Institute for Visualization and Interactive Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

# Flow Prediction Meets Flow Learning: Combining Different Learning Strategies for Computing the Optical Flow

Haining Yang

| | |
|---|---|
| **Course of Study:** | Informatik |
| **Examiner:** | Prof. Dr. -Ing. Andrés Bruhn |
| **Supervisor:** | Daniel Maurer |
| **Commenced:** | November 14, 2018 |
| **Completed:** | May 14, 2019 |

## Abstract

Optical flow estimation is an important topic in computer vision. The goal is to compute the inter-frame displacement field between two consecutive frames of an image sequence. In practice, optical flow estimation plays a significant role in multiple application domains including autonomous driving and medical imaging. Different categories of methods exist for solving the optical flow problem. The most common technique is based on a variational framework, where an energy functional is designed and minimized in order to calculate the optical flow. Recently, other approaches like pipeline-based approach and learning-based approach also attract much attention. Despite the great advances achieved by these algorithms, it is still difficult to find an algorithm that can perform well under all the challenges, e.g. lightning changes, large displacements, and occlusions. Hence, it is worth combining different algorithms to create a new approach that can combine their advantages. Inspired by this idea, in this thesis we select two top-performing algorithms PWC-Net and ProFlow as candidate approaches and conduct a combination of these two algorithms. While PWC-Net performs generally well in the estimation of non-occluded areas, ProFlow can especially provide an accurate estimation for the occluded areas. Thereby, we expect that the combination of these two algorithms can yield an algorithm that performs well in both occluded and non-occluded areas. Since ProFlow is a pipeline approach, we first integrate the PWC-Net in the ProFlow pipeline, then evaluate the new created pipeline PWC-ProFlow based on the MPI Sintel and KITTI 2015 benchmarks. Contrary to our expectations, the newly created algorithm does not exceed the candidate methods PWC-Net and ProFlow on either benchmark. Through the analysis of the evaluation results, we explore the problems hidden in the PWC-ProFlow pipeline that can lead to its underperformance, and organize some modification ideas. Based on these ideas, we propose six new pipelines with the purpose of improving the estimation accuracy of PWC-ProFlow. All the new generated pipelines are also evaluated on the Sintel and KITTI benchmarks. The experiment results demonstrate that all the modifications created achieve great improvements on both datasets compared to PWC-ProFlow. Further, all of them also outperform the ProFlow pipeline on both benchmarks. Compared to PWC-Net, one modification exceeds PWC-Net on the KITTI dataset, however, all our modifications achieve a better performance on the Sintel dataset, in particular, one modification presents a significant improvement with a more than 10% lower average endpoint error on the Sintel dataset.

# Contents

# 1 Introduction

## 1.1 Motivation

The extraction of motion information from image sequences is one of the fundamental problems in computer vision. Typically, one considers the motion estimation problem to be equivalent to the optical flow problem [VP89], which computes the inter-frame motion field between two consecutive images, as presented in Figure 1.1. The applications of optical flow estimation cover different domains including autonomous navigation, advanced driver assistance systems (ADAS), video analysis, and medical imaging.

Since optical flow allows to estimate the location of an object in the next frame, it can be applied to object tracking. Combined with navigation information, such as rotational velocities and terrain positions, optical flow prediction can assist the process of solving different tasks in robots and vehicle navigation, such as collision detection and obstacle avoidance [CGN14]. The same benefits to ADAS applications, where computing optical flow is highly useful for the estimation of the vanishing point, the horizon line, and the ego-motion [Onk13]. In combination with modern object detection frameworks employing deep neural networks, tasks in the fields of video analysis and video surveillance can be performed more efficiently, such as action recognition, gesture recognition, and facial expression recognition[FBK15; HZZ17]. Similarly, in a biomedical context, the estimation of blood flow and the observation of radiation dose distribution are examples of medical applications that require accurate motion analysis [FBK15]. All of these can indicate the important role of optical flow in real applicative domains.

In most scenarios, approaches that can solve the optical flow problem need to face many difficult situations, e.g. motion discontinuities, large displacements, lightning changes, and occlusions. Despite the great progress achieved by the recent algorithms, it seems unlikely that an existing algorithm can handle all these issues. While some approaches may perform well on some of the challenges, other methods may overcome some other difficulties. Hence, it would be desirable to combine different computing strategies in a reasonable way to create a novel approach that can outperform these algorithms and combine their advantages.

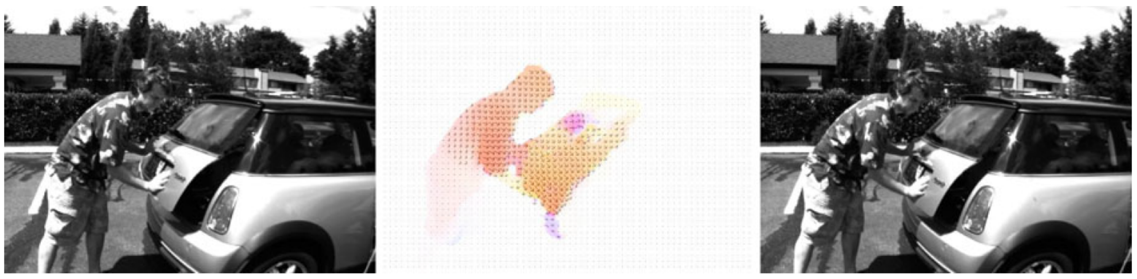


**Figure 1.1:** Optical flow estimation between two consecutive frames. **Left:** Image frame at time t. **Right:** Image frame at time t+1. **Middle:** Optical flow: estimated displacement vectors on the top of the flow color scheme. Figure from [WC11].

