is inefficient. SPyNet [RB17] is a lightweight model that aggregates information with a spatial pyramid network. Large motions can be handled with this approach. Compared to FlowNet, it is faster and yields better accuracy. PWC-Net [SYLK18] and LiteFlowNet [HTC18] add warping and cost volume layers to the pyramid feature extractor, which improves dense optical flow accuracy. PSMNet [CC18] uses a spatial pyramid pooling module to enlarge the receptive field of feature maps for stereo matching. Instead of using a generic CNN as feature extractor in PWC-Net [SYLK18], the previously presented PWOC-3D [SSWS19] (Section 7.1) employs the FPN architecture [LDGH+17] and utilizes these features for scene flow estimation with stereo images. ResFPN contributes to many kinds of deep end-to-end networks – including PWOC-3D – by improving the feature representation on multiple scales.

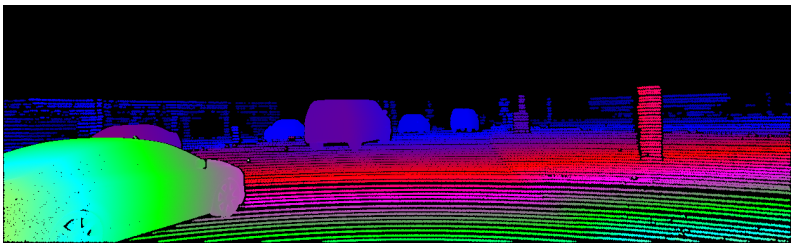
**(a)** *Reference image.*




**(b)** *Disparity result of PSMNet [CC18]*




**(c)** *Improved result with* ***ResFPN*** *[RSBW+21].*



**(d)** *Ground truth disparity.*

**Figure 7.6.:** *ResFPN is a deep architecture to compute feature representations for dense matching. ResFPN applied together with state-of-the-art matching networks like PSMNet [CC18] preserves details better and is more robust under challenging conditions.*

**Connecting Layers in Deep Neural Networks.** Traditional CNN architectures establish strictly sequential connections between layers [KSH12; LBBH+98; SZ15]. Recently, more involved connections have been proposed. DenseNet [HLVW17] uses connections in a feedforward fashion so that for each layer the feature maps of all preceding layers are used as input to strengthen the feature propagation. ResNet [HZRS16] and InceptionNet [SIVA17; SLJS+15] aim to improve deep networks through parallel shortcut connections.

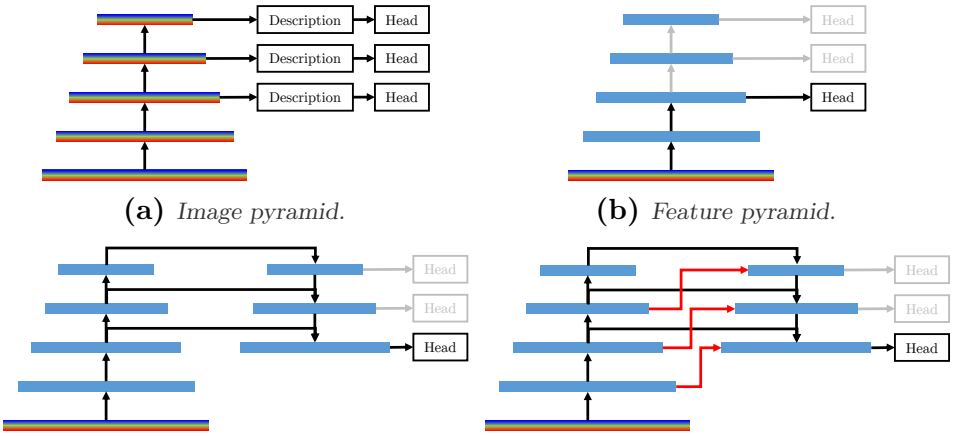
Among modern architectures, FPN [LDGH+17] leverages the concept of lateral connections for multi-level predictions based on features of multiple scales. Similar to the U-Net architecture [RFB15], it fuses feature maps between the same levels of top-down and bottom-up paths using element-wise addition. Moreover, TDM [SSMG16] changes the lateral connections to convolutional layers and channel-wise concatenation, which makes it computationally inefficient. Reverse Densely Connected Feature Pyramid Network [XWLZ+18] proposes to add reverse dense connections for the top-down module (decoder). Similarly, (A)RDFPN [ZLZC+19; ZLZZ+19] add dilated residual connections to the top-down stream of FPNs. The previous feature modules are presented in the context of object detection. Recently, HRNet [SXLW19] has used multi-resolution feature maps to improve localization in the estimation of human poses.

Apart from the aforementioned applications, ResFPN uses the advantages of pyramidal networks to extract dense feature maps for dense matching tasks in terms of stereo matching, optical flow, and scene flow estimation. Not only connections between similar levels of feature maps across bottom-up and top-down parts are utilized like in FPN, but the spatial accuracy is further enhanced by adding new connections across high resolution feature maps of the bottom-up part and feature maps in the top-down part as shown in Figure 7.7d.

### 7.2.2. Multiple Residual Skip Connections from Higher Resolutions

ResFPN is a generic concept that can be applied in many different applications for different tasks. The general idea is to increase the number of lateral skip connections between encoder and decoder in feature pyramid networks in order to improve the spatial accuracy while maintaining high-level feature representations.

The work continues with the logical extension of regular lateral skip connections to further improve localization and feature abstraction in feature pyramid networks [LDGH+17]. The reasoning is that additional connections from higher resolved levels of the encoder can benefit the final feature description (cf. Figure 7.7). Further, more densely connected networks are assumed to have a better flow of gradients during training [HLVW17; HZRS16] which improves convergence properties. Most recently, pyramidal feature extractors have also been shown to be more robust to adversarial attacks [RJGB19]. Moreover, the idea of ResFPN is independent of hyper-parameters of the pyramid like the number of levels, or the scale factor. It is applicable together with any building blocks for down-/up-sampling, like Residual [HZRS16], Dense [HLVW17], or Inception [SIVA17; SLJS+15] units. The idea of additional residual skip




**(a)** *Image pyramid.* **(b)** *Feature pyramid.*

**(c)** *Feature Pyramid Network [LDGH+17].* **(d)** *Exemplary structure of* ***ResFPN****.*

**Figure 7.7.:** *Feature computation with different types of pyramids. (a) A simple image pyramid is used together with heuristic descriptors for multi-scale predictions. (b) Feature pyramids successively compress and encode the input image for multi-scale predictions. (c) Feature Pyramid Networks traverse the entire encoder and decode the representation until the required scale is reached. (d) Additional feature encodings of higher resolutions are combined during up-sampling in ResFPN. Here, only a single additional connection per layer is visualized. Details about up-sampling and merging of ResFPN can be found in Figure 7.8.*

connections between encoder and decoder can be applied in all cases.

The theoretical idea of ResFPN can include any additional connection of layers in pyramid networks that goes beyond regular lateral skip connections, e.g. dense connections. However, for dense matching it is argued that the set of possible connections can be restricted. More precisely, additional connections from lower resolved feature maps towards higher resolutions [XWLZ+18] are assumed to improve semantics only and do not contribute to the goal of better localization (they might even accomplish the opposite). As a result, the focus is on (multiple) connections from higher resolution feature maps of the encoder to feature maps of the decoder (see Figure 7.7d).

Along with these additional connections, novel questions arise. Higher resolution feature maps need to be adjusted to fit the spatial dimensions of the connected layer of the decoding branch. This can be done with any size-changing layer, e.g. strided convolution or pooling. Joining multiple feature maps into a single one requires a suitable strategy for merging. Commonly, either element-wise addition or concatenation is used. While the latter allows to maintain the separation of features, it can also lead to heavy computational loads for large and deep feature maps. Finally, it can be asked which layers should be additionally connected. In theory, the more higher levels are used, the more the focus is shifted towards localization. On the other hand, a dense connection of every higher resolution to every lower one might be impractical. Answers to these questions are given in the ablation study in Section 7.2.3.

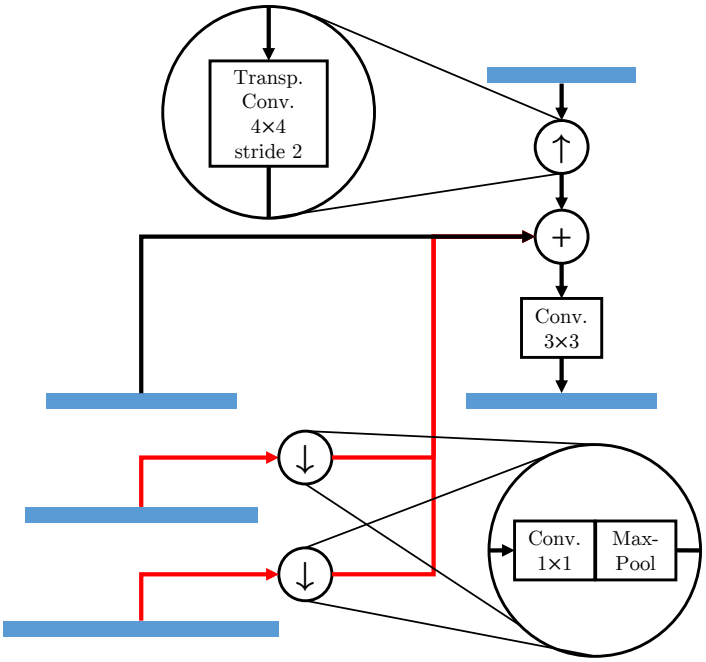



**Figure 7.8.:** *A single up-sampling block in the decoder of ResFPN combines four different resolutions. The previous lower resolved representation of the decoder is up-sampled with a transposed convolution, the equally resolved feature map from the encoder is connected through a classical skip connection, and two higher resolution feature encodings are additionally connected after down-sampling. For down-sampling,* $1 \times 1$ *convolution and max-pooling are applied. Merging is performed by element-wise addition followed by convolution.*

A final remark of the theoretical discussion of ResFPN is related to the spectrum of applications. It is argued that ResFPN is especially powerful when used for deeper pyramids that realize a (coarse-to-fine, incremental) multi-level prediction at multiple scales. However, the application of ResFPN is not limited to this use case. It is also possible to use only a certain level of the decoder for a single final prediction. The experiments (Section 7.2.3) cover a broad range of end-to-end differentiable, dense matching networks to demonstrate the flexibility of ResFPN.

**Feature Extraction Network.** A general ResFPN consists of $l_d$ arbitrary down-sampling blocks with a sub-sampling factor $s$ (usually $s = 2$), a bottleneck at a sub-sampled factor of $s^{l_d}$, and $l_d \leq l_u$ up-sampling blocks using the same factor $s$. In regular FPNs [LDGH+17], the up-sampling block merges the corresponding feature encoding of the target resolution with the up-sampled result to produce a refined feature map. In this extension of ResFPN, $h$ *additional* feature encodings of the next higher resolutions are also used during merging. The feature encodings of higher resolutions need to be reshaped to fit the spatial dimensions (and possibly the feature depth) of the target feature map. In theory, any resizing operation could be used for this task, e.g. strided convolution. Different strategies for reshaping and merging are compared in Section 7.2.3. Each (or one) of the feature maps of the decoder can then be used as features for the prediction.

One possible way to implement a ResFPN is described here. The architecture is based on the FPN of PWOC-3D which is an extension of the feature pyramid of PWC-Net [SYLK18]. In detail, $l_d = 6$ down-sampling blocks with a sub-sampling factor of $s = 2$ are used to compute 6 feature maps, where the first one has 1/2 of the input resolution and the deepest encoding has 1/64 of the input resolution. This is followed by $l_u = 4$ up-sampling blocks to reconstruct a feature map of 1/4 of the original image resolution. Higher resolutions are not required for most of the prediction heads in the experiments [CC18; SSWS19; SYLK18], but are possible. The down-sampling is performed by two $3 \times 3$ convolutions, where the first one applies a stride of 2. For up-sampling, a $4 \times 4$ transposed convolution with stride 2 is applied, the up-sampled features are merged with a regular skip connection and $h = 2$ additional lateral connections through element-wise addition, and the fused features are then refined with a $3 \times 3$ convolution. To align spatial size and feature depth for the merging of higher resolution feature encodings, a $1 \times 1$ convolution followed by max-pooling with a kernel size and stride of $s \cdot \Delta l$ is proposed. Reshaping, merging, and refinement during up-sampling is illustrated in Figure 7.8. In the experience, the combination of $1 \times 1$ convolution and max-pooling is in the sweet spot of preserving spatial accuracy and computational efficiency, especially when feature depth is increased during the convolution (which is usually the case from higher to lower resolutions) (cf. Section 7.2.3). LeakyReLU activation [MHN13] is used for all convolutions to introduce non-linearity into the model. The entire architecture of ResFPN with all details is given in Table 7.3.

**Table 7.3.:** *The detailed architecture of ResFPN. (Up)Conv$(c, k, s, d)$ and MaxPool$(k, s)$ describe (transposed) convolution and max-pooling with $c$ kernels, square kernel size $k$, stride $s$, and dilation rate $d$.*

| Name | Input | Layer | Output Shape |
|---|---|---|---|
| *input* | – | – | $H \times W \times 3$ |
| *enc-1-1* | *input* | Conv(16,3,2,1) | $\frac{1}{2}H \times \frac{1}{2}W \times 16$ |
| *enc-1-2* | *enc-1-1* | Conv(16,3,1,1) | $\frac{1}{2}H \times \frac{1}{2}W \times 16$ |
| *enc-2-1* | *enc-1-2* | Conv(32,3,2,1) | $\frac{1}{4}H \times \frac{1}{4}W \times 32$ |
| *enc-2-2* | *enc-2-1* | Conv(32,3,1,1) | $\frac{1}{4}H \times \frac{1}{4}W \times 32$ |
| *enc-3-1* | *enc-2-2* | Conv(64,3,2,1) | $\frac{1}{8}H \times \frac{1}{8}W \times 64$ |
| *enc-3-2* | *enc-3-1* | Conv(64,3,1,1) | $\frac{1}{8}H \times \frac{1}{8}W \times 64$ |
| *enc-4-1* | *enc-3-2* | Conv(96,3,2,1) | $\frac{1}{16}H \times \frac{1}{16}W \times 96$ |
| *enc-4-2* | *enc-4-1* | Conv(96,3,1,1) | $\frac{1}{16}H \times \frac{1}{16}W \times 96$ |
| *enc-5-1* | *enc-4-2* | Conv(128,3,2,1) | $\frac{1}{32}H \times \frac{1}{32}W \times 128$ |
| *enc-5-2* | *enc-5-1* | Conv(128,3,1,1) | $\frac{1}{32}H \times \frac{1}{32}W \times 128$ |
| *enc-6-1* | *enc-5-2* | Conv(196,3,2,1) | $\frac{1}{64}H \times \frac{1}{64}W \times 196$ |
| *enc-6-2* | *enc-6-1* | Conv(196,3,1,1) | $\frac{1}{64}H \times \frac{1}{64}W \times 196$ |
| *bottleneck* | *enc-6-2* | Conv(196,1,1,1) | $\frac{1}{64}H \times \frac{1}{64}W \times 196$ |
| *skip-5-6* | *enc-5-2* | Conv(196,1,1,1)<br>MaxPool(2,2) | $\frac{1}{64}H \times \frac{1}{64}W \times 196$ |
| *skip-4-6* | *enc-4-2* | Conv(196,1,1,1)<br>MaxPool(4,4) | $\frac{1}{64}H \times \frac{1}{64}W \times 196$ |
| *dec-6-2* | *bottleneck*<br>*+skip-5-6*<br>*+skip-4-6* | Conv(196,3,1,1) | $\frac{1}{64}H \times \frac{1}{64}W \times 196$ |
| *dec-5-1* | *dec-6-2* | UpConv(128,4,2,1) | $\frac{1}{32}H \times \frac{1}{32}W \times 128$ |
| *skip-4-5* | *enc-4-2* | Conv(128,1,1,1)<br>MaxPool(2,2) | $\frac{1}{32}H \times \frac{1}{32}W \times 128$ |
| *skip-3-5* | *enc-3-2* | Conv(128,1,1,1)<br>MaxPool(4,4) | $\frac{1}{32}H \times \frac{1}{32}W \times 128$ |
| *dec-5-2* | *dec-5-1*<br>*+enc-5-2*<br>*+skip-4-5*<br>*+skip-3-5* | Conv(128,3,1,1) | $\frac{1}{32}H \times \frac{1}{32}W \times 128$ |
| *dec-4-1* | *dec-5-2* | UpConv(96,4,2,1) | $\frac{1}{16}H \times \frac{1}{16}W \times 96$ |
| *skip-3-4* | *enc-3-2* | Conv(96,1,1,1)<br>MaxPool(2,2) | $\frac{1}{16}H \times \frac{1}{16}W \times 96$ |
| *skip-2-4* | *enc-2-2* | Conv(96,1,1,1)<br>MaxPool(4,4) | $\frac{1}{16}H \times \frac{1}{16}W \times 96$ |
| *dec-4-2* | *dec-4-1*<br>*+enc-4-2*<br>*+skip-3-4*<br>*+skip-2-4* | Conv(96,3,1,1) | $\frac{1}{16}H \times \frac{1}{16}W \times 96$ |
| *dec-3-1* | *dec-4-2* | UpConv(64,4,2,1) | $\frac{1}{8}H \times \frac{1}{8}W \times 64$ |
| *skip-2-3* | *enc-2-2* | Conv(64,1,1,1)<br>MaxPool(2,2) | $\frac{1}{8}H \times \frac{1}{8}W \times 64$ |
| *skip-1-3* | *enc-1-2* | Conv(64,1,1,1)<br>MaxPool(4,4) | $\frac{1}{8}H \times \frac{1}{8}W \times 64$ |
| *dec-3-2* | *dec-3-1*<br>*+enc-3-2*<br>*+skip-2-3*<br>*+skip-1-3* | Conv(64,3,1,1) | $\frac{1}{8}H \times \frac{1}{8}W \times 64$ |
| *dec-2-1* | *dec-3-2* | UpConv(32,4,2,1) | $\frac{1}{4}H \times \frac{1}{4}W \times 32$ |
| *skip-1-2* | *enc-1-2* | Conv(32,1,1,1)<br>MaxPool(2,2) | $\frac{1}{4}H \times \frac{1}{4}W \times 32$ |
| *skip-0-2* | *input* | Conv(32,1,1,1)<br>MaxPool(4,4) | $\frac{1}{4}H \times \frac{1}{4}W \times 32$ |
| *dec-2-2* | *dec-2-1*<br>*+enc-2-2*<br>*+skip-1-2*<br>*+skip-0-2* | Conv(32,3,1,1) | $\frac{1}{4}H \times \frac{1}{4}W \times 32$ |

### 7.2.3. Experiments and Results

ResFPN is designed to extract features for dense matching such as stereo disparity, optical flow, or scene flow estimation. The experiments cover end-to-end networks for all these matching tasks (cf. Section 7.2.1). In particular, PWOC-3D [SSWS19] (Section 7.1) is used for scene flow estimation, PWC-Net [SYLK18] and LiteFlowNet [HTC18] represent optical flow estimators, and PSMNet [CC18] is the network used for disparity estimation.