## 1.2 Previous Work

The primary research on the optical flow problem dates back decades. However, considering its significance in different applications, the optical flow problem remains to be one of the most popular topics in computer vision. A wide range of approaches exist for solving this problem, from traditional variational approaches to modern learning-based and pipeline-based approaches.

**Variational approach.** In 1981, Horn and Schunck first proposed a variational optical flow approach, which computes the motion field as a minimizer of an energy functional consisting of a data term and a smoothness term [HS81]. The data term penalizes deviations from the brightness constancy assumption, while the smoothness term enforces smoothness on the estimation result. Even with this new approach, difficulties like illumination changes, large displacements, and robustness are still unresolved [Bru18]. To overcome the different challenges, diverse kinds of data terms and smoothness terms have been explored over the years. These data terms consider adopting the gradient constancy assumption to better resist illumination changes, using RGB channels to fully exploit information in color sequences, and applying non-quadratic penalty functions to reduce the influence of outliers [Bru18]. Different smoothness terms are also proposed to cope with image discontinuities and to increase robustness, such as image-driven isotropic and anisotropic smoothness terms [Bru18]. Beyond that, researchers proposed to combine variational approaches with other techniques. For example, embedding the variational approach in a coarse-to-fine scheme can help to estimate large displacements, and applying feature matching in the estimation process can further increase the detection accuracy of small and rapidly moving objects [FBK15]. Even thought variational approaches become more and more advanced, they still yield complex optimization problems, hence, extensive computational costs.

**Partially learning-based approach.** In recent years, deep learning is regarded as a great tool for tackling some high-level computer vision problems, such as object detection and object segmentation [LDY18; ZZXW19]. Inspired by its good performance, researchers also integrated this technique into traditional optical flow approaches, known as partially learning-based approach. In these methods, CNN networks are normally used as components to solve some intermediate tasks that are computationally expensive or difficult to solve with traditional solutions. For example, Xu *et al.* [XRK17] took advantage of CNN networks to learn nonlinear features for constructing a full cost volume. Methods like [BVS17; GW16; SRLB08] also adopted CNN networks as a feature extractor. Further, in the MRFlow proposed by Wulff *et al.* [WSB17], convolutional networks were applied to classify rigid and non-rigid regions in a scene, so that different regions can be handled separately. For example, moving objects are estimated using unconstrained flow method, and for static regions, the *Plane-Parallax* framework is adopted to estimate the scene structure and the camera motion. Bai *et al.* [BLKU16] used deep learning to perform instance-level segmentation, aiming to produce individual optical flow estimation for each object and the background. Sevilla-Lara *et al.* [SSJB16] employed CNN models for semantic scene segmentation to build different motion models according to semantic class labels. Since all these methods treat CNN networks only as partial components, a real-time estimation is still hard to accomplish.

**Pure learning-based approach.** Instead of using CNN networks merely as parts of the methods, pure learning-based approaches aim to train an end-to-end network that allows to estimate the flow field directly from input images. With the pretrained network, this kind of approaches can realize a real-time estimation. Dosovitskiy *et al.* [DFI15] first implemented this paradigm by proposing two CNN models, i.e. FlowNetS and FlowNetC. Both models demonstrate the possibility of casting the optical flow problem as a supervised learning problem. However, their performance could not compete with the top-performing variational approaches. In order to improve the accuracy, Ilg *et al.* [IMS17] stacked multiple FlowNetS and FlowNetC together to create a more complex model FlowNet2, where small and large displacements are estimated by two different networks separately and then fused to produce the final estimation. Due to the deep network, FlowNet2 performs on par with the state-of-the-art. To decrease the model size, Ranjan *et al.* [RB17] developed a network called SpyNet, which embeds traditional principles in neural networks, such as image pyramid and warping strategy. Inspired by SpyNet, PWC-Net [SYLK18] is proposed recently, which is more accurate than all existing approaches based on an end-to-end network. This network will be explained in Chapter 3.

**Pipeline approach.** Another type of approaches is the pipeline approach, which includes four steps: matching, filtering, inpainting, and variational refinement [MSB17]. Pipeline approaches aim to provide a good initial flow, which is computed in the first three steps, for the final variational refinement. Compared to other methods, pipeline approaches possess higher flexibility and adaptivity, i.e. the algorithms used in each step can be configurated in terms of application scenarios.

The matching step requires to solve a correspondence problem with two given images. To solve this problem, we first need to compute features that are optimally invariant under illumination changes and image deformations, such as using the famous SIFT descriptor [Low04]. The descriptors can be applied to each pixel of an image or only to interest locations, such as corners. For each input image, a set of features are computed, then it is necessary to create matching criteria based on similarities to find the best matches between the two sets [Bru18]. Since the second step in a pipeline approach is filtering, it would be desirable to generate rather dense matches in the first step. In this context, algorithms like [LHS18; MHG15; WRHS13] are examples of appropriate candidates.

The matches computed in the first step usually contain certain outliers, such as wrong matches caused by occlusions or the case where two or more pixels in the reference image correspond to the same pixel in the next frame. It is important to detect and filter these outliers, in order to guarantee the quality of the inpainting step and the overall estimation. The bidirectional consistency checking and uniqueness checking are two of the approaches that can implement the filtering [Bru18]. More details about these two methods will be given in Section 2.3.

In the filtering step, a certain number of matches are filtered out. As a result, a non-dense flow field is generated. However, the final variational refinement requires a fully dense flow field as input. In this context, an inpainting step is applied to achieve a sparse-to-dense interpolation. Normally, the non-available matches are filled in based on the neighboring information and the contour structure in order to retain the sharp edges, such as in EpicFlow [RWHS15] and RIC [HLS17].

The final step makes use of a variational approach to refine the inpainted flow field. Here, a standard energy minimization framework is employed to gain a sub-pixel precision [Bru18]. In addition to the traditional first-order refinement, Maurer *et al.* [MSB17] proposed an advanced variational model for refinement, which combines an illumination aware data term with an order adaptive smoothness term. In terms of the evaluation on public benchmarks, this algorithm can bring more improvements to the input flow field than traditional methods.

## 1.3 Goal of the Thesis

The goal of this thesis is to combine two top-performing algorithms, PWC-Net and ProFlow, to create a new approach that can optimally combine the advantages of these two algorithms. PWC-Net is a pure learning-based approach which allows a real-time estimation of optical flow with high accuracy, especially in non-occluded areas. ProFlow is a pipeline approach, combined with a shallow CNN network, which serves to determine the underlying motion model, thus generating the same accurate estimation in occluded areas. After combining, the newly created approach is evaluated based on the public benchmarks MPI Sintel [BWSB12] and KITTI 2015 [MG15], with an emphasis on accuracy, robustness, and runtime. In addition, the interaction between PWC-Net and different steps of ProFlow is also investigated. Based on the analysis of the interaction and evaluation, the new generated method is further modified and optimized.

## 1.4 Outline

The rest of the thesis is organized as following: Chapter 2 introduces the background of the optical flow problem including the definition of optical flow, the detection of occlusions, the common method used to cope with large displacements, the evaluation metrics, and the standard visualization techniques. In Chapter 3, the two algorithms PWC-Net and ProFlow are explained to lay a foundation for the further combination. Chapter 4 first explains the combination of PWC-Net with the ProFlow pipeline, and then presents the evaluation results of the new created pipeline PWC-ProFlow on the Sintel and KITTI benchmarks. After analyzing the evaluation results, six different modifications are proposed in Chapter 5 to optimize the PWC-ProFlow pipeline. All the modifications are also evaluated on the two datasets. Further, the estimation performance of different modifications is compared with each other and with the performance of PWC-Net, ProFlow, and PWC-ProFlow. Lastly, Chapter 6 summarizes the work completed in the thesis with an overview of the future work.

# 2 Background

This chapter introduces the basic concepts regarding the optical flow estimation and lays an important foundation for the following chapters. Section 2.1 introduces the formal definition of the optical flow problem and the brightness constancy assumption. The aperture problem, which is an inherent problem contained in the optical flow estimation, is described in Section 2.2. Since handling occlusions is one of the focuses in this thesis, two occlusion detection methods are explained in Section 2.3, followed by the introduction of the coarse-to-fine scheme in Section 2.4. In order to better evaluate and demonstrate estimation results, quality measures and the commonly used visualization techniques are discussed in Section 2.5 and Section 2.6.

## 2.1 Optical Flow Problem

Given two consecutive frames of an image sequence taken by a camera at different times, optical flow problem is to estimate the inter-frame displacement field between the two input frames. The displacement field usually represents the apparent motion of brightness patterns, which is caused by the relative movements between the camera and the object scene, such as a moving camera or moving objects [FBK15].

The foundation of solving the optical flow problem is to find corresponding pixels between two frames. For this purpose, we always assume that certain features of a pixel remain constant over time. This assumption allows to obtain pixel matches by tracking these features between frames. Based on the pixel matches, the corresponding displacement vector of each pixel can be computed.

In the following parts , we use $I(x, y, t)$ to represent a continuous image sequence, where $(x, y)$ denotes a location within the image domain, $t$ records the time of a frame, and $I(x, y, t)$ yields the intensity of position $(x, y)$ on a frame with time $t$.

#### The Grey Value Constancy Assumption

The most widely used assumption is the grey value constancy assumption, which describes the intensity of a point in the image domain remaining constant as it moves. Here, we denote the displacement vector of a point as $(u(x, y, t), v(x, y, t))^\top$, then the grey value constancy assumption reads as following [Bru18]:

$$I(x+u, y+v, t+1) - I(x, y, t) = 0. \tag{2.1}$$

However, this equation often produces a difficult optimization process due to the implicit functions. A common way to deal with it is to linearize the equation via a first-order Taylor expansion around the point $(x, y, t)^\top$, thus obtaining the **brightness constancy constraint equation (BCCE)** [Bru18]:

$$I_x u + I_y v + I_t = 0, \tag{2.2}$$

where $I_x$ and $I_y$ represent the partial derivatives of $I$ w.r.t. $x$ and $y$, respectively. $I_t$ can be considered as the derivative in the time orientation.