The experiments consider three well established data sets. FT3D [MIHF+16] is used in all cases for pre-training and evaluation. It provides dense scene flow ground truth and is thus also applicable for the training of optical flow or disparity networks. Further, the networks are fine-tuned on KITTI [GLU12; MG15] and Sintel [BWSB12]. The KITTI 2015 scene flow data set also provides (sparse) labels for scene flow and can therefore be used for all evaluations. Sintel is a data set for optical flow and is thus used for experiments related to optical flow only. For validation and evaluation, the random split of [SSWS19] (Section 7.1) is used for KITTI, and 5 out of the 23 sequences are randomly sampled for Sintel. These sequences are *alley_2*, *ambush_4*, *bamboo_2*, *cave_4*, and *market_5*. For augmentation, photometric transformations as in [DFIH+15; SSWS19] and temporal flipping for pre-training on FT3D are applied. Unless mentioned otherwise, pre-training and fine-tuning are done with a batch size of 2 and 1, respectively. The metrics being considered are the ones presented in Section 2.3.2. For all, lower is better. Using these setups, two sets of experiments are performed. Firstly, the design choices are evaluated in Section 7.2.3 and different ways to implement ResFPN are compared. Secondly, the features of ResFPN are applied together with different end-to-end matching networks in Section 7.2.3.

#### Design Decisions

There are multiple ways to implement the idea of ResFPN. In this section, different entities of ResFPN are compared by varying the number of additional skip connections $h$, the merging operation, and the method to adjust size and depth of the skip features. For these experiments, the up- and down-sampling blocks presented in Section 7.2.2 and Table 7.3 are used with the prediction head of PWOC-3D [SSWS19] for scene flow. The different variants are compared in Table 7.4.

The number of skip connections is varied from 1 ($h = 0$, the original FPN) to 3 ($h = 2$). More than three lateral connections are not realizable with the given hardware, yet it can be noticed clearly that an increase of connections improves the final results. Furthermore, concatenation versus addition is tested. Since the concatenation is independent of the feature depths of the merging input, it is not necessary to reshape the depth of the additional skip features. However, when this step is omitted, the performance decreases. If this step is included, it is not obvious what the output depth of the $1 \times 1$ convolution should be. For the numbers reported in Table 7.4, the output depth of the up-sampled target feature map is used, i.e. the same number of output channels

**Table 7.4.:** *Ablation study on the validation split of KITTI data for different numbers and kinds of residual connections with different strategies for merging. A simple FPN establishes only a single skip connection between layers of the same resolution. ResFPN adds two residual connections of higher resolutions (cf. Figure 7.8). Scene flow predictions of PWOC-3D [SSWS19] (Section 7.1) validate that the setup of ResFPN yields the best results while at the same time increases the computational effort and network size only marginally.*

| | $h$ | Re-shaping | Merging | FT3D [MIHF+16] KOE | FT3D [MIHF+16] EPE | KITTI [MG15] KOE | KITTI [MG15] EPE | Parameters $\times 10^6$ | FLOPs $\times 10^{12}$ |
|---|---|---|---|---|---|---|---|---|---|
| FPN [SSWS19] | 0 | – | addition | 21.49 | 9.15 | 12.55 | 3.22 | **8.05** | **6.07** |
| | 1 | $1 \times 1$, max-pool | addition | 20.95 | 8.28 | 11.37 | 3.09 | 8.09 | 6.50 |
| | 2 | max-pool | concatenation | 19.90 | 7.91 | 11.21 | 3.04 | 8.67 | 8.94 |
| | 2 | $1 \times 1$, max-pool | concatenation | 21.16 | 8.34 | 11.83 | 3.02 | 9.03 | 12.09 |
| | 2 | $3 \times 3$, stride | addition | 21.65 | 8.42 | 13.67 | 3.50 | 8.74 | 7.43 |
| | 2 | $1 \times 1$, bilinear | addition | 20.89 | 8.09 | 11.55 | 3.21 | 8.12 | 7.26 |
| | 2 | max-pool, $1 \times 1$ | addition | 20.28 | 7.67 | 12.24 | 3.06 | 8.12 | 6.24 |
| **ResFPN** | 2 | $1 \times 1$, max-pool | addition | **18.91** | **7.19** | **10.63** | **2.98** | 8.12 | 7.30 |

**Table 7.5.:** *Comparison of feature extractors. For different prediction networks on different data sets, two version are compared, the original network and a version where nothing but the feature module is changed to the improved ResFPN. To validate if the additional lateral connections in ResFPN are the reason for the improvement, a simple FPN [LDGH+17] is also included in the comparison.*

| | FT3D [MIHF+16] | | | | | | KITTI [MG15] | | | | | | Sintel [BWSB12] | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Original | | FPN | | **ResFPN** | | Original | | FPN | | **ResFPN** | | Original | | FPN | | **ResFPN** | |
| Prediction Head | KOE | EPE | KOE | EPE | KOE | EPE | KOE | EPE | KOE | EPE | KOE | EPE | KOE | EPE | KOE | EPE | KOE | EPE |
| PWOC-3D [SSWS19] | – | – | 21.5 | 9.2 | **18.9** | **7.2** | – | – | 12.6 | 3.2 | **10.6** | **3.0** | – | – | – | – | – | – |
| PWC-Net [SYLK18] | 19.9 | 8.5 | 19.4 | 8.4 | **18.7** | **8.2** | 15.6 | 3.7 | 14.6 | 3.3 | **13.9** | **3.2** | 20.2 | 6.0 | 19.6 | **5.7** | **18.5** | **5.7** |
| LiteFlowNet [HTC18] | 23.1 | 9.8 | 22.8 | 9.9 | **20.9** | **9.0** | 18.0 | 3.7 | 18.0 | 3.6 | **16.4** | **3.5** | 20.7 | 5.7 | 19.6 | 5.7 | **18.3** | **5.6** |
| PSMNet [CC18] | 16.0 | 5.3 | **10.9** | 5.2 | **10.9** | **4.9** | 3.0 | **1.0** | 2.6 | **1.0** | **2.2** | **1.0** | – | – | – | – | – | – |

that is required for merging by element-wise addition. As a consequence, the computational effort increases. Lastly, the re-shaping strategy to align spatial shapes is changed, and in case of addition the depth of the skip feature maps. The approach of $1 \times 1$ convolution followed by max-pooling is opposed to strided convolution, convolution followed by bilinear down-sampling, and max-pooling followed by convolution to show the importance of the order. Out of all strategies, the reshaping approach with element-wise addition and $1 + h = 3$ skip connections (visualized in Figure 7.8) performs the best while, at the same time, is computationally affordable. The overhead of the additional residual connections in terms of numbers of parameters and floating point operations is negligibly small, but the outlier rate (KOE) and end-point error (EPE) drop by 7 to 22 %. Note that the feature computation with either FPN or ResFPN requires less than 10 % of the entire floating point operations for the prediction of the scene flow with PWOC-3D [SSWS19]. In detail, feature computation with ResFPN on a GeForce GTX 1080 Ti for a single image ($\sim 0.5$ MP) takes about 5 ms.

### Dense Matching with ResFPN

Four different end-to-end networks for scene flow, optical flow, and disparity estimation are used for dense matching. For the experiments, nothing but the feature computation module is replaced with ResFPN (and a simple FPN [LDGH+17] for comparison). The predictions for the three different feature extractors are then compared. The evaluation is conducted on all mentioned data sets if ground truth for the respective task is available. The training schedules are as close as possible to the original, including multi-stage training if relevant, learning rate schedules, and more. Deviations from the original training schedule are explicitly mentioned. The results for all networks on all data sets are presented in Table 7.5.

**Stereo Disparity.** PSMNet [CC18] is used to compute stereo disparity. This network predicts single scale dense stereo displacements at 1/4 resolution, i.e. only the output of *dec-2-2* (see Table 7.3) from ResFPN is used for the prediction. For the comparison between baseline and ResFPN, the *CNN* module for feature extraction (see Table 1 in [CC18]) is replaced with ResFPN. To smooth the interface between ResFPN and the *SPP* module of PSMNet, the used feature representation is passed through a $1 \times 1$ convolution with 128 output channels to unify the feature shapes. For any training of PSMNet, a batch-size of 3 is used for pre-training. For the training of PSMNet together with ResFPN, the entire learning rate schedule is reduced by factor 10, because the additional skip connections affect the flow of gradients and thus can influence the stability.

The results show a significant reduction of outliers (KOE) for both stereo data sets when using ResFPN. End-point errors on FT3D are also reduced. ResFPN also outperforms the simple FPN with a single lateral skip connection.

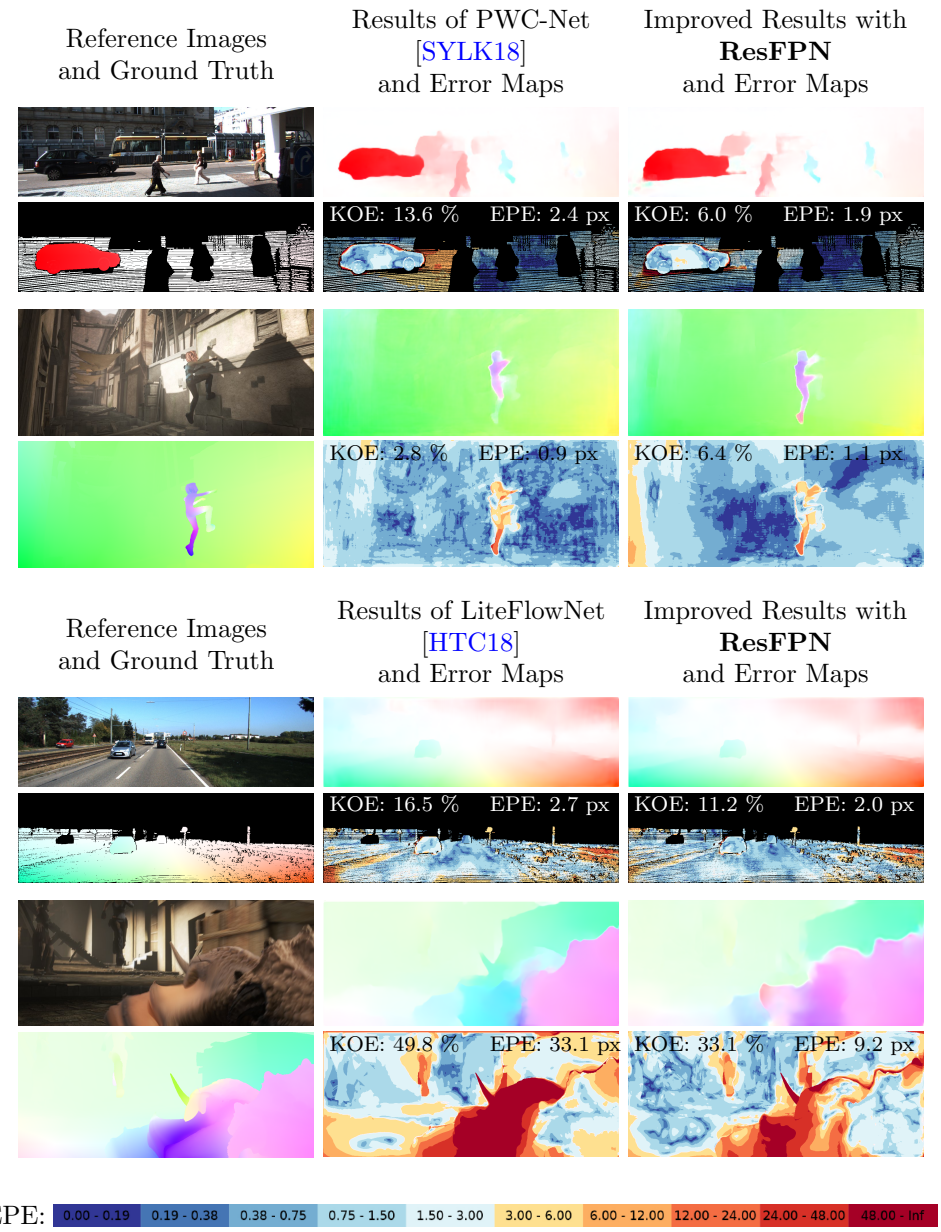



**Figure 7.9.:** *Some examples of how ResFPN improves optical flow prediction on KITTI [GLU12; MG15] and Sintel [BWSB12]. Please note the subtle differences around objects, e.g. vehicles.*

**Optical Flow.** PWC-Net [SYLK18] and LiteFlowNet [HTC18] are used for the estimation of optical flow. For the experiments with PWC-Net, the exact ResFPN as described in Section 7.2.2 and Table 7.3 is used. For LiteFlowNet, the flexibility of ResFPN is demonstrated by testing a version that is closer to the original feature computation module of LiteFlowNet. The concept of multiple residual skip connections in a pyramidal encoder-decoder network with the up-sampling block shown in Figure 7.8 is still applied, but the hyper-parameters are changed to fit the settings of the encoder of LiteFlowNet [HTC18]. In detail, the feature encoder is formed by the input image, a first feature representation at full resolution, and the five additional, down-sampled feature maps. This setup reaches a minimal resolution of 1/32 with feature depths of $3, 32, 32, 64, 96, 128, 192$ for the seven parts of the encoder (including the image itself). For the prediction, multiple scales are used iteratively until 1/2 of the input resolution is reached. This is different from all other networks, which use a final resolution for prediction of 1/4.

For both optical flow networks, the results improve on all data sets when features from ResFPN are used. This holds for both metrics, outlier rate and average end-point error. ResFPN also outperforms a simple FPN [LDGH+17] in all experiments on optical flow. A visual comparison of the results of the baselines and ResFPN is given in Figure 7.9 for exemplary images from KITTI and Sintel. It is evident that not only the localization of features is improved to capture more details during matching. Moreover, ResFPN shows an increased robustness compared with its competitors in general. In the first sample from Sintel, the relatively small, insufficiently illuminated character is outlined much better when ResFPN features are used for the matching, even if the overall results for this frame are slightly worse. On a global scale, especially for large displacements or occluded areas, ResFPN outperforms the baseline (e.g. in the last example of Figure 7.9).

**Scene Flow.** For estimation of scene flow with PWOC-3D [SSWS19] (Section 7.1), the original design of ResFPN is applied again. The major differences here are that four instead of two images are processed for matching with ResFPN and that the baseline is already using a FPN with lateral skip connections [LDGH+17; SSWS19]. Therefore, this experiment has the strongest baseline. Still, ResFPN achieves a considerable reduction of outliers of about 15 % and cuts the end-point errors by ~6 % and ~22 % for KITTI and FT3D, respectively.

In summary, using ResFPN for feature computation in end-to-end matching networks reduces outlier rates and end-point errors (or maintains them) in all experiments. The better localized features preserve details during matching and produce more consistent and smooth results in comparison to a simple Feature Pyramid (FP) and a basic FPN. ResFPN achieves this for networks with very different characteristics, e.g. single and multi-scale estimation, different encoding (down-sampling) blocks, and different final resolutions.

**Table 7.6.:** *Evaluation in boundary regions of objects on KITTI [MG15]. $d_n$ defines the average end-point error for areas around object boundaries of $n$ pixels width.*

| Predictor | Original | | | | with **ResFPN** | | | |
|---|---|---|---|---|---|---|---|---|
| | $d_3$ | $d_5$ | $d_{10}$ | $d_{20}$ | $d_3$ | $d_5$ | $d_{10}$ | $d_{20}$ |
| PWOC [SSWS19] | 10.75 | 10.23 | 8.24 | 6.54 | 10.18 | 9.87 | 8.30 | 6.80 |
| PWC [SYLK18] | 10.34 | 9.79 | 8.25 | 6.94 | 9.46 | 9.06 | 7.91 | 6.84 |
| LFN [HTC18] | 13.72 | 12.57 | 10.13 | 8.74 | 12.36 | 11.43 | 9.15 | 7.72 |
| PSM [CC18] | 2.07 | 2.31 | 2.12 | 1.70 | 2.77 | 2.46 | 1.79 | 1.35 |

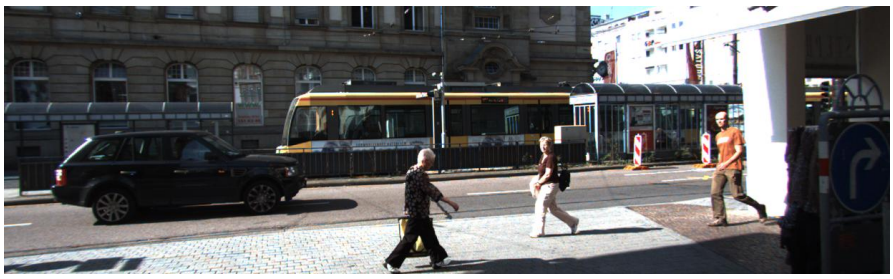
**Improved Localization.** The previous section confirms that matching with ResFPN yields an overall better result on various domains with all kinds of networks. However, one of the major claims is improved localization by the utilization of multiple feature maps of higher resolution. Therefore, a final experiment is conducted to validate this claim. Towards this end, the object masks provided by the KITTI data set [MG15] are used to repeat the previous experiment on the regions around objects. The average end-point error for different maximum distances to object boundaries is evaluated and reported in Table 7.6.

The numbers indicate that results obtained from the proposed feature module are better at discontinuities around objects in most cases. Except for very narrow evaluation regions for predictions with PSMNet [CC18] and wide boundaries for scene flow prediction with PWOC-3D [SSWS19], ResFPN reduces the error in these difficult image regions.

## 7.3. Deep Temporal Fusion of Motion Between Multiple Time Steps

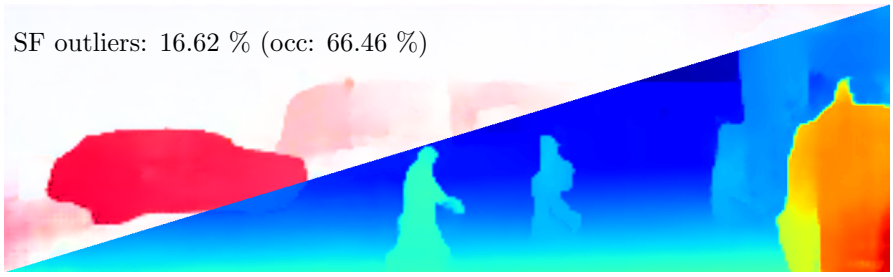
For all presented and existing dual-frame approaches, occlusions and out-of-view motions are a limiting factor. In the context of environmental perception for vehicles this issue is especially serious due to the large (ego-) motion of objects. This part of the dissertation proposes a novel data-driven approach for Deep Temporal Fusion (DTF) [SUS21] of scene flow estimates in a multi-frame setup to overcome the issue of occlusion. In contrast to most previous multi-frame methods, this approach does not rely on a constant motion model, but instead learns a generic temporal relation of motion from data. In a second step, a neural network combines bidirectional scene flow estimates from a common reference frame, yielding a refined estimate and a natural byproduct of occlusion masks. This way, the framework provides a fast multi-frame extension for a variety of scene flow estimators, which outperforms the underlying dual-frame approaches (see Figure 7.10).