The linearized expansion usually appropriates to small displacements or very smooth images. In the case of large displacements, the approximation is less accurate. Since the linearization plays an important role in most differential methods, other techniques need to be combined to overcome this drawback. One of the standard strategies to cope with large displacements is to employ a coarse-to-fine scheme [BBPW04], which will be described in Section 2.4.

## 2.2 Aperture Problem

The BCCE gives an equation with two unknown variables, $u$ and $v$. Using this equation alone can only determine the flow component in the direction of image gradient, which is referred to as the normal flow and computed as following:

$$0 = I_x u + I_y v + I_t = \begin{pmatrix} u \\ v \end{pmatrix}^\top \nabla I + I_t \tag{2.3}$$

$$\begin{pmatrix} u \\ v \end{pmatrix}^\top_{normal} = \left[ \begin{pmatrix} u \\ v \end{pmatrix}^\top \frac{\nabla I}{|\nabla I|} \right] = \frac{-I_t}{|\nabla I|} \frac{\nabla I}{|\nabla I|} \tag{2.4}$$

However, the flow component perpendicular to the image gradient cannot be calculated by the BCCE. According to Equation 2.3, adding any arbitrary value in the perpendicular direction does not violate the BCCE, thus, an infinite number of $(u(x, y, t), v(x, y, t))^\top$ can be regarded as solution. This is known as the aperture problem, indicating that motion of linear or untextured structures is by nature ambiguous, without regard to neighboring information [FBK15], as shown in Figure 2.1.

The non-uniqueness makes optical flow problem ill-posed in the sense of Hadamard. To make the two-dimensional optical flow vector uniquely solvable, other information or assumptions need to be involved, such as the neighboring information [LK81] or the smoothness assumption of the optical flow field [HS81].

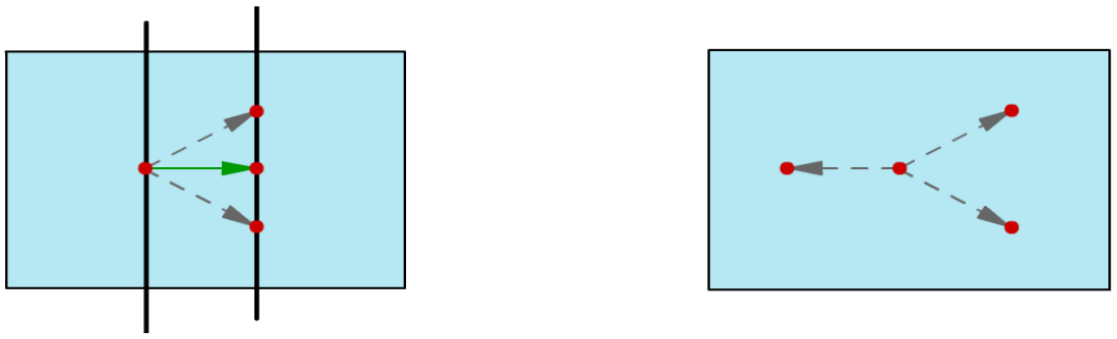


**Figure 2.1:** Aperture problem. **Left:** In case $|\nabla I| \neq 0$, one single point corresponds to multiple points. **Right:** In case $|\nabla I| = 0$, no information can be obtained. Figure from [Bru18].

## 2.3 Occlusion Handling

In the previous two sections, we explained the definition of the optical flow problem and introduced the basic ideas of how to solve this problem. In this section, we deal with one of the most challenging difficulties regarding optical flow estimation, i.e. the occlusions. In most cases, occluded pixels are the main source of outliers, thereby handling these pixels appropriately is essential to a reliable and accurate estimation.

Occluded regions in a frame are defined as a set of pixels which become hidden by moving objects in the next frame or leave the camera field of view while moving [FBK15]. As shown in Figure 2.2, the pixels marked in white are from the occluded regions. Obviously, occluded pixels possess no matching correspondence in the next frame. Hence, wrong matches or inconsistent matches could appear in the occluded regions.

A standard method of occlusion detection is bidirectional consistency checking, which makes use of both forward and backward matches to detect flow consistency. Suppose we have two images $t$ and $t+1$. As a first step, we compute the forward flow $(u^{fw}, v^{fw})$ from image $t$ to image $t+1$ and the backward flow $(u^{bw}, v^{bw})$ from image $t+1$ to image $t$. Then the consistency checking is performed in terms of the following expression [Bru18]:

$$\begin{aligned}\left|\triangle u(x,y,t)\right| = \Big\{&(u^{fw}(x,y,t) + u^{bw}(x+u^{fw}(x,y,t), y+v^{fw}(x,y,t), t+1))^2 + \\ &(v^{fw}(x,y,t) + v^{bw}(x+u^{fw}(x,y,t), y+v^{fw}(x,y,t), t+1))^2\Big\}^{1/2}, \quad (2.5)\end{aligned}$$

where $(x, y, t)$ and $(x + u^{fw}(x, y, t), y + v^{fw}(x, y, t), t + 1)$ denote a location in the frame $t$ and its corresponding location in the frame $t + 1$, respectively.

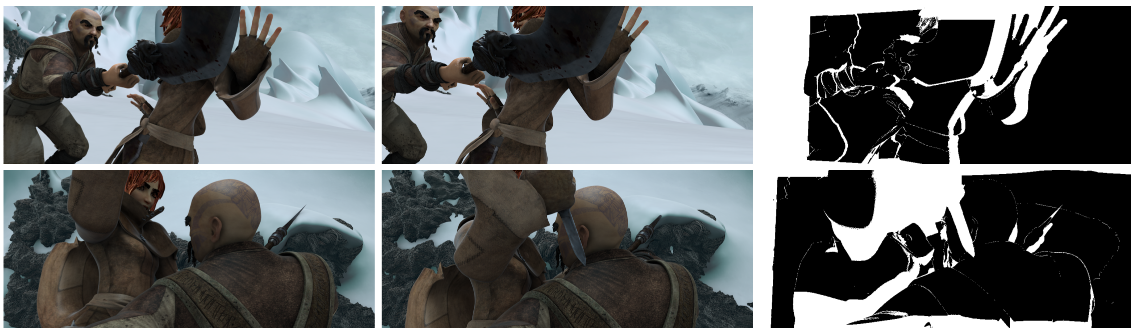


Figure 2.2: Demonstration of occluded regions in a frame (ambush_2 #4 and ambush_6 #15 of the Sintel dataset). **From left to right:** Reference frame, subsequent frame, and occlusion map [BWSB12].

In case of correctly estimated pixels, the sum terms vanish due to consistent flow in both directions and the value of $\left|\triangle u(x,y,t)\right|$ approximates to 0. Inconsistent matches can produce a large $\left|\triangle u(x,y,t)\right|$ due to an inaccurate prediction in at least one direction, which can be seen as an indicator of an occluded pixel. In practice, we usually set a threshold $T_{occ}$ for $\left|\triangle u(x,y,t)\right|$ and treat all pixels with $\left|\triangle u(x,y,t)\right|$ larger than the threshold as occluded.

As one can see, the bidirectional checking requires the computation of flow fields in two directions, which can double the computational costs. An alternative strategy for occlusion detection is uniqueness checking, which is based on the consideration of matching uniqueness, i.e. multiple pixels mapping to the same pixel in the next frame using forward warping are possibly occluded by each other. This idea is adopted in [XJM12]. The occluded regions are detected through the following equation:

$$o(x,y,t) = \begin{cases} 0 & \text{if } m(x+u^{fw}(x,y,t), y+v^{fw}(x,y,t), t+1) = 1 \\ 1 & \text{else} \end{cases} \tag{2.6}$$

where $(x+u^{fw}(x,y,t), y+v^{fw}(x,y,t), t+1)$ represents the target position in the next frame. The $m(z)$ function counts the total number of pixels landing at position $z$. In the case that the target position leaves the image boundary, $m(z)$ is set to be 0 by default. $o(x,y,t)$ is assigned a value of 0 (non-occluded) only when $(x,y,t)$ is the unique correspondence for its target position.

## 2.4 The Coarse-to-fine Approach

After introducing how to detect occluded regions, we now deal with another challenging problem, i.e. computing optical flow for large displacements. A widely used strategy is to embed the variational approach in the coarse-to-fine scheme, which yields the coarse-to-fine variational approach. This approach is based on an image pyramid, consisting of down-sampled versions of the input image. For linearization-based variational approaches, large displacements can be accurately estimated at coarser levels, since they become small due to the effect of down-sampling. Further, the flow field estimated at one level is usually interpolated and used as an initialization for the next level, which can help to reduce the search scope and avoid falling into large local minima during estimation [Bru18; FBK15].

Given an input image $f$, we first construct an image pyramid with $L$ levels. The zeroth level places the original image $f$ and the $L-1$ level is the coarsest level of the pyramid. Starting with zeroth level, the image at an arbitrary level $l$ can be obtained through the following two steps: (1) apply Gaussian filtering to the image at $l-1$th level to remove noise and to avoid aliasing effect in the subsequent sampling (2) perform down-sampling to the filtered image to reduce its resolution by a predefined factor $\eta$.