Most recently, data-driven deep learning approaches have pushed the limits of scene flow estimation even further [APTM20; JSJL+19; MWHX+19; SSWS19; YR20]. These approaches achieve state-of-the-art results at run times close

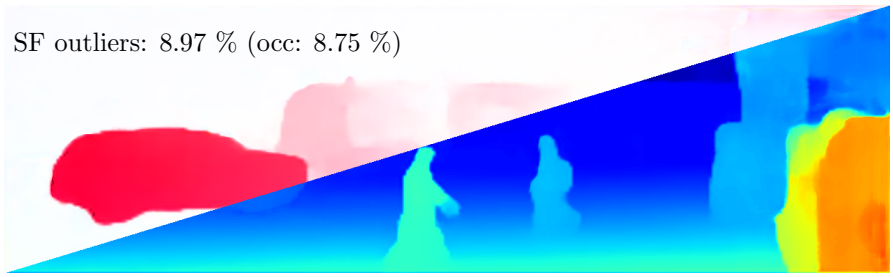


**(a)** *Reference image.*


**(b)** *Soft occlusion weights for fusion.*




**(c)** *Dual-frame result from PWOC-3D [SSWS19] (Section 7.1).*




**(d)** *Result of the proposed fusion framework.*

**Figure 7.10.:** *Deep Temporal Fusion refines an initial dual-frame estimate by combination with an inverted backward scene flow. The fusion is realized as a pixel-wise weighted averaging and thus yields (soft) occlusion maps (b). This way, the initial results are significantly outperformed, especially in the difficult occluded areas.*

to real time. Yet, none of these deep learning methods utilizes a multi-frame setup which has been shown to improve over a conceptually similar dual-frame approach for heuristic algorithms [NŠ17; SWUK+20; TSS17; VSR15] (cf. Section 4.2). Many of these traditional, heuristic approaches use the additional information from multiple views to regularize the matching, making them more complex and reliable on specific, simplified motion models (e.g. a constant motion assumption). At the same time, all previous approaches (even multi-frame based) perform considerably worse in occluded areas (cf. Table 7.8), which suggests that there is a lot of unused potential in multi-frame scene flow estimation.

More generic concepts for learning-based multi-frame settings are proposed in the context of optical flow [LLKX19; MB18; NŠM18; RGSY+19]. But these methods do not model the underlying issue of occlusions at all, or tackle the estimation of occlusions by bidirectional flow estimation (twice as much effort). The framework can be used together with any auxiliary scene flow estimator.

### 7.3.1. Related Networks in the Literature

**Deep End-to-End Scene Flow Estimation.** A boost in run time has been achieved with the introduction of the first deep learning algorithms due to the massive parallelization on GPUs. At the same time, many of the newly proposed deep neural networks have reached state-of-the-art results despite the lack of realistic, labeled training data [APTM20; JSJL+19; MWHX+19; SSWS19; YR20]. Yet, no existing deep learning architecture for scene flow estimation makes use of the multi-frame nature of image sequences, which naturally exists in realistic applications. DTF fills this gap with a trainable, generic multi-frame solution for scene flow estimation.

**Deep Multi-Frame Models for Optical Flow.** For optical flow there exists some previous work on deep multi-frame neural networks. MFF [RGSY+19] computes forward flow for two consecutive time steps together with a backward flow for the central frame. The backward flow is used to warp the previous forward motion towards the reference frame realizing a constant motion assumption. A fusion network then combines the initial forward prediction and the warped one. This fusion effectively relaxes the constant motion constraint. Occlusions are not modeled explicitly here. ContinualFlow [NŠM18] uses previous flow estimates as additional input during the estimation of the current time step. Here, occlusions are learned as attention maps in a self-supervised manner similar to MaskFlownet [ZSDC+20] or PWOC-3D [SSWS19] (Section 7.1), but based on a cost volume instead of image features. ProFlow [MB18] proposes an online inverter for motion that is trained for every frame on the fly. In DTF, this idea is adopted to avoid warping, but only a single inverter is trained once to further avoid the re-training on every sample and the explicit estimation of occlusions at an early stage. In SelFlow [LLKX19] as in ProFlow also, occlusions are detected by a forward-backward consistency check. SelFlow uses the additional multi-frame information by constructing cost volumes for

forward and backward direction which are then used for the flow estimation. DTF eradicates any consistency checks, avoids warping in order to shift the handling of occlusions to a later stage, and learns a dedicated universal model for the inversion of motion. Contrary to all mentioned cases for optical flow estimation, the proposed deep multi-frame model solves the more complex problem of scene flow estimation.

### 7.3.2. Deep Multi-Frame Scene Flow

Consider a stream of stereo image pairs $I_t^l$ and $I_t^r$ for the left and right camera at a given time $t$. While dual-frame solutions only consider the four images at the two time steps $t$ and $t+1$, a multi-frame method incorporates information from at least one additional time (usually $t-1$ to avoid delay in the prediction and account for the symmetry in motion). The DTF framework builds on this exact setup using three stereo pairs at time $t-1$, $t$, and $t+1$. The idea is outlined in Figure 7.11 and can be summarized as follows. An arbitrary auxiliary model is used for scene flow estimation to predict forward ($t \rightarrow t+1$) and backward ($t \rightarrow t-1$) scene flow with respect to the reference view. By using a shared reference view for the fusion, warping can be avoided, and thus the problem to handle occlusions is postponed. Then, a learned motion model transforms the backward estimate into a forward motion. Finally, a temporal fusion module combines the forward and inverted estimate to obtain a refined result. For the fusion, a strategy of weighted averages is applied. This implicitly yields soft occlusion maps for the two motion directions without explicit supervision on occlusions. The underlying dual-frame model that is used mainly is PWOC-3D [SSWS19] of Section 7.1 due to its simple training schedule compared to other approaches. However, in the experiments (Section 7.3.3) it is shown that the framework is not limited to this model. The novel sub-networks for motion inversion and fusion are presented in more detail in the next sections.

**Temporal Scene Flow Inversion.** Instead of a constant motion assumption, which is often applied in previous work, a compact neural network is trained that utilizes a learned motion model to temporally invert scene flow. The architecture is inspired by the inversion module of [MB18] but it is deeper since the framework requires a generic model that can invert motion for arbitrary sequences without the need of re-training on every frame. In detail, the inversion sub-network consists of four convolutional layers with a kernel size $3 \times 3$ and a fifth one with a $7 \times 7$ kernel and output feature dimensions of $16, 16, 16, 16, 4$ respectively. The last layer is activated linearly. Similarly to ProFlow [MB18], the inverter is equipped with a mechanism for spatial variance by concatenating the input scene flow with normalized ($[-1, 1]$) spatial image coordinates of the x- and y-direction. This way and together with the depth information from the backward scene flow, the inversion network is able to operate fully in (hypothetical) 3D space. For a qualitative impression of the inverter, Figure 7.12 visualizes the results for a validation sample from FT3D.

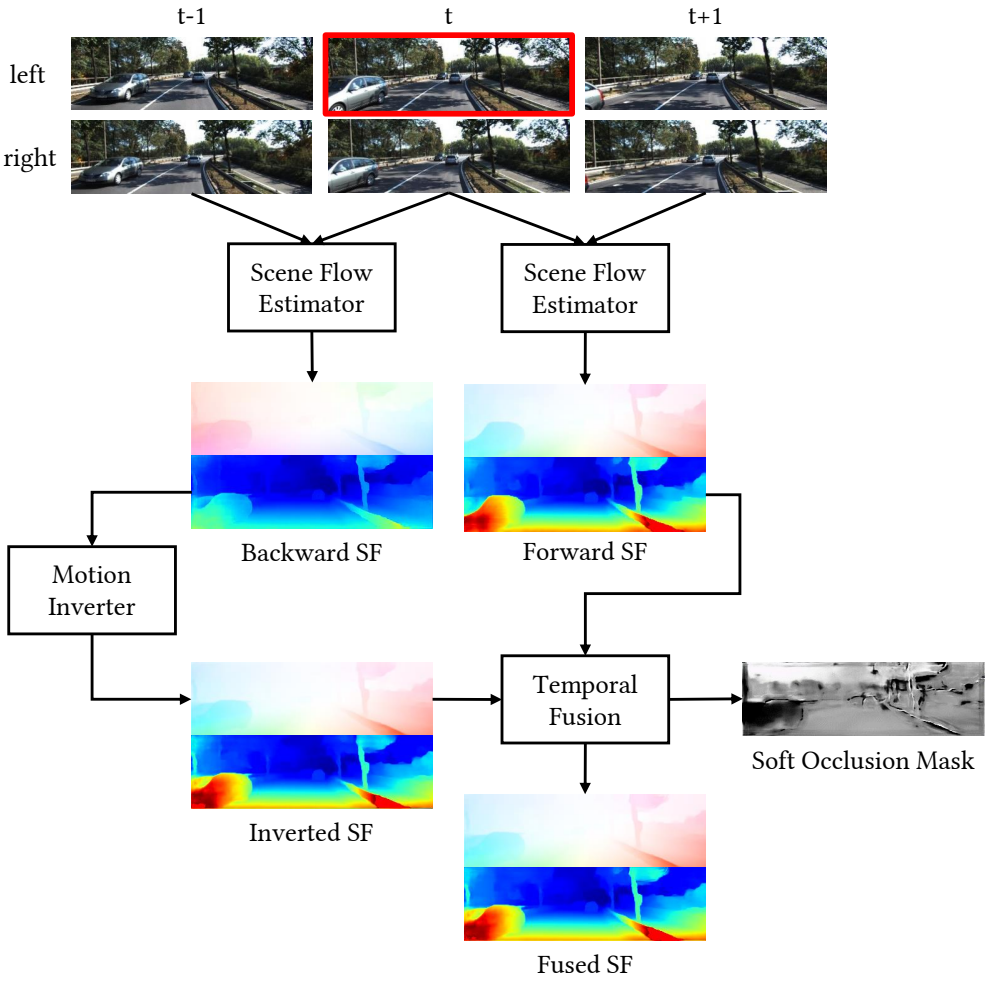



**Figure 7.11.:** *Overview of the proposed framework for Deep Temporal Fusion (DTF) with a trainable motion model.*

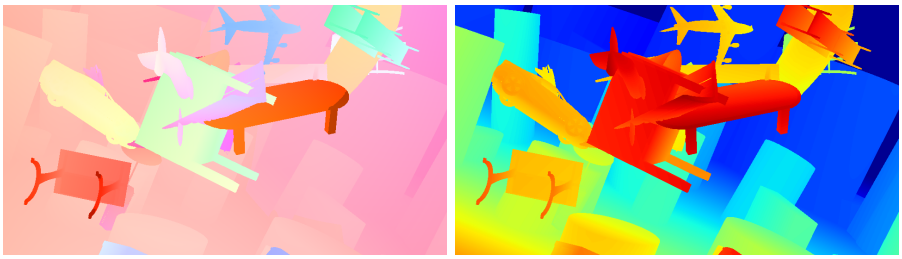
**Deep Forward-Backward Fusion.** After the prediction of scene flow in the forward and backward direction (using the same reference frame) and inverting the backward estimate, the forward and inverted backward predictions can be merged. The refined results can potentially overcome errors in difficult regions of occlusion or out-of-view motion, because occlusions occur rarely across multiple views [SWUK+20] (cf. Section 4.2.1). The fusion strategy follows a weighted average approach, where a fusion module predicts pixel-wise weights (that sum up to one) for the combination of the original forward estimate and the inverted backward scene flow. Interestingly, these weights correspond to (soft) occlusion masks, revealing the main reason why the inverted backward motion should be preferred over a forward dual-frame estimate (cf. Figures 7.10 and 7.11). While the direct prediction of a refined (or residual) scene flow during fusion is also possible, this would neither model the underlying issue nor produce occlusion masks.

For the fusion module, the architecture of the context network of PWC-Net [SYLK18] and PWOC-3D [SSWS19] (cf. Section 7.1.2) is adopted. It consists of seven convolutional layers with a kernel size of $3 \times 3$, an output depths of $32, 64, 128, 128, 64, 32, 2$, and dilation rates of $1, 2, 4, 8, 16, 1, 1$ respectively. The last layer predicts pseudo probabilities in a one-hot encoding for the forward and inverted backward scene flow which are used for weighted averaging after a softmax activation. As input for this module, the forward and inverted backward estimates are concatenated.

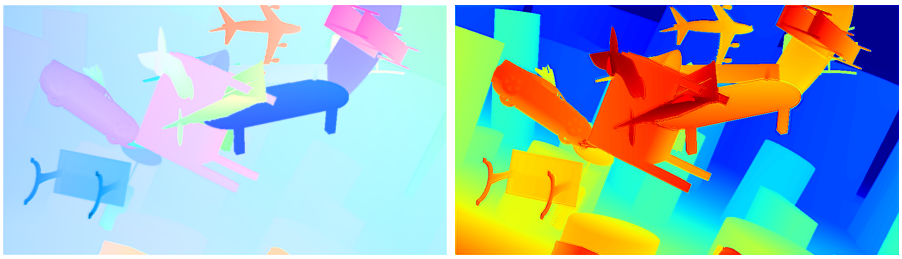
The above described model is a simple baseline for temporal fusion of scene flow (*basic*). Within the experiments in Section 7.3.3 different variants of the fusion module are compared. Though the network can detect occlusion based on the depth (disparity) and motion of neighboring pixels, it can not estimate out-of-view motion without knowing where the field of view ends. This information could be guessed from padding artifacts during convolution, however for more explicit modeling, additional spatial information is fed to the module, analogous to the inverter. This variant is denoted as *spatial*. Another variant is again motivated by the issue of occlusion. Since in multiple views different parts of the reference image are occluded, it is argued that the predicted occlusion masks (fusion weights) should differ for the different components of the scene flow, e.g. between the left and right view of a stereo camera, there are no occlusions due to motion. Therefore this variant predicts a separate occlusion map for each channel of the scene flow representation (in image space) and is depicted as *4ch* since it predicts fusion weights for four scene flow channels (two for optical flow and two for initial and future disparities). Lastly, both strategies are combined and the combination is named *spatial-4ch*. In Figures 7.10 and 7.11, the occlusion maps (fusion weights) for the *basic* variant are shown for the sake of clarity and space.

### 7.3.3. Experiments and Results

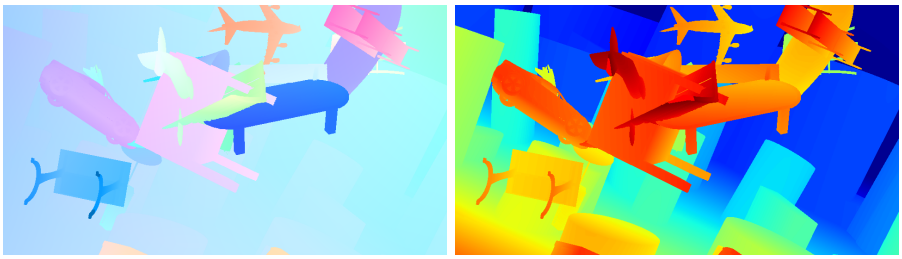
The experiments and results are split into three sets with the following main intentions. First of all, it is validate that the overall framework improves over the initial dual-frame estimates of different auxiliary scene flow models.



**(a)** *Backward scene flow visualized by optical flow and future disparity at time $t+1$.*



**(b)** *Inverted scene flow visualized by optical flow and future disparity at time $t+1$.*



**(c)** *Forward scene flow visualized by optical flow and future disparity at time $t+1$.*

**Figure 7.12.:** *An example of the learned inversion of motion on data of FlyingThings3D [MIHF+16]. The left and right columns show the optical flow and disparity at $t+1$ components of the scene flow. The first and last row give the ground truth in backward and forward direction, respectively. The center row presents the results of the generic motion inverter.*

Secondly, DTF is compared to existing multi-frame scene flow algorithms using the official public KITTI benchmark [GLU12; MG15]. Lastly, the goal is to validate each step of the framework separately by means of an extensive ablation study. As usual, the metrics in use are these of Section 2.3.2.

**Training and Implementation Details.** As before, the limited size of the KITTI data set [GLU12; MG15] requires alternative training strategies. Despite the success on unsupervised scene flow estimation [HR20] or knowledge distillation from teacher networks [APTM20; JSJL+19], transfer learning by pre-training and fine-tuning is the most common strategy to overcome this issue [MIFH+18; SSWS19; SYLK18; SYLK19]. This strategy is applied in Sections 7.1 and 7.2 and is also used here. The one large-scale data set which provides sufficient labeled data for scene flow is FT3D [MIHF+16].

The validation set on KITTI is the one consistently used in Chapter 7. For FT3D, the validation set consists of the last 50 sequences from each subset $A$, $B$, and $C$ of the *train* split.

If required, the auxiliary scene flow estimators are initialized with the published pre-trained weights. The rich ground truth of FT3D [MIHF+16] is used to separately pre-train the inverter on forward and backward ground truth motion with a L2-loss for 40 epochs, a batch size of 4, and an initial learning rate of $1 \times 10^{-4}$ that is decreased to $5 \times 10^{-5}$ and $1 \times 10^{-5}$ after 20 and 30 epochs, respectively. The rest of the pipeline is initialized from scratch.

Afterwards, the fusion pipeline is fine-tuned on KITTI [MG15] for 100 epochs. The learning rate for fine-tuning starts at $5 \times 10^{-5}$ and is again reduced to $1 \times 10^{-5}$ after 75 epochs. Due to memory limitations, a batch size of 1 is used whenever the entire pipeline is used for training.

Unless mentioned otherwise, LeakyReLU [MHN13] with a leak factor of 0.1 is used after each convolution. For all training stages, the Adam optimizer [KB15] is utilized with its default parameters.

The used robust loss function for the 4-dimensional scene flow in image space is similar to the one in [SSWS19; SYLK18] and defined by

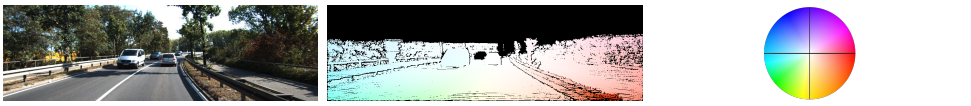
$$\mathcal{L} = \frac{1}{N_{GT}} \cdot \sum_{\boldsymbol{s} \epsilon GT} \left( |\hat{\boldsymbol{s}} - \boldsymbol{s}|_1 + \epsilon \right)^{0.4} . \tag{7.7}$$

Here $\hat{s}$ and $s$ are the estimated and corresponding ground truth scene flow vectors, $|\cdot|_1$ is the $L_1$-norm, $\epsilon = 0.01$ is a small constant for numerical stability, and the power of 0.4 gives less weight to strong outliers.