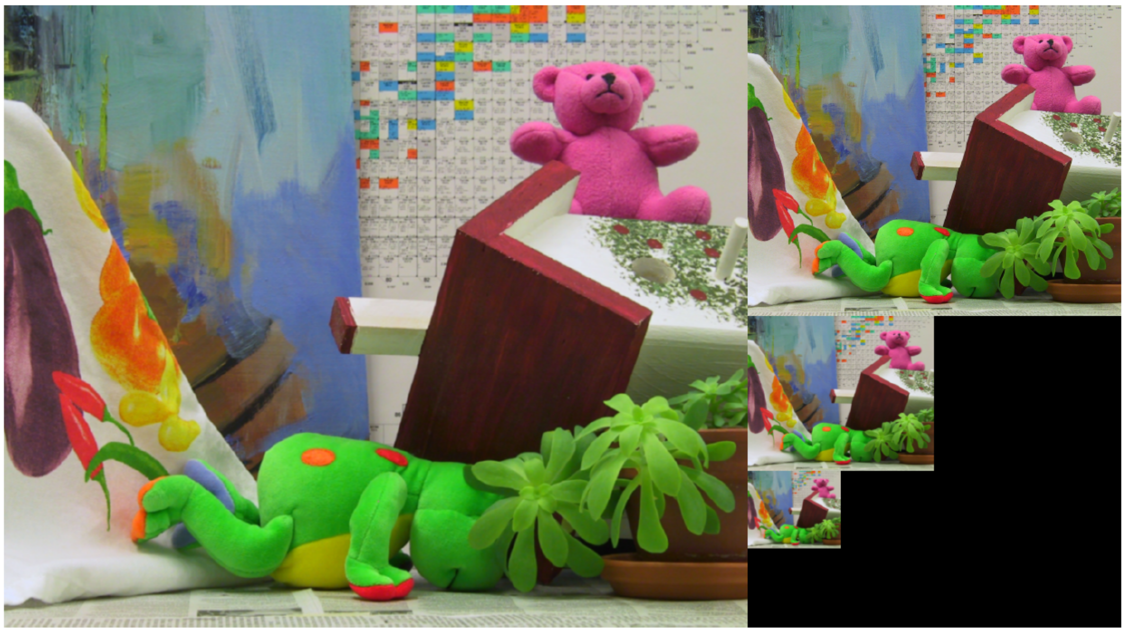


**Figure 2.3:** Image pyramid of 4 levels with the down-sampling factor $\eta = 0.5$. Image is from the Teddy sequence of the Middlebury dataset [BSL11].

After generating the image pyramids for two input images, the whole estimation process is started from the coarsest level. Two pyramid levels are connected through the so-called warping strategy. For example in Figure 2.4, we first compute the flow field at coarsest level $L-1$ using a variational approach. After that, the computed flow field is up-sampled to the resolution of image at $L-2$ level and then applied to warp the second image towards the first. The first and warped images are then applied to the variational approach at $L-2$th level. The flow field is refined iteratively from the coarsest level to the zeroth level, where the finial estimation result is produced.

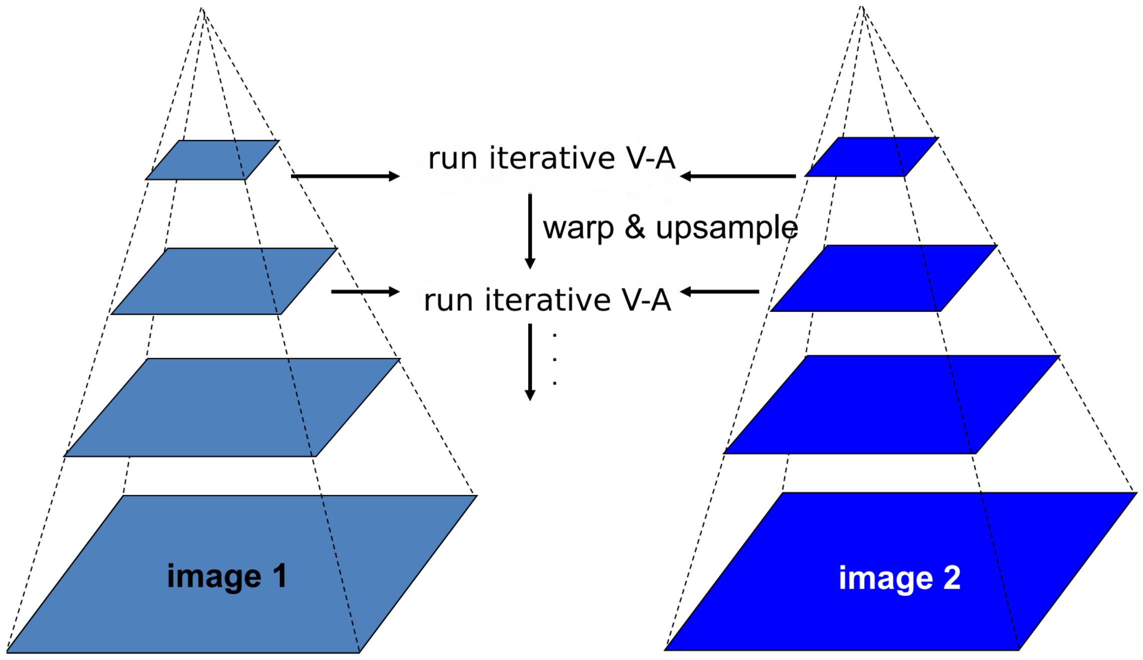


**Figure 2.4: Left:** Image pyramid of the first input image. **Right:** Image pyramid of the second input image. **Middle:** Operations applied at each level. Figure is the modified version of an image from [SB12].

## 2.5 Quality Measures

In the previous sections, we described the common approaches to detect occlusions and to solve the problem regarding large displacements in the estimation process. In this Section, we introduce two commonly used error metrics to evaluate the performance of different optical flow approaches. Normally, the error between the estimated flow and the ground truth flow is quantitatively measured With the evaluation metrics. In the following, we adopt the discrete version of a continuous image signal $I(x, y, t)$, where $u_{i,j}$ denotes a pixel located at $(i, j)$ in the image. The width and height of the image are $N$ and $M$, respectively.