For the entire pipeline, this loss is imposed on the forward estimate, the inverted backward scene flow, and the final fusion:

$$\mathcal{L}_{total} = \mathcal{L}_{fw} + \mathcal{L}_{inv} + \mathcal{L}_{fused} \tag{7.8}$$

This multi-stage loss avoids that during training the fusion flips to one side and does not recover because the other side would not receive any updates anymore.



**(a)** *Reference image.* **(b)** ***Forward*** *ground truth (enhanced).* **(c)** *Color code of the optical flow.*

**(d)** *SENSE [JSJL+19]* **(e)** *Optical Expansion (OE) [YR20]* **(f)** *DWARF [APTM20]*

**(g)** *PWOC-3D [SSWS19], original* **(h)** *PWOC-3D [SSWS19], re-trained* **(i)** *SFF [SWKB+18]*

**Figure 7.13.:** *Visualization of the* ***backward*** *optical flow for different scene flow estimators. Most auxiliary estimators used in the experiments have difficulties with backward motion because they do not perform actual matching, but rather rely on the image information of the reference frame alone, especially for street surfaces. Significant improvements are noticeable once the backward branch is trained end-to-end within the DTF framework (h), even though backward ground truth is not available.*

**Table 7.7.:** *Comparison of the multi-frame fusion approach to the dual-frame results of the underlying auxiliary scene flow estimator for the entire image (all) and occluded areas only (occ ∈ all ∖ noc) on the KITTI validation split. The last column gives the maximum relative improvement of DTF over the respective dual-frame baseline.*

| Scene Flow Estimator | Setup | all | | | | occ | | | | max. rel. Improv. |
|---|---|---|---|---|---|---|---|---|---|---|
| | | D1 | D2 | OF | SF | D1 | D2 | OF | SF | |
| SENSE [JSJL+19] | Dual | 0.97 | 2.22 | 3.00 | 4.04 | 2.08 | 8.23 | 7.19 | 11.84 | |
| | **DTF** | 0.97 | 1.66 | 3.01 | 3.52 | 2.05 | 4.81 | 7.21 | 8.57 | 41.6 % |
| OE [YR20] | Dual | 1.11 | 2.58 | 5.56 | 6.61 | 2.53 | 7.34 | 15.06 | 17.73 | |
| | **DTF** | 1.12 | 2.46 | 5.46 | 6.39 | 2.54 | 6.97 | 14.57 | 16.86 | 5.0 % |
| DWARF [APTM20] | Dual | 2.35 | 3.49 | 7.07 | 8.16 | 3.94 | 7.59 | 17.70 | 19.63 | |
| | **DTF** | 1.17 | 2.63 | 5.64 | 6.75 | 2.82 | 7.54 | 14.90 | 17.82 | 50.2 % |
| PWOC-3D [SSWS19] | Dual | 4.65 | 6.72 | 11.50 | 13.64 | 8.02 | 15.20 | 29.17 | 32.15 | |
| | **DTF** | 3.34 | 4.85 | 8.22 | 9.70 | 5.63 | 10.10 | 18.68 | 21.24 | 36.0 % |
| SFF [SWKB+18] | Dual | 6.61 | 10.28 | 12.39 | 15.76 | 9.94 | 19.57 | 26.08 | 30.74 | |
| | **DTF** | 6.04 | 9.03 | 11.43 | 14.30 | 8.77 | 15.91 | 22.85 | 26.25 | 18.7 % |

**Comparison to the Auxiliary Estimators.** In Table 7.7 it is validated that the deep temporal fusion framework surpasses a diverse set of underlying dual-frame estimators in terms of scene flow outliers. Especially in the difficult areas of occlusion, the approach achieves significantly better results, reducing the scene flow outlier rate by up to ~ 30 %. The fusion improves the scene flow estimates for non-occluded areas also, resulting in an overall improvement over *all* image areas. For OE [YR20], the relative improvement is less compared to other auxiliary estimators. This has two reasons. First of all, some scene flow algorithms are heavily biased towards forward motions (cf. Figure 7.13) and therefore provide much less reliable information for fusion in the backward branch. Secondly, the estimate of motion-in-depth from OE depends a lot on the optical flow estimate, which amplifies the previous limitation and expands it to the complete scene flow estimation in backward direction. The first reason additionally motivates an end-to-end training of the fusion framework together with the auxiliary estimator. This is performed for PWOC-3D [SSWS19] (Section 7.1) because it is easiest to train. The other auxiliary estimators are used as off-the-shelf replacements with the officially provided pre-trained weights. DTF is even able to improve non-learning-based results of SceneFlowFields (SFF) [SWKB+18] (Section 4.1), with a noticeable margin of more than 10 % in occluded areas. Here, the smaller relative improvements are accounted to the ego-motion model applied in SFF which is able to estimate out-of-view motions in forward direction for the background more reliably. A visual comparison between PWOC-3D and the multi-frame extension of the DTF framework is conducted in Figure 7.14.

**Comparison to State-of-the-Art.** To check the generalization of the model on more unseen data, results obtained with the deep multi-frame model are submitted to the KITTI online benchmark. The results of all multi-frame methods and related dual-frame baselines are presented in Table 7.8. Due to the limited number of training samples on KITTI, some over-fitting can be observed when comparing the numbers to the results on the validation split. However, improvements over the underlying dual-frame models (SENSE and PWOC-3D) are still evident, again with margins of ~15 - 20 % in occluded areas. Since KITTI evaluates the submitted results only for non-occluded (*noc*) and all valid pixels, the results for occluded areas (*occ*) are reconstructed from the available data. To this end, the ratio of non-occluded image areas on the KITTI *training* set are computed (84.3 %), and this distribution is used to estimate the results in occluded areas only on the KITTI *testing* set based on the benchmark results of non-occluded (*noc*) and *all* areas according to the following formula:

$$occ_r = \frac{all_r - noc_r \cdot 0.843}{0.157} \tag{7.9}$$

for the regions $r \in \{bg, fg, all\}$. This strategy reveals that even for the top performing multi-frame methods, moving vehicles which leave the field of view are the most challenging cases. In these regions (*occ-fg*), the fusion approach achieves top performance. In foreground regions it furthermore

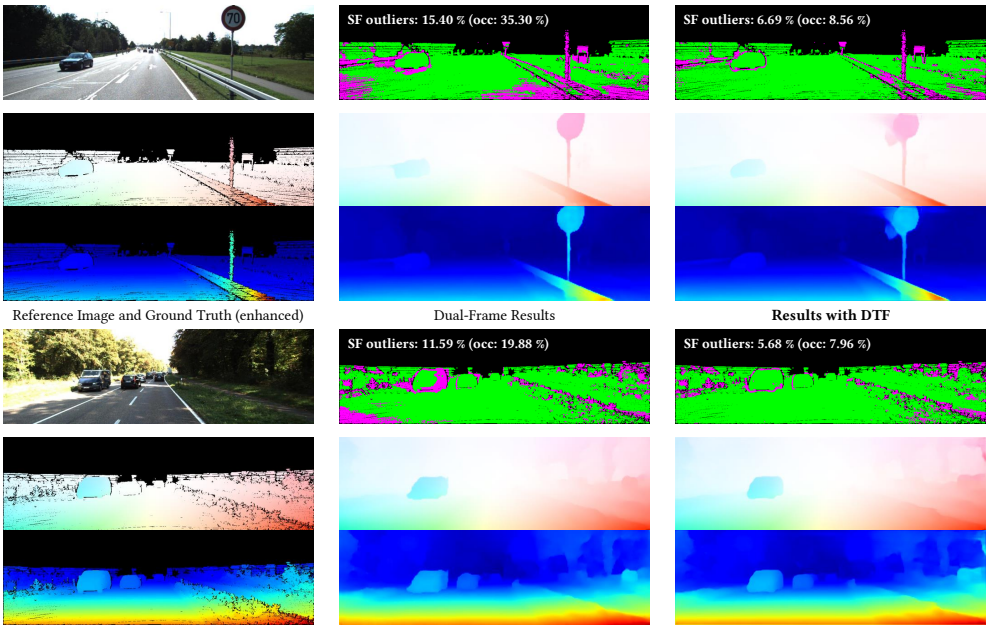



**Figure 7.14.:** *Visual comparison of the deep multi-frame fusion framework to the auxiliary dual-frame model PWOC-3D [SSWS19] of Section 7.1. Scene flow results are shown by optical flow and disparity at time $t + 1$. The error maps indicate scene flow outliers in magenta and inliers in green. Notice the improvements in occluded areas (e.g. in front of and around vehicles) or the out-of-view occlusions due to ego-motion (e.g. the close-by part of the guardrail in the first example and the lower image corners).*

performs significantly better (*all-fg*) than other multi-frame methods. Lastly, it is highlighted that since this is the first deep method for multi-frame scene flow estimation, the run time is close to real time and thus 2 to 5 orders of magnitude faster than the run time of most other multi-frame methods. The inversion and fusion without auxiliary scene flow estimation takes 0.12 seconds. A GeForce RTX 2080 Ti is used for inference in these experiments.

**Ablation Study.** For completeness, each part of the DTF framework is evaluated separately in Table 7.9. The first two rows show the results of the forward prediction and the inverted backward scene flow after end-to-end training. It can be seen that within multi-frame training, the plain forward prediction improves over the dual-frame baseline (cf. Table 7.7). Further, the results of the backward branch after inversion indicate that the motion inversion of optical flow is a bottleneck. Yet, for occluded areas the inversion outperforms the forward prediction already in terms of change of disparity, validating its importance. Both of these observations are confirmed by an evaluation of the inverter only on data of FT3D [MIHF+16] as shown in the fourth row of Table 7.9 (cf. Figure 7.12) compared to a naïve constant linear motion assumption in 2D. This is, optical flow and change of disparity are multiplied by −1. The learned motion model outperforms the constant motion model in terms of optical flow. Though, one might doubt whether the quality of the inversion is good enough to improve the forward prediction. Therefore, an *oracle* fusion is computed using the ground truth to select the better estimate from the forward and inverted backward branch. This experiment produces a theoretical bound for the fusion module and makes apparent that the inverted backward scene flow contains a lot of valuable information. Within the last four rows of Table 7.9 the different variants of the fusion module as described in Section 7.3.2 are compared. Results in occluded areas reveal that all variants including the *basic* one effectively tackle the problem of occlusion. Among all, the *spatial* version performs the worst unless combined with the *4ch* variant. However, stronger over-fitting is observed for this model with most representation power (and highest number of parameters). As a result, over the entire image area, the fusion module using four weight channels performs the best. Worth highlighting is that the fusion results in occluded areas reach the level of the oracle prediction almost.

## 7.4. Summary

PWOC-3D is a novel end-to-end CNN pipeline to predict scene flow (optical flow, stereo disparity, and disparity change jointly) directly from stereo image sequences. The approach is significantly more efficient than earlier classical approaches due to the massive parallelization of CNNs on GPUs, and much more accurate than variational methods. Moreover, unlike most previous techniques, PWOC-3D does not make any assumptions about the consistency or smoothness of motion, or the rigidity of objects. This makes the method more general and applicable to realistic scenarios in which such assumptions

**Table 7.8.:** *Results of the KITTI scene flow benchmark for all multi-frame approaches. Results of the auxiliary scene flow methods used in the pipeline and conceptual dual-frame counterparts for other multi-frame methods are also presented, if existent. Scene flow outlier rates (SF) are presented for foreground (fg), background (bg), and all regions, as well as for non-occluded areas (noc), occluded areas only (occ, details in the text), and the union (all).*

| | Method | SF Outliers [%] | | | | | | | | | Run Time [s] |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | occ | | | noc | | | all | | | |
| | | bg | fg | all | bg | fg | all | bg | fg | all | |
| multi-frame | PRSM [VSR15] | **12.36** | 37.65 | **15.74** | 5.54 | 17.65 | 7.71 | **6.61** | 20.79 | **8.97** | 300 |
| | **DTF+SENSE** | 16.37 | **37.49** | 19.65 | 6.69 | **9.72** | **7.23** | 8.21 | **14.08** | 9.18 | 0.76 |
| | OSF+TC [NŠ17] | 15.46 | 43.98 | 19.49 | **5.52** | 15.57 | 7.32 | 7.08 | 20.03 | 9.23 | 3000 |
| | SFF++ [SWUK+20] | 26.40 | 48.36 | 30.91 | 9.84 | 21.04 | 11.55 | 12.44 | 25.33 | 14.59 | 78 |
| | **DTF+PWOC-3D** | 31.91 | 51.14 | 34.29 | 8.79 | 21.01 | 10.98 | 12.42 | 25.74 | 14.64 | **0.38** |
| | FSF+MS [TSS17] | 21.59 | 65.48 | 27.63 | 9.23 | 28.03 | 12.60 | 11.17 | 33.91 | 14.96 | 2.7 |
| dual-frame | SENSE [JSJL+19] | 17.22 | 44.86 | 21.63 | 6.71 | 10.02 | 7.30 | 8.36 | 15.49 | 9.55 | 0.32 |
| | OSF [MG15] | 15.01 | 47.98 | 19.41 | 5.52 | 22.31 | 8.52 | 7.01 | 26.34 | 10.23 | 3000 |
| | PWOC-3D [SSWS19] | 41.20 | 47.52 | 41.62 | 9.29 | 18.03 | 10.86 | 14.30 | 22.66 | 15.69 | 0.13 |
| | SFF [SWKB+18] | 25.58 | 63.26 | 30.76 | 10.04 | 26.51 | 12.99 | 12.48 | 32.28 | 15.78 | 65 |
| | PRSF [VSR13] | 41.09 | 58.82 | 42.80 | 8.35 | 26.08 | 11.53 | 13.49 | 31.22 | 16.44 | 150 |

**Table 7.9.:** *Evaluation of intermediate results in the multi-frame fusion pipeline on the KITTI validation split. For this experiment, PWOC-3D [SSWS19] (Section 7.1) is the auxiliary estimator and is trained end-to-end. The inversion module is separately evaluated on FT3D.*

| Output | all | | | | occ | | | |
|---|---|---|---|---|---|---|---|---|
| | D1 | D2 | OF | SF | D1 | D2 | OF | SF |
| forward (fw) | 3.47 | 5.83 | 8.95 | 10.76 | 5.89 | 14.39 | 23.17 | 26.93 |
| inverted backward (bw-inv) | 4.15 | 6.00 | 20.34 | 22.14 | 6.64 | 9.92 | 31.74 | 33.81 |
| constant linear inversion (FT3D) | – | **1.27** | 47.16 | 47.18 | – | – | – | – |
| proposed inverter (FT3D) | 2.19 | 3.25 | **41.98** | **42.34** | – | – | – | – |
| fw + bw-inv + oracle | 2.63 | 3.91 | 6.25 | 7.51 | 4.53 | 8.40 | 16.39 | 18.43 |
| fw + bw-inv + fusion-basic | **3.22** | 4.90 | 9.01 | 10.48 | 4.88 | 10.23 | 19.27 | 21.66 |
| fw + bw-inv + fusion-spatial | 3.48 | 5.51 | 8.85 | 10.55 | 6.13 | 13.66 | 22.23 | 25.40 |
| fw + bw-inv + fusion-4ch | 3.34 | 4.85 | **8.22** | **9.70** | 5.63 | 10.10 | 18.68 | 21.24 |
| fw + bw-inv + fusion-spatial-4ch | 3.43 | **4.84** | 8.67 | 10.19 | **5.45** | **9.25** | **18.46** | **20.82** |

do not hold, e.g. highly dynamic road scenes.

However, as shown in Section 7.3.3 and Figure 7.13, a limitation of PWOC-3D and other end-to-end networks for scene flow estimation is the low generalization to unseen domains. These domain changes can be as simple as an inverted temporal order of the images.

Moreover, PWOC-3D employs special constructs such as pyramid processing, warping and occlusion reasoning to tackle common challenges in scene flow like large motion and occlusions. In this regard, a novel self-supervised scheme is proposed to estimate occlusion from images without any labeled occlusion data. PWOC-3D demonstrates competitive results on the KITTI benchmark and the FT3D data set. Notably, the method has significantly fewer parameters than contemporary methods and achieves second place on KITTI among end-to-end deep learning methods with 48 times fewer parameters than the top-performing method at the time of publication.

ResFPN – a multi-resolution feature pyramid network with residual skip connections provides a general concept to improve end-to-end networks for dense matching, e.g. scene flow estimation. With this novel design it is possible to significantly improve the representativity and localization of features for end-to-end learned dense pixel matching tasks. The design is validated in a comprehensive ablation study. In various experiments, ResFPN achieves significant improvements in application for optical flow, scene flow and disparity estimation. These improvements have been confirmed for a wide range of state-of-the-art methods over a large number of renowned data sets.

As future work, it is planned to explicitly consider further input modalities like LiDAR [BSWR+19] or radar [MK19] in the design of ResFPN. The additional 3D information plays an essential role for various applications. Furthermore, ResFPN can be analyzed with respect to its robustness against adversarial attacks [RJGB19].

DTF is a straight-forward integration of multiple frames to improve scene flow estimates for a wide range of dual-frame algorithms. Significant improvements could be achieved by inverting the backward motion of the reference view and fusing it with an initial forward estimate. Moreover, the fusion strategy of weighted averaging yields additional estimates of (soft) occlusion maps without the need for bidirectional consistency checks.

The experiments reveal that the inversion of optical flow is a limiting factor of the proposed approach, thus for future work it is suggested to equip the motion inverter with more domain knowledge to overcome this limitation and further to apply end-to-end training with other, more complicated auxiliary estimators.

Chapter 8

# Conclusions and Future Directions

*"Nothing clears up a case so much as stating it to another person."*

— Sir Arthur Conan Doyle, *Silver Blaze*

This thesis contributes to several aspects of scene flow estimation. First of all and most importantly, it has been shown that the sparse-to-dense concept works and is, with respect to previous techniques, favorable in terms of complexity and generalization. The accuracy is competitive. After a first attempt for sparse-to-dense scene flow estimation, the biggest challenges of sparse distribution and size of interpolation gaps are successfully tackled by framing the problem in a multi-frame setting.

By the modularity of the approach, it is possible to replace individual parts with faster and more robust deep modules. The dense, learned feature representation (SDC) brings great improvement not just for scene flow, but also for other dense matching tasks on a broad range of data sets and domains. While the computation on full resolution is necessary to obtain well-localized dense features, it limits the run time. Yet, compared to a dense heuristic description, it is still faster. The insights of the extensive evaluation study for the training of SDC push the overall performance of the descriptor even further and increase the invariance (especially to changes in the spatial scale space). The deep module for sparse-to-dense interpolation erases a number of previously necessary preconditions. Other than the raw image for guidance, no derived information needs to be pre-computed (e.g. edges, boundaries, semantic regions), and local assumptions on planarity or rigidity are removed. However, the accuracy of state-of-the-art is not reached in all cases. Instead, the deep interpolation module has great properties in terms of robustness. It shows an extreme invariance to noise and the distribution or density of the sparse input. The architecture is therefore useful in a wide range of applications, including the (guided) densification of less accurate, low-resolution LiDAR sensors. For both modules, the success on scene flow data – despite the limited availability of the same – is achieved by considering additional data of related problems during the design and training.

The sparse-to-dense concept is further applied together with the combination approach to obtain dense scene flow from auxiliary results. The two strategies together yield a very flexible and modular pipeline, that can be easily tuned in favor of any primary target, e.g. real-time performance. Most interestingly, the proposed pipeline is used for a similarly easy computation of dense scene flow with a monocular camera. To this end, depth estimation from single images is exploited, which is at the same time the limiting factor of the accuracy. Still, for sub-second run times this strategy sets the new state-of-the-art.

The rise of large synthetic training data for a multitude of computer vision problems has enabled a strategy to learn the estimation of dense scene flow from data in end-to-end networks. For the actual application scenario, these networks are adjusted by transfer learning and fine-tuning from the synthetic pre-trained domain to the realistic domain. The proposed PWOC-3D follows this strategy as one of the first networks for scene flow estimation. Two other end-to-end trainable models have been proposed for the frequently occurring issues of feature extraction and occlusions. The refined feature extraction of ResFPN introduces a more localized, dense multi-scale representation of images that improves diverse dense matching networks without much computational overhead. The deep multi-frame framework models occlusions as the main source of errors and thereby improves any dual-frame results, especially in occluded areas. The overall contribution answers the initially posed questions:

1. The sparse-to-dense concept can be successfully applied to the problem of scene flow estimation and is with this regard advantageous in terms of model complexity and generalization. However, the run times of classical matching algorithms are not yet fast enough for real-time applications, and the overall accuracy is limited by the ratio of sparse and missing information. Also, to obtain non-dense matches a consistency check is necessary. This doubles the matching effort and imposes only weak guarantees on the reliability of the sparse matches.

2. For data-driven approaches, it can be clearly stated that the relation of the scene flow problem to other tasks in computer vision allows to exploit multiple sources of data to model sub-solutions that contribute to the overall problem. It is also possible to model and train scene flow networks end-to-end on the limited amount of annotated data, however in the general case without further assumptions, the accuracy does not surpass traditional approaches. Run times of highly parallelized models on Graphics Processing Units (GPUs) are superior, though.

### Possible Next Steps

To immediately continue the work on the ideas of this thesis, overcome the identified limitations, and improve the results, the following propositions are made. Feature extraction and matching could be extended by a mechanism for uncertainty estimation that indicates the reliability of the matching result in order to replace the consistency check. While the matching score is a proxy

of this, it alone is insufficient for accurate filtering. Like feature extraction and interpolation, more parts of the pipeline could be replaced by dedicated deep modules, e.g. for (semantic/motion) boundary detection, or ego-motion estimation. The interpolation network could be split into a two-stage model (geometry and motion) that fully operates in 3D space, similar to the heuristic interpolation models (EPIC3D and RIC3D). For the monocular setting, depth estimation is not limited to the use of single images. Instead, two- or multi-view depth reconstruction should be used and further developed. More suggestions have been made during the discussion of the results in each chapter.

### Long Term Directions

Considering a longer time frame, based on the results and observations in this thesis, the development of novel scene flow methods should consider the following. While a dense (in terms of image resolution) representation and perception of the environment is rich, powerful, and certainly favorably for the geometry, semantics, and more, it is questionable whether this is also required for motion. Even for highly dynamic environments, (large) parts of the scene are static and the relevant motion is limited to dynamic behavior of objects. It should therefore be considered whether motion estimation (in 3D) could be reformulated together with the detection/segmentation of relevant objects. Even if a clustering of objects is not desired, the question arises whether (scene) flow should be estimated for non-occluded parts of the scene only. This suggestion gains importance when frame rates should increase and occluded areas start to vanish in the visual measurement.

Independent of that, the availability of more and more diverse sensors in the automotive context motivates to use and fuse more of them for the estimation of scene flow. E.g. the fusion of LiDAR and camera could erase the mutual disadvantages (accuracy versus density). In terms of input sensors, it would be also interesting to investigate the other extreme and try to solve single image scene flow estimation. While single image depth cues have already been exploited by deep neural networks, information about motion can also be encoded in a single image, e.g. by motion blur. Such an approach would not be affected by occlusions.

With respect to the training of deep neural networks, the lack of large amounts of labels can be compensated by self- and semi-supervised training strategies. This has to be used a lot more in the future and mixed with supervised training (i.e. to build semi-supervised strategies). Also important are recent techniques for domain adaptation to obtain domain invariance, especially for end-to-end trainable networks. This is an emerging research field which upon reaching maturity should find its way into the field of scene flow estimation.

# Acronyms

| | |
|---|---|
| **ADAS** | Advanced Driver Assistance System |
| **BCA** | Brightness Constancy Assumption |
| **BRIEF** | Binary Robust Independent Elementary Features |
| **CNN** | Convolutional Neural Network |
| **CRF** | Conditional Random Field |
| **DNN** | Deep Neural Network |
| **DTF** | Deep Temporal Fusion |
| **ELU** | Euclidean Linear Unit |
| **FLOPs** | Floating Point Operations |
| **FPN** | Feature Pyramid Network |
| **FT3D** | FlyingThings3D |
| **GPU** | Graphics Processing Unit |
| **LiDAR** | Light Detection and Ranging |
| **OE** | Optical Expansion |
| **ORB** | Oriented FAST and Rotated BRIEF |
| **OSF** | Object Scene Flow |
| **PnP** | Perspective-n-Point |
| **PRSF** | Piece-wise Rigid Scene Flow |
| **PRSM** | Piece-wise Rigid Scene Model |
| **PWOC-3D** | Pyramid, Warping, Occlusions, and Cost Correlation for 3D Scene Flow |
| **ReLU** | Rectified Linear Unit |
| **ResFPN** | Residual Feature Pyramid Network |
| **SDC** | Stacked Dilated Convolution |
| **SED** | Structured Edge Detection |
| **SFF** | SceneFlowFields |
| **SFF++** | SceneFlowFields++ |
| **SGM** | Semi-Global Matching |

| | |
|---|---|
| **SIFT** | Scale-Invariant Feature Transform |
| **SOR** | Successive Over-Relaxation |
| **SPS** | Slanted Plane Stereo |
| **SSGP** | Sparse Spatial Guided Propagation |
| **SURF** | Speeded-Up Robust Features |
| **WHT** | Walsh-Hadamard-Transform |

# List of Figures

# List of Tables

# Bibliography


[APTM20] Filippo Aleotti, Matteo Poggi, Fabio Tosi, and Stefano Mattoccia. "Learning end-to-end scene flow by distilling single tasks knowledge." In: *Conference on Artificial Intelligence (AAAI)*. 2020 21, 24, 28, 158, 160, 165–167, 212.

[BAM19] Fabian Brickwedde, Steffen Abraham, and Rudolf Mester. "Mono-SF: Multi-view geometry meets single-view depth for monocular scene flow estimation of dynamic traffic scenes." In: *International Conference on Computer Vision (ICCV)*. 2019 18, 24, 122, 124, 127, 128, 208, 209, 212.

[BBPW04] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. "High accuracy optical flow estimation based on a theory for warping." In: *European Conference on Computer Vision (ECCV)* (2004) 34.

[BJMA+17] Aseem Behl, Omid Hosseini Jafari, Siva Karthik Mustikovela, Hassan Abu Alhaija, Carsten Rother, and Andreas Geiger. "Bounding Boxes, Segmentations and Object Coordinates: How Important is Recognition for 3D Scene Flow Estimation in Autonomous Driving Scenarios?" In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 20, 22, 24, 26, 54, 55, 118, 120, 144, 212.

[BJTM16] Vassileios Balntas, Edward Johns, Lilian Tang, and Krystian Mikolajczyk. "PN-Net: Conjoined Triple Deep Network for Learning Local Image Descriptors." In: *arXiv preprint arXiv:1601.05030* (2016) 69, 71.

[BM11] Thomas Brox and Jitendra Malik. "Large displacement optical flow: descriptor matching in variational motion estimation." In: *Transactions on Pattern Analysis and Machine Intelligence* (2011) 118, 120, 212.

[BMK13] Tali Basha, Yael Moses, and Nahum Kiryati. "Multi-view scene flow estimation: A view centered variational approach." In: *International Journal of Computer Vision (IJCV)* (2013) 20.

[BPDG19] Aseem Behl, Despoina Paschalidou, Simon Donné, and Andreas Geiger. "PointFlowNet: Learning representations for rigid motion estimation from point clouds." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 19, 24.

[BSLR+11] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. "A Database and Evaluation Methodology for Optical Flow." In: *International Journal of Computer Vision (IJCV)* (2011) 14, 63, 66, 76–79, 83, 91, 92.

[BSWR+19] Ramy Battrawy, René Schuster, Oliver Wasenmüller, Qing Rao, and Didier Stricker. "LiDAR-Flow: Dense Scene Flow Estimation from Sparse LiDAR and Stereo Images." In: *International Conference on Intelligent Robots and Systems (IROS)*. 2019 19, 173.

[BTS15] Christian Bailer, Bertram Taetz, and Didier Stricker. "Flow Fields: Dense correspondence fields for highly accurate large displacement optical flow estimation." In: *International Conference on Computer Vision (ICCV)*. 2015 22, 29–31, 42, 80, 95, 103, 109, 115, 146.

[BTS19] Christian Bailer, Bertram Taetz, and Didier Stricker. "Flow Fields: Dense Correspondence Fields for Highly Accurate Large Displacement Optical Flow Estimation." In: *Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2019) 22, 31, 95, 103, 107–109, 115, 119.

[BTV06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded Up Robust Features." In: *European Conference on Computer Vision (ECCV)*. 2006 63.

[BVS17] Christian Bailer, Kiran Varanasi, and Didier Stricker. "CNN-based patch matching for optical flow with thresholded hinge embedding loss." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 22, 62, 64, 66, 68, 69, 71–73, 83, 115, 145.

[BWSB12] Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. "A naturalistic open source movie for optical flow evaluation." In: *European Conference on Computer Vision (ECCV)*. 2012 13, 20, 44, 47, 55–57, 63, 66, 75–79, 83, 91, 92, 101, 107, 146, 153, 154, 156.

[CC18] Jia-Ren Chang and Yong-Sheng Chen. "Pyramid stereo matching network." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 145–147, 151, 153–155, 158.

[CGSC16] Christopher B. Choy, JunYoung Gwak, Silvio Savarese, and Manmohan Chandraker. "Universal correspondence network." In: *Advances in Neural Information Processing Systems (NIPS)*. 2016 146.

[CHL05] Sumit Chopra, Raia Hadsell, and Yann LeCun. "Learning a Similarity Metric Discriminatively with Application to Face Verification." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2005 63.

[Cho17] François Chollet. "Xception: Deep Learning with Depthwise Separable Convolutions." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 100.

[CK16] Qifeng Chen and Vladlen Koltun. "Full Flow: Optical flow estimation by global optimization over regular grids." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 20.

[CLSF10] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. "BRIEF: Binary Robust Independent Elementary Features." In: *European Conference on Computer Vision (ECCV)*. 2010 63, 71, 72, 93, 94.

[CMH20] Yohann Cabon, Naila Murray, and Martin Humenberger. "Virtual KITTI 2." In: *arXiv preprint arXiv:2001.10773* (2020) 13, 14.

[CPKM+18] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs." In: *Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2018) 65.

[ČSH11] Jan Čech, Jordi Sanchez-Riera, and Radu Horaud. "Scene flow estimation by growing correspondence seeds." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2011 212.

[CUH16] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)." In: *International Conference on Learning Representations (ICLR)*. 2016 65, 71, 89.

[CWGY20] Xinjing Cheng, Peng Wang, Chenye Guan, and Ruigang Yang. "CSPN++: Learning Context and Resource Aware Convolutional Spatial Propagation Networks for Depth Completion." In: *Conference on Artificial Intelligence (AAAI)* (2020) 110.

[CWY18] Xinjing Cheng, Peng Wang, and Ruigang Yang. "Depth estimation via affinity learned with convolutional spatial propagation network." In: *European Conference on Computer Vision (ECCV)*. 2018 95, 96, 99, 101, 110.

[CYLU19] Yun Chen, Bin Yang, Ming Liang, and Raquel Urtasun. "Learning Joint 2D-3D Representations for Depth Completion." In: *International Conference on Computer Vision (ICCV)*. 2019 110.

[DFIH+15] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. "FlowNet: Learning optical flow with convolutional networks." In: *International Conference on Computer Vision (ICCV)*. 2015 21, 82, 103, 132, 134, 138, 142, 146, 153.

[DGZ17] Qi Dong, Shaogang Gong, and Xiatian Zhu. "Class rectification hard mining for imbalanced deep learning." In: *International Conference on Computer Vision (ICCV)*. 2017 86.

[DPSL16] Maxime Derome, Aurelien Plyer, Martial Sanfourche, and Guy Le Besnerais. "A prediction-correction approach for real-time optical flow computation using stereo." In: *German Conference on Pattern Recognition (GCPR)*. 2016 54, 120, 212.

[DT05] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2005 146.

[DZ13] Piotr Dollár and C. Lawrence Zitnick. "Structured forests for fast edge detection." In: *International Conference on Computer Vision (ICCV)*. 2013 31–33, 44, 45, 49, 50, 55.

[EFK19] Abdelrahman Eldesokey, Michael Felsberg, and Fahad Shahbaz Khan. "Confidence Propagation Through CNNs for Guided Sparse Depth Regression." In: *Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2019) 96, 110.

[FGWB+18] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. "Deep ordinal regression network for monocular depth estimation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 120, 122.

[FRRR+14] David Ferstl, Christian Reinbacher, Gernot Riegler, Matthias Rüther, and Horst Bischof. "aTGV-SF: Dense Variational Scene Flow through Projective Warping and Higher Order Regularization." In: *International Conference on 3D Vision (3DV)*. 2014 20.

[GB10] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2010 89.

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks." In: *International Conference on Artificial Intelligence and Statistics (AIStats)*. 2011 103.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012 13–15, 44, 83, 96, 101, 111, 118, 123, 126, 153, 156, 165.

[GMB17] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. "Unsupervised monocular depth estimation with left-right consistency." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 120, 122.

[GMFB19] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. "Digging into self-supervised monocular depth estimation." In: *International Conference on Computer Vision (ICCV)*. 2019 120, 122, 127, 128.

[GRU10] Andreas Geiger, Martin Roser, and Raquel Urtasun. "Efficient Large-scale Stereo Matching." In: *Asian Conference on Computer Vision (ACCV)*. 2010 75–77.

[GS03] David Gibson and Michael Spann. "Robust optical flow estimation based on a sparse motion trajectory set." In: *Transactions on Image Processing (TIP)* (2003) 95, 96.

[GW16] David Gadot and Lior Wolf. "Patchbatch: A Batch Augmented Loss for Optical Flow." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 64, 66, 71, 72.

[GWWL+19] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. "HPLFlownNet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 19.

[HA15] Elad Hoffer and Nir Ailon. "Deep Metric Learning Using Triplet Network." In: *International Workshop on Similarity-Based Pattern Recognition (SIMBAD)*. 2015 68, 69, 71.

[HD07] Frédéric Huguet and Frédéric Devernay. "A variational method for scene flow estimation from stereo sequences." In: *International Conference on Computer Vision (ICCV)*. 2007 12, 20, 24, 29, 34, 118, 212.

[HF16] Alexander Hermans and Georgios Floros. *No Title*. https://omnomnom.vision.rwth-aachen.de/data/rwth_kitti_semantics_dataset.zip. 2016 32.

[HFR14] Michael Hornacek, Andrew Fitzgibbon, and Carsten Rother. "SphereFlow: 6 DoF scene flow from RGB-D pairs." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014 19–21, 24, 54, 115, 120, 212.

[HH05] Yacov Hel-Or and Hagit Hel-Or. "Real-time pattern matching using projection kernels." In: *Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2005) 28.

[Hir08] Heiko Hirschmuller. "Stereo processing by semiglobal matching and mutual information." In: *Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2008) 20, 21, 31, 42, 54, 76, 77, 115, 118–120, 206, 207.

[HLJS+15] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. "MatchNet: Unifying Feature and Metric Learning for Patch-based Matching." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015 63.

[HLS17] Yinlin Hu, Yunsong Li, and Rui Song. "Robust interpolation of correspondences for large displacement optical flow." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 22, 42, 76, 80, 95, 96, 103, 107–109.

[HLVW17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. "Densely Connected Convolutional Networks." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 148.

[HR17] Junhwa Hur and Stefan Roth. "MirrorFlow: Exploiting symmetries in joint optical flow and occlusion estimation." In: *International Conference on Computer Vision (ICCV)*. 2017 133, 138.

[HR20] Junhwa Hur and Stefan Roth. "Self-Supervised Monocular Scene Flow Estimation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 18, 24, 122, 127, 128, 165, 209, 212.

[HRF13] Evan Herbst, Xiaofeng Ren, and Dieter Fox. "RGB-D Flow: Dense 3-D Motion Estimation Using Color and Depth." In: *International Conference on Robotics and Automation (ICRA)*. 2013 19, 20, 24, 29.

[HS12] Kaiming He and Jian Sun. "Computing nearest-neighbor fields via propagation-assisted kD-trees." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012 30, 103.

[HS81] Berthold K. P. Horn and Brian G. Schunck. "Determining optical flow." In: *Artificial Intelligence* (1981) 11, 20.

[HSL16] Yinlin Hu, Rui Song, and Yunsong Li. "Efficient coarse-to-fine patchmatch for large displacement optical flow." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 22, 36, 42, 75–77, 80, 95, 103, 107, 109, 146.

[HTC18] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. "LiteFlowNet: A lightweight convolutional neural network for optical flow estimation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 145, 146, 153, 154, 156–158.

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2003 10.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." In: *International Conference on Computer Vision (CVPR)*. 2015 89.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 148.

[IMSK+17] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. "FlowNet 2.0: Evolution of optical flow estimation with deep networks." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 21, 82, 101, 132, 146.

[ISKB18] Eddy Ilg, Tonmoy Saikia, Margret Keuper, and Thomas Brox. "Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation." In: *European Conference on Computer Vision (ECCV)*. 2018 21, 134, 142, 144.