The first error metric refers to the average endpoint error (AEE), proposed by Otto and Nagel in 1994 [ON94]. AEE computes the average Euclidean distance between all the estimated flow vectors $(u^e, v^e)$ and the ground truth flow vectors $(u^t, v^t)$. The equation reads as following:

$$AEE(u^t, u^e) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \sqrt{(u^t_{i,j} - u^e_{i,j})^2 + (v^t_{i,j} - v^e_{i,j})^2}. \tag{2.7}$$

The second metric bad pixel error (BP) is based on the computation of the endpoint error (EE). One pixel is defined as a bad pixel, when its EE greater than a predefined tolerant value $T$. Equation 2.8 demonstrates the computation of BP, where $\mathbb{1}(x)$ represents an indicator function. In the equation, the total number of bad pixels is first accumulated, and then the percentage of bad pixels relative to all pixels is calculated.

$$BP(u^t, u^e) = \frac{100}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \mathbb{1}_{\left(\sqrt{(u^t_{i,j} - u^e_{i,j})^2 + (v^t_{i,j} - v^e_{i,j})^2} > T\right)}. \tag{2.8}$$

## 2.6 Visualization Techniques

Another way to compare the estimation results is to visualize them. The visualization of displacement fields can provide a direct insight into how good the estimation performance is. There exist two main visualization methods, i.e. color visualization and arrow visualization [FBK15]. In the arrow visualization, seeing Figure 2.5 (c), the estimated motion vectors are directly presented, which can provide an intuitive perception of physical motion. If the motion arrows of all the pixels in a frame are visualized, we may see a whole black area. Hence, the flow arrows used for display are typically sampled from the original motion field, for the sake of a clean visualization.

For color visualization, a color code is adopted, where the color hue and saturation illustrate the direction and magnitude of a flow vector. The flow vector of each pixel associates to a color, thus, all flow vectors can be visualized via an image, as shown in Figure 2.5 (d). Further, since humans are more sensitive to colors, the color visualization can provide a better observation of small motion vectors.

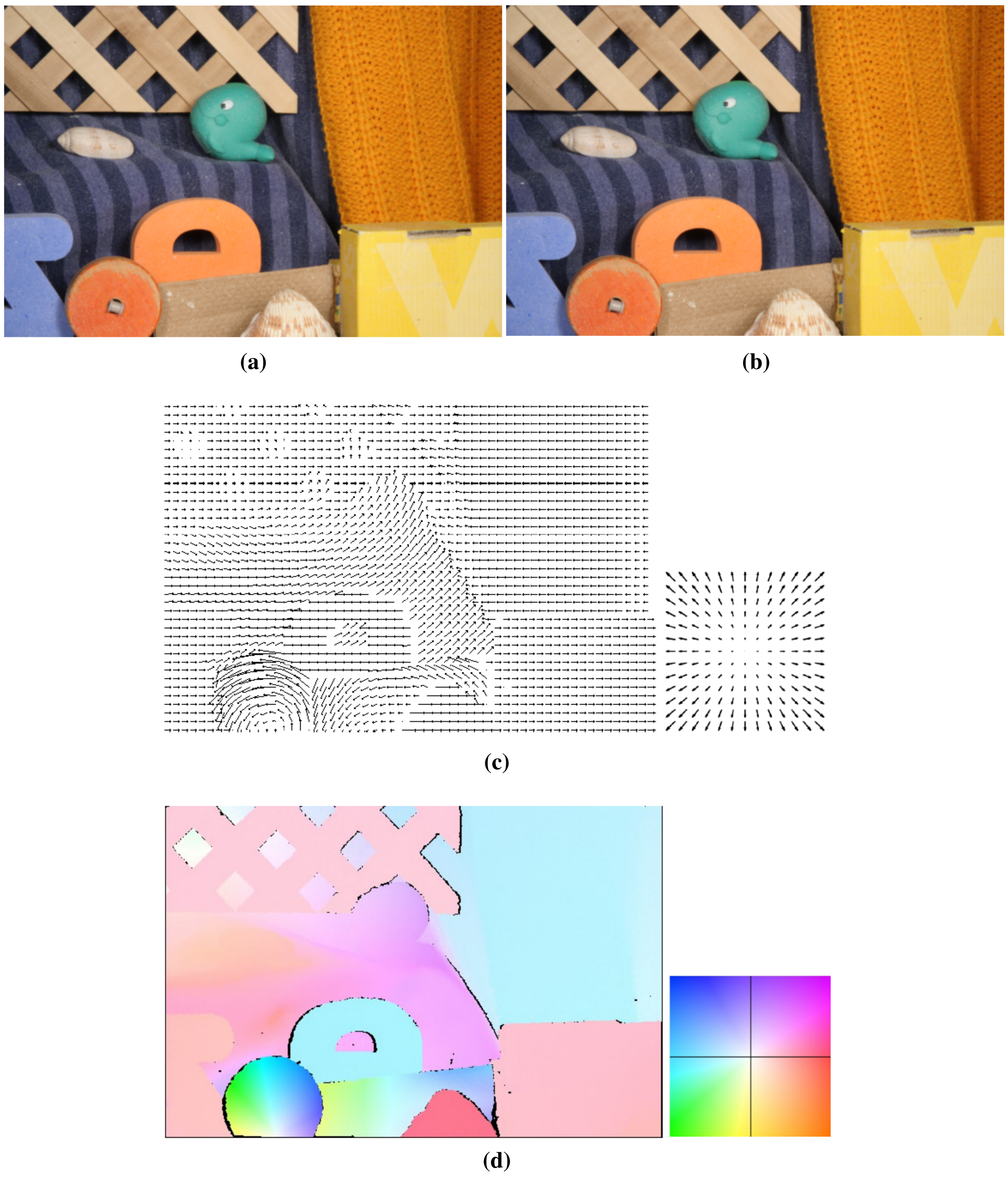


**Figure 2.5:** Visualization techniques of optical flow. **(a)** Input image at time $t$. **(b)** Input image at time $t+1$. **(c)** Optical flow visualization in arrows. **(d)** Color code and optical flow visualization in colors. Figure from [FBK15].