[JDWP+18] Maximilian Jaritz, Raoul De Charette, Emilie Wirbel, Xavier Perrotton, and Fawzi Nashashibi. "Sparse and dense data with CNNs: Depth completion and semantic segmentation." In: *International Conference on 3D Vision (3DV)*. 2018 95, 110.

[JSGC15] Mariano Jaimez, Mohamed Souiai, Javier Gonzalez-Jimenez, and Daniel Cremers. "A primal-dual framework for real-time dense RGB-D scene flow." In: *International Conference on Robotics and Automation (ICRA)*. 2015 19, 20, 24, 115.

[JSJL+19] Huaizu Jiang, Deqing Sun, Varun Jampani, Zhaoyang Lv, Erik Learned-Miller, and Jan Kautz. "Sense: A shared encoder network for scene-flow estimation." In: *International Conference on Computer Vision (ICCV)*. 2019 22, 24, 28, 158, 160, 165–167, 171, 212.

[KB15] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *International Conference for Learning Representations (ICLR*. 2015 69, 103, 142, 165.

[KNHK+16] Daniel Kondermann, Rahul Nair, Katrin Honauer, Karsten Krispin, Jonas Andrulis, Alexander Brock, Burkhard Gussefeld, Mohsen Rahimimoghaddam, Sabine Hofmann, Claus Brenner, et al. "The HCI Benchmark Suite: Stereo and Flow Ground Truth with Uncertainties for Urban Autonomous Driving." In: *Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2016 63, 66, 76–79, 83, 91, 92, 101, 107.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in Neural Information Processing Systems (NIPS)*. 2012 148.

[LBAL+16] Zhaoyang Lv, Chris Beall, Pablo F. Alcantarilla, Fuxin Li, Zsolt Kira, and Frank Dellaert. "A continuous optimization approach for efficient and accurate scene flow." In: *European Conference on Computer Vision (ECCV)*. 2016 20, 22, 24, 26, 29, 51, 54, 118, 120, 144, 212.

[LBBH+98] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* (1998) 148.

[LDGH+17] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. "Feature Pyramid Networks for Object Detection." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 134, 135, 145, 146, 148, 149, 151, 154, 155, 157.

[LDGZ+17] Sifei Liu, Shalini De Mello, Jinwei Gu, Guangyu Zhong, Ming-Hsuan Yang, and Jan Kautz. "Learning affinity via spatial propagation networks." In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017 96.

[LHAY16] Yijun Li, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. "Deep joint image filtering." In: *European Conference on Computer Vision (ECCV)*. 2016 96.

[LHKS19] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. "From big to small: Multi-scale local planar guidance for monocular depth estimation." In: *arXiv preprint arXiv:1907.10326* (2019) 120, 122, 123, 126, 127, 129.

[LK81] Bruce D. Lucas and Takeo Kanade. "An iterative image registration technique with an application to stereo vision." In: (1981) 11.

[LLKK20] Sihaeng Lee, Janghyeon Lee, Doyeon Kim, and Junmo Kim. "Deep Architecture With Cross Guidance Between Single Image and Sparse LiDAR Data for Depth Completion." In: *IEEE Access* (2020) 110.

[LLKX19] Pengpeng Liu, Michael Lyu, Irwin King, and Jia Xu. "SelFlow: Self-supervised learning of optical flow." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 160.

[LLYL+18] Fengmao Lv, Qing Lian, Guowu Yang, Guosheng Lin, Sinno Jialin Pan, and Lixin Duan. "Domain Adaptive Semantic Segmentation Through Structure Enhancement." In: *European Conference on Computer Vision (ECCV)*. 2018 21.

[LML21] Congcong Li, Haoyu Ma, and Qingmin Liao. "Two-stage adaptive object scene flow using hybrid CNN-CRF model." In: *International Conference on Pattern Recognition (ICPR)*. 2021 20, 22, 24, 212.

[Low99] David G. Lowe. "Object recognition from local scale-invariant features." In: *International Conference on Computer Vision (ICCV)*. 1999 29, 62, 63, 71, 72, 77, 78, 80, 93, 94, 146.

[LQG19] Xingyu Liu, Charles R. Qi, and Leonidas J Guibas. "FlowNet3D: Learning Scene Flow in 3D Point Clouds." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 19, 24.

[LRSW+18] Guilin Liu, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. "Image inpainting for irregular holes using partial convolutions." In: *European Conference on Computer Vision (ECCV)*. 2018 97.

[LSU16] Wenjie Luo, Alexander G. Schwing, and Raquel Urtasun. "Efficient Deep Learning for Stereo Matching." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 64.

[LWAS+12] Manuel Lang, Oliver Wang, Tunc Aydin, Aljoscha Smolic, and Markus Gross. "Practical temporal consistency for image-based graphics applications." In: *Transactions on Graphics (ToG)* (2012) 95.

[LYLC+20] Ang Li, Zejian Yuan, Yonggen Ling, Wanchao Chi, Chong Zhang, et al. "A Multi-Scale Guided Cascade Hourglass Network for Depth Completion." In: *Winter Conference on Applications of Computer Vision (WACV)*. 2020 110.

[LYT11] Ce Liu, Jenny Yuen, and Antonio Torralba. "SIFT Flow: Dense correspondence across scenes and its applications." In: *Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2011) 29, 81.

[LZC18] Yuhong Li, Xiaofan Zhang, and Deming Chen. "CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 65.

[MB18] Daniel Maurer and Andrés Bruhn. "ProFlow: Learning to Predict Optical Flow." In: *British Machine Vision Conference (BMVC)*. 2018 160, 161.

[MCK19] Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. "Self-Supervised Sparse-to-Dense: Self-supervised Depth Completion from LiDAR and Monocular Camera." In: *International Conference on Robotics and Automation (ICRA)*. 2019 95, 110.

[MFTM01] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics." In: *International Conference on Computer Vision (ICCV)*. 2001 44, 46.

[MG15] Moritz Menze and Andreas Geiger. "Object scene flow for autonomous vehicles." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015 13–15, 20, 22, 24, 26, 29, 35, 46, 50–55, 63, 66, 68, 75–79, 81, 83, 92, 101, 107, 114, 118, 120, 123, 126, 142, 144, 146, 153, 154, 156, 158, 165, 171, 212.

[MHG15] Moritz Menze, Christian Heipke, and Andreas Geiger. "Discrete Optimization for Optical Flow." In: *German Conference on Pattern Recognition (GCPR)*. 2015 109.

[MHG18] Moritz Menze, Christian Heipke, and Andreas Geiger. "Object Scene Flow." In: *Journal of Photogrammetry and Remote Sensing (JPRS)* (2018) 53, 54, 120, 144, 212.

[MHN13] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. "Rectifier nonlinearities improve neural network acoustic models." In: *International Conference on Machine Learning (ICML)*. 2013 139, 151, 165.

[MHR18] Simon Meister, Junhwa Hur, and Stefan Roth. "UnFlow: Unsupervised Learning of Optical Flow with a Bidirectional Census Loss." In: *Conference on Artifical Intelligence (AAAI)*. 2018 133, 138.

[MIFH+18] Nikolaus Mayer, Eddy Ilg, Philipp Fischer, Caner Hazirbas, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. "What makes good synthetic training data for learning disparity and optical flow estimation?" In: *International Journal of Computer Vision (IJCV)* (2018) 82, 91, 165.

[MIHF+16] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 13, 21, 83, 92, 101, 103, 134, 142, 144, 146, 153, 154, 164, 165, 170.

[MK19] Michael Meyer and Georg Kuschk. "Deep learning based 3d object detection for automotive radar and camera." In: *European Radar Conference (EuRAD)*. 2019 173.

[MOH20] Himangi Mittal, Brian Okorn, and David Held. "Just go with the flow: Self-supervised scene flow estimation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 19.

[MSB17] Daniel Maurer, Michael Stoll, and Andrés Bruhn. "Order-Adaptive and Illumination-Aware Variational Optical Flow Refinement." In: *British Machine Vision Conference (BMVC)*. 2017 23.

[MWHX+19] Wei-Chiu Ma, Shenlong Wang, Rui Hu, Yuwen Xiong, and Raquel Urtasun. "Deep Rigid Instance Scene Flow." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 20–22, 24, 28, 158, 160, 212.

[NH10] Vinod Nair and Geoffrey E Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines." In: *International Conference on Machine Learning (ICML)*. 2010 71.

[NHTSA17] National Highway Traffic Safety Administration et al. "Automated driving systems 2.0: A vision for safety." In: *Washington, DC: US Department of Transportation, DOT HS* (2017) 1.

[NM03] Mircea Nicolescu and Gérard Medioni. "Layered 4D representation and voting for grouping from motion." In: *Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2003) 95, 96.

[NŠ17] Michal Neoral and Jan Šochman. "Object Scene Flow with Temporal Consistency." In: *Computer Vision Winter Workshop (CVWW)*. 2017 20, 22, 26, 54, 144, 160, 171, 212.

[NŠM18] Michal Neoral, Jan Šochman, and Jiří Matas. "Continual occlusion and optical flow estimation." In: *Asian Conference on Computer Vision (ACCV)*. 2018 160.

[OKM19] Matthias Ochs, Adrian Kretz, and Rudolf Mester. "SDNet: Semantically guided depth estimation network." In: *German Conference on Pattern Recognition (GCPR)*. 2019 120.

[ORBH18] Maria Oliver, Lara Raad, Coloma Ballester, and Gloria Haro. "Motion inpainting by an image-based geodesic AMLE method." In: *International Conference on Image Processing (ICIP)*. 2018 95.

[PKF05] J-P Pons, Renaud Keriven, and Olivier Faugeras. "Modelling dynamic scenes by registering multi-view image sequences." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2005 19, 24.

[PKF07] Jean-Philippe Pons, Renaud Keriven, and Olivier Faugeras. "Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score." In: *International Journal of Computer Vision (IJCV)* (2007) 19.

[POJT+12] Jaesik Park, Tae Hyun Oh, Jiyoung Jung, Yu-Wing Tai, and In So Kweon. "A tensor voting approach for multi-view 3D scene flow estimation and refinement." In: *European Conference on Computer Vision (ECCV)*. 2012 20.

[QBDC14] Julian Quiroga, Thomas Brox, Frédéric Devernay, and James Crowley. "Dense semi-rigid scene flow estimation from rgbd images." In: *European Conference on Computer Vision (ECCV)*. 2014 19, 20, 24.

[QCZZ+19] Jiaxiong Qiu, Zhaopeng Cui, Yinda Zhang, Xingdi Zhang, Shuaicheng Liu, Bing Zeng, and Marc Pollefeys. "DeepLiDAR: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 110.

[RB17] Anurag Ranjan and Michael J Black. "Optical flow estimation using a spatial pyramid network." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 134, 146.

[RBSW+20] Rishav, Ramy Battrawy, René Schuster, Oliver Wasenmüller, and Didier Stricker. "DeepLiDARFlow: A Deep Learning Architecture For Scene Flow Estimation Using Monocular Camera and Sparse LiDAR." In: *International Conference on Intelligent Robots and Systems (IROS)*. 2020 19.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." In: *International Conference on Medical Image Computing and Computer-assisted Intervention (MICCAI)*. 2015 97, 148.

[RGSY+19] Zhile Ren, Orazio Gallo, Deqing Sun, Ming-Hsuan Yang, Erik Sudderth, and Jan Kautz. "A fusion approach for multi-frame optical flow estimation." In: *Winter Conference on Applications of Computer Vision (WACV)*. 2019 38, 160.

[RJGB19] Anurag Ranjan, Joel Janai, Andreas Geiger, and Michael J Black. "Attacking Optical Flow." In: *International Conference on Computer Vision (ICCV)*. 2019 148, 173.

[RKVT16] Christian Richardt, Hyeongwoo Kim, Levi Valgaerts, and Christian Theobalt. "Dense wide-baseline scene flow from two handheld video cameras." In: *International Conference on 3D Vision (3DV)*. 2016 212.

[RRGB+15] German Ros, Sebastian Ramos, Manuel Granados, Amir Bakhtiary, David Vazquez, and Antonio M. Lopez. "Vision-based Offline-Online Perception Paradigm for Autonomous Driving." In: *Winter Conference on Applications of Computer Vision (WACV)*. 2015 32.

[RRKB11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. "ORB: An Efficient Alternative to SIFT or SURF." In: *International Conference on Computer Vision (ICCV)*. 2011 63.

[RSBW+21] Rishav, René Schuster, Ramy Battrawy, Oliver Wasenmüller, and Didier Stricker. "ResFPN: Residual Skip Connections in Multi-Resolution Feature Pyramid Networks for Accurate Dense Pixel Matching." In: *International Conference on Pattern Recognition (ICPR)*. 2021 145, 147.

[RSKS17] Zhile Ren, Deqing Sun, Jan Kautz, and Erik B Sudderth. "Cascaded Scene Flow Prediction using Semantic Segmentation." In: *International Conference on 3D Vision (3DV)*. 2017 21, 24, 51, 54, 55, 120, 144, 212.

[RWHS15] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. "EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015 22, 26, 31–33, 35, 42, 76, 95, 96, 103, 107–109, 115.

[SBWS18a] René Schuster, Christian Bailer, Oliver Wasenmüller, and Didier Stricker. "Combining Stereo Disparity and Optical Flow for Basic Scene Flow." In: *Commercial Vehicle Technology Symposium (CVT)*. 2018 119, 212.

[SBWS18b] René Schuster, Christian Bailer, Oliver Wasenmüller, and Didier Stricker. "FlowFields++: Accurate Optical Flow Correspondences Meet Robust Interpolation." In: *International Conference on Image Processing (ICIP)*. 2018 76, 78–80, 115, 119.

[SDBR15] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. "Striving for Simplicity: The All Convolutional Net." In: *International Conference on Learning Representations Workshops (ICLRW)*. 2015 64.

[SIVA17] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. "Inception-v4, Inception-ResNet and the impact of residual connections on learning." In: *Conference on Artificial Intelligence (AAAI)*. 2017 148.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015 86.

[SLJS+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015 148.

[SLR13] Robert Spangenberg, Tobias Langner, and Raúl Rojas. "Weighted Semi-Global Matching and Center-Symmetric Census Transform for Robust Driver Assistance." In: *International Conference on Computer Analysis of Images and Patterns (CAIP)*. 2013 76.

[SMG14] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks." In: *International Conference on Learning Representations (ICLR)*. 2014 89.

[SRB14] Deqing Sun, Stefan Roth, and Michael J Black. "A quantitative analysis of current practices in optical flow estimation and the principles behind them." In: *International Journal of Computer Vision (IJCV)* (2014) 118, 120, 206, 207, 212.

[SS02] Daniel Scharstein and Richard Szeliski. "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms." In: *International Journal of Computer Vision (IJCV)* (2002) 63, 66, 76, 77, 83, 91, 92.

[SSGS+17] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. "A Multi-View Stereo Benchmark with High-Resolution Images and Multi-Camera Videos." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 63, 66, 75–77, 83, 91, 92.

[SSMG16] Abhinav Shrivastava, Rahul Sukthankar, Jitendra Malik, and Abhinav Gupta. "Beyond skip connections: Top-down modulation for object detection." In: *arXiv preprint arXiv:1612.06851* (2016) 148.

[SSWS19] Rohan Saxena, René Schuster, Oliver Wasenmuller, and Didier Stricker. "PWOC-3D: Deep occlusion-aware end-to-end scene flow estimation." In: *Intelligent Vehicles Symposium (IV)*. 2019 101, 132, 145, 146, 151, 153–155, 157–161, 163, 165–169, 171, 172, 212.

[SUS20] René Schuster, Christian Unger, and Didier Stricker. "MonoComb: A Sparse-to-Dense Combination Approach for Monocular Scene Flow." In: *Computer Science in Cars Symposium (CSCS)*. 2020 122, 212.

[SUS21] René Schuster, Christian Unger, and Didier Stricker. "A deep temporal fusion framework for scene flow using a learnable motion model and occlusions." In: *Winter Conference on Applications of Computer Vision (WACV)*. 2021 158, 212.

[SWKB+18] René Schuster, Oliver Wasenmüller, Georg Kuschk, Christian Bailer, and Didier Stricker. "SceneFlowFields: Dense Interpolation of Sparse Scene Flow Correspondences." In: *Winter Conference on Applications of Computer Vision (WACV)*. 2018 41, 42, 47, 48, 52–54, 75, 76, 80, 81, 95, 96, 106, 107, 117, 119, 120, 144, 146, 166–168, 171, 206, 207, 212.

[SWS18] René Schuster, Oliver Wasenmüller, and Didier Stricker. "Dense Scene Flow from Stereo Disparity and Optical Flow." In: *Computer Science in Cars Symposium (CSCS)*. 2018 212.

[SWUK+20] René Schuster, Oliver Wasenmüller, Christian Unger, Georg Kuschk, and Didier Stricker. "SceneFlowFields++: Multi-frame Matching, Visibility Prediction, and Robust Interpolation for Scene Flow Estimation." In: *International Journal on Computer Vision (IJCV)* (2020) 47, 53, 54, 95, 96, 106, 107, 109, 110, 160, 163, 171, 212.

[SWUS19] René Schuster, Oliver Wasenmüller, Christian Unger, and Didier Stricker. "SDC - Stacked Dilated Convolution: A Unified Descriptor Network for Dense Matching Tasks." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 84, 93, 94, 135, 146.

[SWUS21] René Schuster, Oliver Wasenmüller, Christian Unger, and Didier Stricker. "SSGP: Sparse Spatial Guided Propagation for Robust and Generic Interpolation." In: *Winter Conference on Applications of Computer Vision (WACV)*. 2021 95, 110, 124, 125, 127, 129.

[SXLW19] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. "Deep high-resolution representation learning for human pose estimation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 148.

[SYLK18] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. "PWC-Net: CNNs for Optical flow using pyramid, warping, and cost volume." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 69, 82, 83, 101, 122, 133–135, 141, 142, 145, 146, 151, 153, 154, 156–158, 163, 165.

[SYLK19] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. "Models matter, so does training: An empirical study of CNNs for optical flow estimation." In: *Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2019) 82, 89, 165.

[SZ15] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." In: *International Conference on Learning Representations (ICLR)*. 2015 148.

[TFW+17] Yurun Tian, Bin Fan, Fuchao Wu, et al. "L2-Net: Deep Learning of Discriminative Patch Descriptor in Euclidean Space." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 62–64, 66, 71–73.

[TLF09] Engin Tola, Vincent Lepetit, and Pascal Fua. "Daisy: An efficient dense descriptor applied to wide-baseline stereo." In: *Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2009) 146.

[TLF10] Engin Tola, Vincent Lepetit, and Pascal Fua. "DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo." In: *Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2010) 63, 71, 72, 93, 94.

[TSS17] Tatsunori Taniai, Sudipta N. Sinha, and Yoichi Sato. "Fast Multi-frame Stereo Scene Flow with Motion Segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 20, 22, 24, 34, 51, 53–55, 144, 160, 171, 212.

[TTFL+20] Jie Tang, Fei-Peng Tian, Wei Feng, Jian Li, and Ping Tan. "Learning Guided Convolutional Network for Depth Completion." In: *Transactions on Image Processing (TIP)* (2020) 95, 97, 100, 105, 110.

[USSF+17] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. "Sparsity invariant CNNs." In: *International Conference on 3D Vision (3DV)*. 2017 95–97, 100, 101, 109, 110, 123.

[VBRC+99] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. "Three-dimensional scene flow." In: *International Conference on Computer Vision (ICCV)*. 1999 18, 19, 23, 24.

[VBZW+10] Levi Valgaerts, Andrés Bruhn, Henning Zimmer, Joachim Weickert, Carsten Stoll, and Christian Theobalt. "Joint estimation of motion, structure and geometry from stereo sequences." In: *European Conference on Computer Vision (ECCV)*. 2010 20.

[VNDV19] Wouter Van Gansbeke, Davy Neven, Bert De Brabandere, and Luc Van Gool. "Sparse and Noisy LiDAR Completion with RGB Guidance and Uncertainty." In: *International Conference on Machine Vision Applications (MVA)*. 2019 110.

[VRCK05] Sundar Vedula, Peter Rander, Robert Collins, and Takeo Kanade. "Three-dimensional scene flow." In: *Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2005) 122.

[VRS14] Christoph Vogel, Stefan Roth, and Konrad Schindler. "View-consistent 3D scene flow estimation over multiple frames." In: *European Conference on Computer Vision (ECCV)*. 2014 20, 22.

[VSR13] Christoph Vogel, Konrad Schindler, and Stefan Roth. "Piecewise rigid scene flow." In: *International Conference on Computer Vision (ICCV)*. 2013 20, 22, 24, 26, 29, 54, 115, 118, 120, 144, 171, 212.

[VSR15] Christoph Vogel, Konrad Schindler, and Stefan Roth. "3D Scene Flow Estimation with a Piecewise Rigid Scene Model." In: *International Journal of Computer Vision (IJCV)* (2015) 20, 22, 24, 26, 29, 51–55, 144, 160, 171, 212.

[WCYL+18] Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell. "Understanding Convolution for Semantic Segmentation." In: *Winter Conference on Applications of Computer Vision (WACV)*. 2018 65.

[WD18] Mei Wang and Weihong Deng. "Deep visual domain adaptation: A survey." In: *Neurocomputing (Neurocomputing)* (2018) 21.

[WHO18] World Health Organization et al. *Global status report on road safety 2018: Summary*. Tech. rep. World Health Organization, 2018 1.

[WKR17] Anne S. Wannenwetsch, Margret Keuper, and Stefan Roth. "ProbFlow: Joint Optical Flow and Uncertainty Estimation." In: *International Conference on Computer Vision (ICCV)*. 2017 22, 30.

[WRHS13] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. "DeepFlow: Large displacement optical flow with deep matching." In: *International Conference on Computer Vision (ICCV)*. 2013 103.

[WRVB+08] Andreas Wedel, Clemens Rabe, Tobi Vaudrey, Thomas Brox, Uwe Franke, and Daniel Cremers. "Efficient dense scene flow from sparse or dense stereo data." In: *European Conference on Computer Vision (ECCV)*. 2008 20, 24, 34.

[WWLL+20] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. "PointPWC-Net: Cost Volume on Point Clouds for (Self-) Supervised Scene Flow Estimation." In: *European Conference on Computer Vision (ECCV)*. 2020 19, 24.

[WYYZ+18] Yang Wang, Yi Yang, Zhenheng Yang, Liang Zhao, Peng Wang, and Wei Xu. "Occlusion aware unsupervised learning of optical flow." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 133, 138.

[WZLY+19] Xianshun Wang, Dongchen Zhu, Yanqing Liu, Xiaoqing Ye, Jiamao Li, and Xiaolin Zhang. "SemFlow: Semantic-Driven Interpolation for Large Displacement Optical Flow." In: *IEEE Access* (2019) 22, 95, 96.

[WZZH18] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. "Fast end-to-end trainable guided filter." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 96.

[XDBZ+16] Philippe Xu, Franck Davoine, Jean-Baptiste Bordes, Huijing Zhao, and Thierry Denœux. "Multimodal information fusion for urban scene understanding." In: *Machine Vision and Applications (MVA)* (2016) 32.

[XJM11] Li Xu, Jiaya Jia, and Yasuyuki Matsushita. "Motion detail preserving optical flow estimation." In: *Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2011) 146.

[XWLZ+18] Yongjian Xin, Shuhui Wang, Liang Li, Weigang Zhang, and Qingming Huang. "Reverse Densely Connected Feature Pyramid Network for Object Detection." In: *Asian Conference on Computer Vision (ACCV)*. 2018 148, 149.

[XZSZ+19] Yan Xu, Xinge Zhu, Jianping Shi, Guofeng Zhang, Hujun Bao, and Hongsheng Li. "Depth Completion from Sparse LiDAR Data with Depth-Normal Constraints." In: *International Conference on Computer Vision (ICCV)*. 2019 110.

[YDY19] Zhichao Yin, Trevor Darrell, and Fisher Yu. "Hierarchical discrete distribution decomposition for match density estimation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 123, 126, 127, 129.

[YMU14] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. "Efficient joint segmentation, occlusion labeling, stereo and flow estimation." In: *European Conference on Computer Vision (ECCV)*. 2014 42, 115, 119.

[YR19] Gengshan Yang and Deva Ramanan. "Volumetric correspondence networks for optical flow." In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019 123, 126, 127, 129.

[YR20] Gengshan Yang and Deva Ramanan. "Upgrading Optical Flow to 3D Scene Flow through Optical Expansion." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 18, 24, 122, 127–129, 158, 160, 166–168, 208, 209, 212.

[YWS19] Yanchao Yang, Alex Wong, and Stefano Soatto. "Dense Depth Posterior (DDP) from single image and sparse range." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 110.

[ZCYZ12] Xiaowei Zhang, Dapeng Chen, Zejian Yuan, and Nanning Zheng. "Dense scene flow based on depth and multi-channel bilateral filter." In: *Asian Conference on Computer Vision (ACCV)*. 2012 20.

[ZDMD+18] Ligeng Zhu, Ruizhi Deng, Michael Maire, Zhiwei Deng, Greg Mori, and Ping Tan. "Sparsely aggregated convolutional networks." In: *European Conference on Computer Vision (ECCV)*. 2018 145.

[ZK15] Sergey Zagoruyko and Nikos Komodakis. "Learning to Compare Image Patches via Convolutional Neural Networks." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015 63, 64, 71, 72.

[ZL15] Jure Zbontar and Yann LeCun. "Computing the Stereo Matching Cost with a Convolutional Neural Network." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015 62, 64, 96.

[ZLZC+19] Xiaotong Zhao, Wei Li, Yifan Zhang, Shuo Chang, Zhiyong Feng, and Ping Zhang. "Aggregated Residual Dilation-Based Feature Pyramid Network for Object Detection." In: *IEEE Access* (2019) 148.

[ZLZZ+19] Xiaotong Zhao, Wei Li, Yifan Zhang, Fan Zhang, Shuo Chang, and Zhiyong Feng. "Residual Dilation Based Feature Pyramid Network." In: *International Conference on Image Processing (ICIP)*. 2019 148.

[ZNMC+19] Yilun Zhang, Ty Nguyen, Ian D. Miller, Steven Chen, Camillo J. Taylor, Vijay Kumar, et al. "DFineNet: Ego-motion estimation and depth refinement from sparse, noisy depth input with rgb guidance." In: *arXiv preprint arXiv:1903.06397* (2019) 110.

[ZSDC+20] Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I-Chao Chang, and Yan Xu. "MaskFlownet: Asymmetric Feature Matching with Learnable Occlusion Mask." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 160.

[ZW17] Shay Zweig and Lior Wolf. "InterpoNet, a Brain Inspired Neural Network for Optical Flow Dense Interpolation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 22, 95, 96, 107–109.

[ZW94] Ramin Zabih and John Woodfill. "Non-parametric local transforms for computing visual correspondence." In: *European Conference on Computer Vision (ECCV)*. 1994 62, 63.

Appendix A

# Additional Visualizations

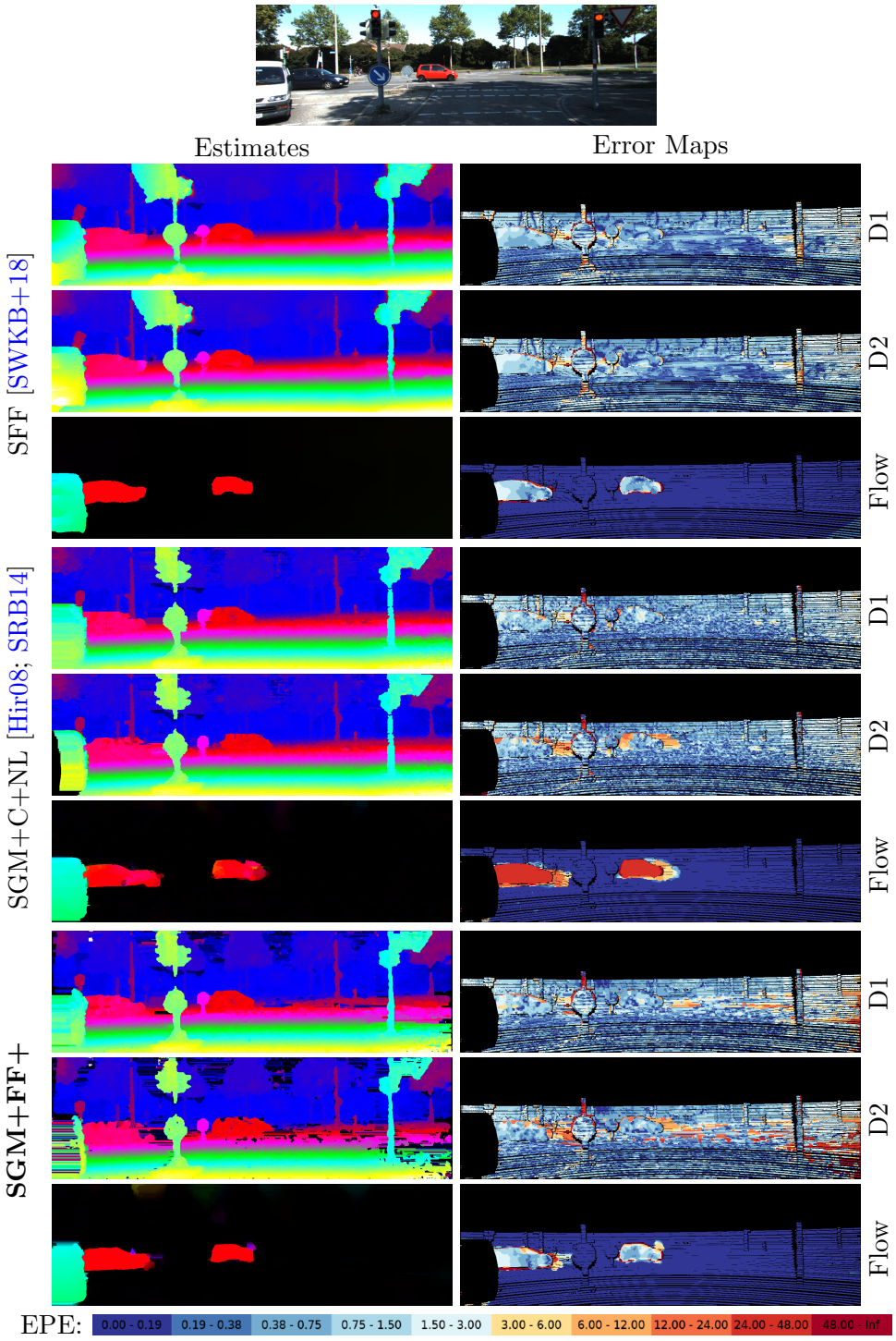



**Figure A.1.:** *Visual comparison of the results on KITTI test image 8 for SceneFlowFields (SFF) [SWKB+18] (Section 4.1), SGM+C+NL [Hir08; SRB14], and the proposed combination approach of Section 6.1.*

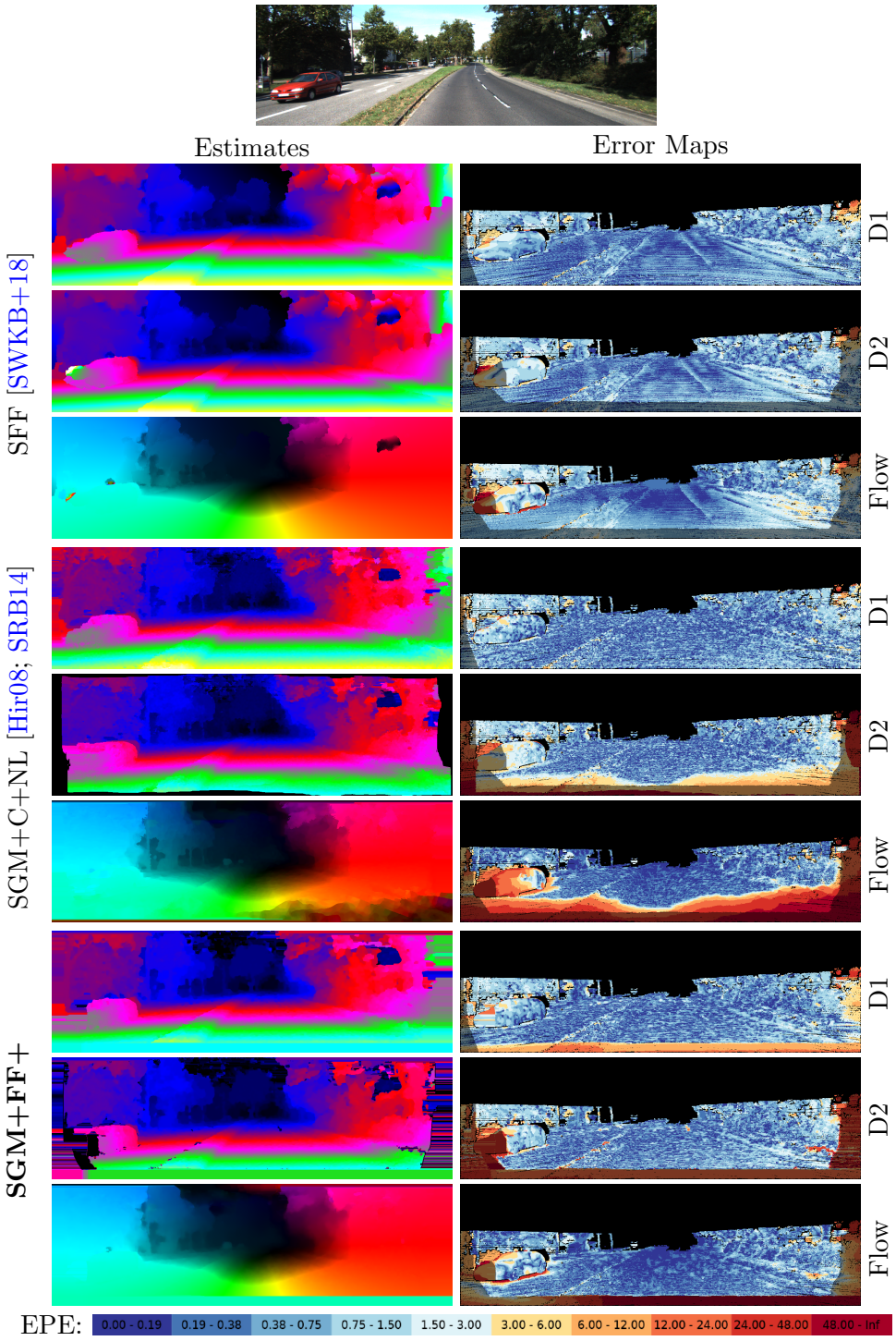
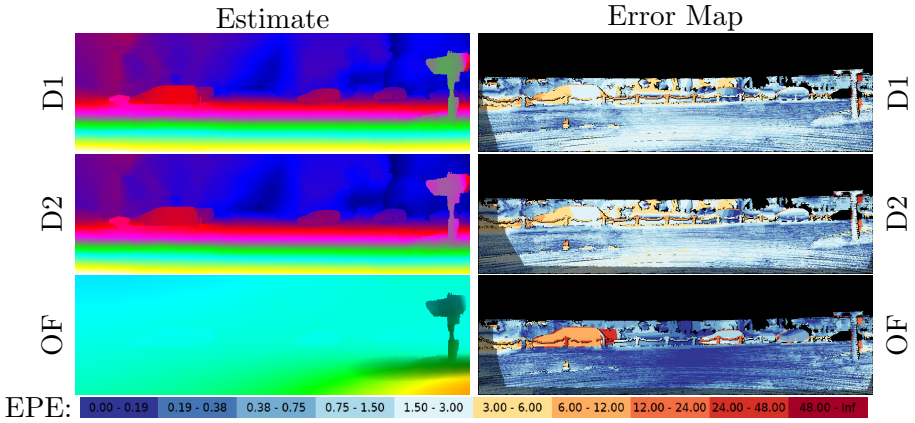
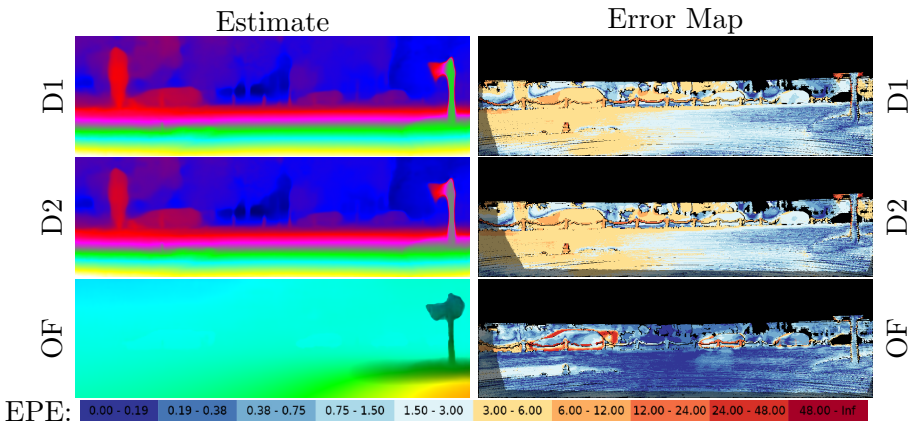



**Figure A.2.:** *Visual comparison of the results on KITTI test image 13 for SFF [SWKB+18] (Section 4.1), SGM+C+NL [Hir08; SRB14], and the proposed combination approach of Section 6.1.*

**(a)** *Reference Image*

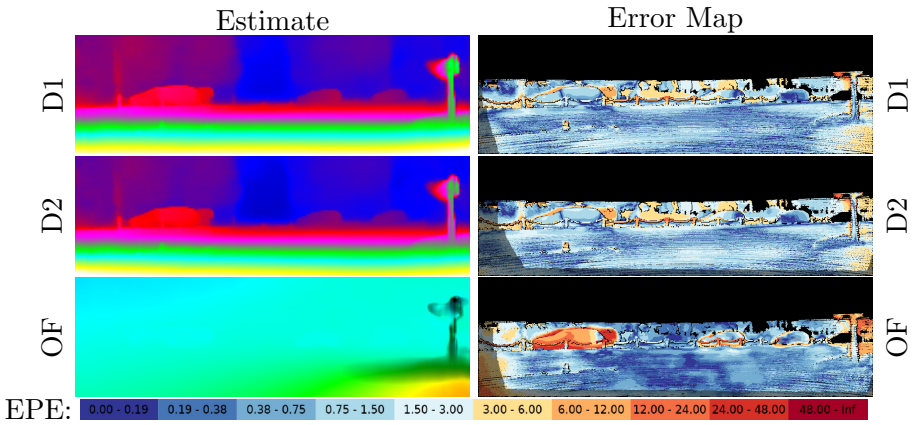



**(b)** *Result of Mono-SF [BAM19]*

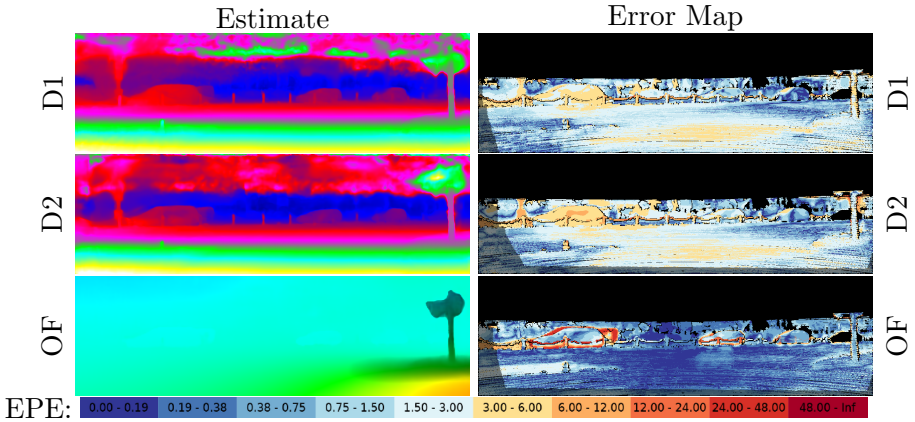



**(c)** *Result of MonoExpansion [YR20]*

**Figure A.3.:** *Continued on the next page.*




**(d)** *Result of Self-Mono-SF (fine-tuned) [HR20]*




**(e)** *Result of the monocular combination approach MonoComb.*

**Figure A.3.:** *Visualization and error maps from the KITTI online benchmark for the first test frame. Mono-SF [BAM19] (b), MonoExpansion [YR20] (c), Self-Mono-SF [HR20] (d), and the proposed MonoComb approach of Section 6.2 (e) are compared.*

Appendix B

# KITTI Scene Flow Leader Board

**Table B.1.:** *This table shows a snapshot of the KITTI scene flow leader board with all published, peer-reviewed methods at the time of writing (April 25, 2021).*

| Method | Reference | Year | Setting | D1 | | | D2 | | | OF | | | SF | | | Run |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | bg | fg | all | bg | fg | all | bg | fg | all | bg | fg | all | time [s] |
| DRISF | [MWHX+19] | 2019 | | 2.16 | 4.49 | 2.55 | 2.90 | 9.73 | 4.04 | 3.59 | 10.40 | 4.73 | 4.39 | 15.94 | 6.31 | 0.75 |
| ACOSF | [LML21] | 2021 | | 2.79 | 7.56 | 3.58 | 3.82 | 12.74 | 5.31 | 4.56 | 12.00 | 5.79 | 5.61 | 19.38 | 7.90 | 300 |
| ISF | [BJMA+17] | 2017 | | 4.12 | 6.17 | 4.46 | 4.88 | 11.34 | 5.95 | 5.40 | 10.29 | 6.22 | 6.58 | 15.63 | 8.08 | 600 |
| OpticalExpansion | [YR20] | 2020 | | 1.48 | 3.46 | 1.81 | 3.39 | 8.54 | 4.25 | 5.83 | 8.66 | 6.30 | 7.06 | 13.44 | 8.12 | 2 |
| PRSM | [VSR15] | 2015 | multi | 3.02 | 10.52 | 4.27 | 5.13 | 15.11 | 6.79 | 5.33 | 13.40 | 6.68 | 6.61 | 20.79 | 8.97 | 300 |
| DTF_SENSE | [SUS21] | 2021 | multi | 2.08 | 3.13 | 2.25 | 4.82 | 9.02 | 5.52 | 7.31 | 9.48 | 7.67 | 8.21 | 14.08 | 9.18 | 0.76 |
| OSF+TC | [NŠ17] | 2017 | | 4.11 | 9.64 | 5.03 | 5.18 | 15.12 | 6.84 | 5.76 | 13.31 | 7.02 | 7.08 | 20.03 | 9.23 | 3000 |
| SENSE | [JSJL+19] | 2019 | | 2.07 | 3.01 | 2.22 | 4.90 | 10.83 | 5.89 | 7.30 | 9.33 | 7.64 | 8.36 | 15.49 | 9.55 | 0.32 |
| OSF 2018 | [MHG18] | 2018 | | 4.11 | 11.12 | 5.28 | 5.01 | 17.28 | 7.06 | 5.38 | 17.61 | 7.41 | 6.68 | 24.59 | 9.66 | 390 |
| SSF | [RSKS17] | 2017 | | 3.55 | 8.75 | 4.42 | 4.94 | 17.48 | 7.02 | 5.63 | 14.71 | 7.14 | 7.18 | 24.58 | 10.07 | 300 |
| OSF | [MG15] | 2015 | | 4.54 | 12.03 | 5.79 | 5.45 | 19.41 | 7.77 | 5.62 | 18.92 | 7.83 | 7.01 | 26.34 | 10.23 | 3000 |
| DWARF | [APTM20] | 2020 | | 3.20 | 3.94 | 3.33 | 6.21 | 9.38 | 6.73 | 9.80 | 13.37 | 10.39 | 11.72 | 18.06 | 12.78 | 0.14 |
| SFF++ | [SWUK+20] | 2020 | multi | 4.27 | 12.38 | 5.62 | 7.31 | 18.12 | 9.11 | 10.63 | 17.48 | 11.77 | 12.44 | 25.33 | 14.59 | 78 |
| DTF_PWOC | [SUS21] | 2021 | multi | 3.91 | 8.57 | 4.68 | 6.25 | 14.03 | 7.55 | 10.78 | 19.99 | 12.31 | 12.42 | 25.74 | 14.64 | 0.38 |
| FSF+MS | [TSS17] | 2017 | multi | 5.72 | 11.84 | 6.74 | 7.57 | 21.28 | 9.85 | 8.48 | 25.43 | 11.30 | 11.17 | 33.91 | 14.96 | 2.7 |
| PWOC-3D | [SSWS19] | 2019 | | 4.19 | 9.82 | 5.13 | 7.21 | 14.73 | 8.46 | 12.40 | 15.78 | 12.96 | 14.30 | 22.66 | 15.69 | 0.13 |
| CSF | [LBAL+16] | 2016 | | 4.57 | 13.04 | 5.98 | 7.92 | 20.76 | 10.06 | 10.40 | 25.78 | 12.96 | 12.21 | 33.21 | 15.71 | 80 |
| SFF | [SWKB+18] | 2018 | | 5.12 | 13.83 | 6.57 | 8.47 | 21.83 | 10.69 | 10.58 | 24.41 | 12.88 | 12.48 | 32.28 | 15.78 | 65 |
| PRSF | [VSR13] | 2013 | | 4.74 | 13.74 | 6.24 | 11.14 | 20.47 | 12.69 | 11.73 | 24.33 | 13.83 | 13.49 | 31.22 | 16.44 | 150 |
| SPS+FF++ | [SWS18] | 2018 | | 5.47 | 12.19 | 6.59 | 13.06 | 20.83 | 14.35 | 15.91 | 20.27 | 16.64 | 18.98 | 29.51 | 20.73 | 36 |
| Mono-SF | [BAM19] | 2019 | mono | 14.21 | 26.94 | 16.32 | 16.89 | 33.07 | 19.59 | 11.40 | 19.64 | 12.77 | 19.79 | 39.57 | 23.08 | 41 |
| SGM+SF | [HFR14] | 2014 | | 5.15 | 15.29 | 6.84 | 14.10 | 23.13 | 15.60 | 20.91 | 25.50 | 21.67 | 23.09 | 34.46 | 24.98 | 2700 |
| MonoComb | [SUS20] | 2020 | mono | 17.89 | 21.16 | 18.44 | 22.34 | 25.85 | 22.93 | 5.84 | 8.67 | 6.31 | 27.06 | 33.55 | 28.14 | 0.58 |
| PCOF-LDOF | [DPSL16] | 2016 | | 6.31 | 19.24 | 8.46 | 19.09 | 30.54 | 20.99 | 14.34 | 38.32 | 18.33 | 25.26 | 49.39 | 29.27 | 50 |
| PCOF + ACTF | [DPSL16] | 2016 | | 6.31 | 19.24 | 8.46 | 19.15 | 36.27 | 22.00 | 14.89 | 60.15 | 22.43 | 25.77 | 67.75 | 32.76 | 0.08 |
| SGM+FF+ | [SBWS18a] | 2018 | | 11.93 | 20.57 | 13.37 | 27.02 | 31.71 | 27.80 | 22.83 | 22.75 | 22.82 | 32.26 | 40.12 | 33.57 | 29 |
| Self-Mono-SF-ft | [HR20] | 2020 | mono | 20.72 | 29.41 | 22.16 | 23.83 | 32.29 | 25.24 | 15.51 | 17.96 | 15.91 | 31.51 | 45.77 | 33.88 | 0.09 |
| SGM+C+NL | [SRB14] | 2014 | | 5.15 | 15.29 | 6.84 | 28.77 | 25.65 | 28.25 | 34.24 | 42.46 | 35.61 | 38.21 | 50.95 | 40.33 | 270 |
| SGM+LDOF | [BM11] | 2011 | | 5.15 | 15.29 | 6.84 | 29.58 | 23.48 | 28.56 | 40.81 | 31.92 | 39.33 | 43.99 | 42.09 | 43.67 | 86 |
| DWBSF | [RKVT16] | 2016 | | 19.61 | 22.69 | 20.12 | 35.72 | 28.15 | 34.46 | 40.74 | 31.16 | 39.14 | 46.42 | 40.76 | 45.48 | 420 |
| Self-Mono-SF | [HR20] | 2020 | mono | 31.22 | 48.04 | 34.02 | 34.89 | 43.59 | 36.34 | 23.26 | 24.93 | 23.54 | 46.68 | 63.82 | 49.54 | 0.09 |
| GCSF | [ČSH11] | 2011 | | 11.64 | 27.11 | 14.21 | 32.94 | 35.77 | 33.41 | 47.38 | 41.50 | 46.40 | 52.92 | 56.68 | 53.54 | 2.4 |
| VSF | [HD07] | 2007 | | 27.31 | 21.72 | 26.38 | 59.51 | 44.93 | 57.08 | 50.06 | 45.40 | 49.28 | 67.69 | 62.93 | 66.90 | 7500 |

Appendix C

# Author's Full List of Publications

## 2022

Ramy Battrawy, **René Schuster**, Mohammad-Ali Nikouei Mahani, and Didier Stricker. "RMS-FlowNet: Efficient and Robust Multi-Scale Scene Flow Estimation for Large-Scale Point Clouds." In: *International Conference on Robotics and Automation (ICRA)*. 2022.

## 2021

Michael Fürst, Shriya T. P. Gupta, **René Schuster**, Oliver Wasenmüller, and Didier Stricker. "HPERL: 3D Human Pose Estimation from RGB and LiDAR." In: *International Conference on Pattern Recognition (ICPR)*. 2021.

Syed Muhammad Kumail Raza, **René Schuster**, and Didier Stricker. "Multi-scale Iterative Residuals for Fast and Scalable Stereo Matching." In: *Computer Science in Cars Symposium (CSCS)*. 2021.

Rishav*, **René Schuster***, Ramy Battrawy, Oliver Wasenmüller, and Didier Stricker. "ResFPN: Residual Skip Connections in Multi-Resolution Feature Pyramid Networks for Accurate Dense Pixel Matching." In: *International Conference on Pattern Recognition (ICPR)*. 2021. *Equal contribution. **Oral**.

**René Schuster**, Christian Unger, and Didier Stricker. "A Deep Temporal Fusion Framework for Scene Flow Using a Learnable Motion Model and Occlusions." In: *Winter Conference on Applications of Computer Vision (WACV)*. 2021.

**René Schuster**, Oliver Wasenmüller, Christian Unger, and Didier Stricker. "SSGP: Sparse Spatial Guided Propagation for Robust and Generic Interpolation." In: *Winter Conference on Applications of Computer Vision (WACV)*. 2021.

## 2020

Ramy Battrawy, **René Schuster**, Oliver Wasenmüller, Qing Rao, and Didier Stricker. "deepRGBXYZ: Dense Pixel Description Utilizing RGB and Depth with Stacked Dilated Convolutions." In: *International Conference on Intelligent Transportation Systems (ITSC)*. 2020.

Rishav, Ramy Battrawy, **René Schuster**, Oliver Wasenmüller, and Didier Stricker. "DeepLiDARFlow: A Deep Learning Architecture For Scene Flow Estimation Using Monocular Camera and Sparse LiDAR." In: *International Conference on Intelligent Robots and Systems (IROS)*. 2020.

**René Schuster**, Christian Unger, and Didier Stricker. "MonoComb: A Sparse-to-Dense Combination Approach for Monocular Scene Flow." In: *Computer Science in Cars Symposium (CSCS)*. 2020.

**René Schuster**, Oliver Wasenmüller, Christian Unger, Georg Kuschk, and Didier Stricker. "SceneFlowFields++: Multi-frame Matching, Visibility Prediction, and Robust Interpolation for Scene Flow Estimation." In: *International Journal on Computer Vision (IJCV)* (2020).

## 2019

Ramy Battrawy, **René Schuster**, Oliver Wasenmüller, Qing Rao, and Didier Stricker. "LiDAR-Flow: Dense Scene Flow Estimation from Sparse LiDAR and Stereo Images." In: *International Conference on Intelligent Robots and Systems (IROS)*. 2019.

Rohan Saxena, **René Schuster**, Oliver Wasenmüller, and Didier Stricker. "PWOC-3D: Deep Occlusion-Aware End-to-End Scene Flow Estimation." In: *Intelligent Vehicles Symposium (IV)*. 2019. **Oral**.

**René Schuster**, Oliver Wasenmüller, Christian Unger, and Didier Stricker. "An Empirical Evaluation Study on the Training of SDC Features for Dense Pixel Matching." In: *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019.

**René Schuster**, Oliver Wasenmüller, Christian Unger, and Didier Stricker. "SDC – Stacked Dilated Convolution: A Unified Descriptor Network for Dense Matching Tasks." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019. **Oral**.

**René Schuster**, Oliver Wasenmüller, Christian Unger, and Didier Stricker. "SDC – Stacked Dilated Convolutions." Extended Abstract for the Computer Science in Cars Symposium (CSCS). 2019.

## 2018

Patrik Feth, Mohammed Naveed Akram, **René Schuster**, and Oliver Wasenmüller. "Dynamic Risk Assessment for Vehicles of Higher Automation Levels by Deep Learning." In: *International Workshop on Artificial Intelligence Safety Engineering (WAISE)*. 2018.

**René Schuster**, Christian Bailer, Oliver Wasenmüller, and Didier Stricker. "Combining Stereo Disparity and Optical Flow for Basic Scene Flow." In: *Commercial Vehicle Technology Symposium (CVT)*. 2018.

**René Schuster**, Christian Bailer, Oliver Wasenmüller, and Didier Stricker. "FlowFields++: Accurate Optical Flow Correspondences Meet Robust Interpolation." In: *International Conference on Image Processing (ICIP)*. 2018.

**René Schuster**, Oliver Wasenmüller, Georg Kuschk, Christian Bailer, and Didier Stricker. "SceneFlowFields: Dense Interpolation of Sparse Scene Flow Correspondences." In: *Winter Conference on Applications of Computer Vision (WACV)*. 2018.

**René Schuster**, Oliver Wasenmüller, and Didier Stricker. "Dense Scene Flow from Stereo Disparity and Optical Flow." Extended Abstract for the Computer Science in Cars Symposium (CSCS). 2018.

Oliver Wasenmüller, **René Schuster**, Didier Stricker, Karl Leiss, Jürgen Pfister, Oleksandra Ganus, Julian Tatsch, Artem Savkin, and Nikolas Brasch. "Automated Scene Flow Data Generation for Training and Verification." Extended Abstract for the Computer Science in Cars Symposium (CSCS). 2018.

## 2017

**René Schuster**, Oliver Wasenmüller, Georg Kuschk, Christian Bailer, and Didier Stricker. "Towards Flow Estimation in Automotive Scenarios." Extended Abstract for the Computer Science in Cars Symposium (CSCS). 2017.

Appendix D

# Curriculum Vitae

## Work Experience

| | |
|---|---|
| **since 2020** | **Team Leader** Automotive Scene Understanding (ASU)<br>Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)<br>Department Augmented Vision, Kaiserslautern |
| **2017–2020** | **Researcher** Deutsches Forschungszentrum<br>für Künstliche Intelligenz (DFKI)<br>Department Augmented Vision, Kaiserslautern |
| **2016** | **Student Assistant** Robert Bosch GmbH<br>Driver Assistance Systems, Leonberg |
| **2015–2016** | **Research Assistant** TU Darmstadt<br>Faculty of Computer Science, Visual Inference Group |
| **2013–2014** | **Software Developer** Gesellschaft für Technische Beratung mbH<br>(GESTEB), Weinheim |
| **2010–2013** | **Dual Education Student** Infineon Technologies AG<br>Reverse Engineering, Neubiberg |

## Education

| | |
|---|---|
| **2013–2017** | **Master of Science in Computational Engineering**<br>TU Darmstadt<br>Master thesis: *3D Object Proposals from Stereo Disparity and Optical Flow* |
| **2010–2013** | **Bachelor of Engineering in Electrical Engineering**<br>DHBW Stuttgart<br>Bachelor thesis: *Konvertierung von präparierten Bilddateien in Vektordarstellung mit Analyse der Bildelemente* |