# 3 Related Algorithms

After explaining the basic concepts regarding optical flow estimation in the previous chapter, we now analyze the two top-performing algorithms PWC-Net and ProFlow. Section 3.1 introduces the PWC-Net, including its network structure and definition of loss function, while Section 3.2 presents the core idea of the ProFlow pipeline and how each step in the pipeline works.

## 3.1 PWC-Net

As described in Section 1.2, PWC-Net belongs to the pure learning-based approach, which trains an end-to-end network to solve the optical flow problem. A common difficulty of this kind of methods is the trade-off between model size and accuracy [SYLK18]. Most of the methods, e.g. FlowNet2 [IMS17], improve the estimation precision by deepening the network, which can in turn produce a large burden on training. To reduce the network size while maintaining high accuracy, PWC-Net combines several basic and well-established principles with deep neural networks. This idea has already been realized in the SpyNet [RB17]. However, it embeds only two traditional principles, i.e. the image pyramid and warping strategy, in convolutional networks. As making more use of domain knowledge, it turns out that PWC-Net can achieve higher accuracy and better runtime performance.

One of the applied principles in PWC-Net is the coarse-to-fine strategy, i.e. pyramid processing. At each level of the pyramid exists an end-to-end CNN network, which generates flow estimation for this level. The result from one level is passed to the next level. The networks at different levels possess the same structure but different parameters. In the following, we first explain the generation of feature pyramids for two input images and then analyze the generic network structure at one pyramid level ($m$th level), as shown in Figure 3.1. Further, the definition of the loss function used for training is also discussed. The explanations are based on paper [SYLK18].

In the following, we use the symbol $w$ to represent the flow vector $(u, v)^\top$ to better formulate the equations.

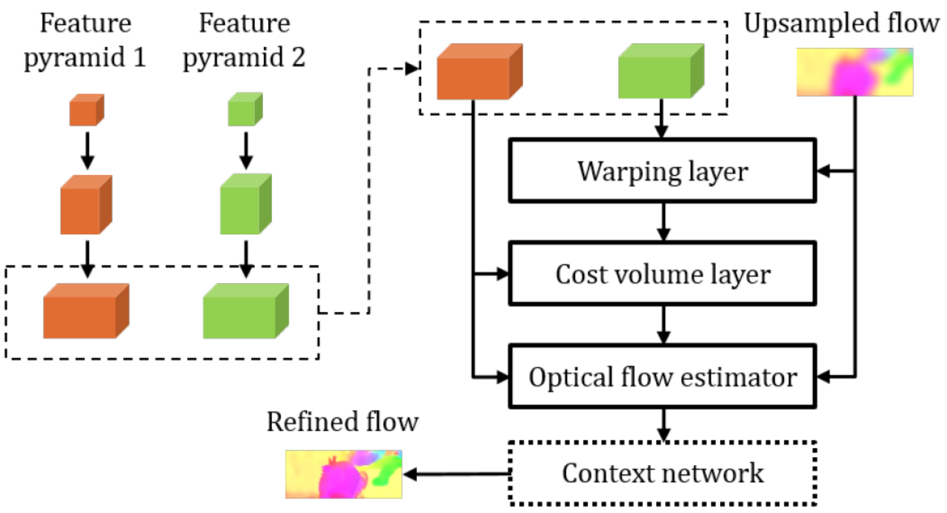


**Figure 3.1:** Feature pyramids and network structure of PWC-Net at one pyramid level. Figure from [SYLK18].

### 3.1.1 Feature Pyramid Extractor

Since raw images are usually very sensitive to different shadows and illuminations, PWC-Net uses a neural network to extract features that remain invariant to different changes and to build learnable feature pyramids. The generation process of the feature pyramid is quite similar to that of the normal image pyramid. In the following, we first introduce the general building process of a feature pyramid.

Assume we want to construct a feature pyramid with $L$ levels for an input image $I$, denoting the levels of the pyramid as $\{c^0, c^1, \ldots, c^{L-1}\}$. By default, the original image is placed at the bottom level, i.e. $c^0 = I$, and the feature map at $m$th level can be obtained by a series of convolution of the feature map at $m$-1th level. Here, we first apply a convolutional layer with stride equal to 2 to downscale the resolution of the feature map at $m$-1th level by a factor of 2. Afterwards, some convolutional layers are used to extract more complex features while maintaining the resolution. The feature pyramid is generated from the finest level to the coarsest by sequentially applying the convolutional layers between each two levels.

In PWC-Net, a feature pyramid with 7 levels is built for the input image. Since PWC-Net takes two consecutive images as inputs, two fully identical feature pyramid extractors are combined to build a Siamese network, which can generate the two 7-level feature pyramids simultaneously. Figure 3.2 presents the network structure of the feature pyramid extractor used in PWC-Net. As we can see, the numbers of feature channels of convolution filters from the first to the sixth level are 16, 32, 64, 96, 128, and 192, respectively.

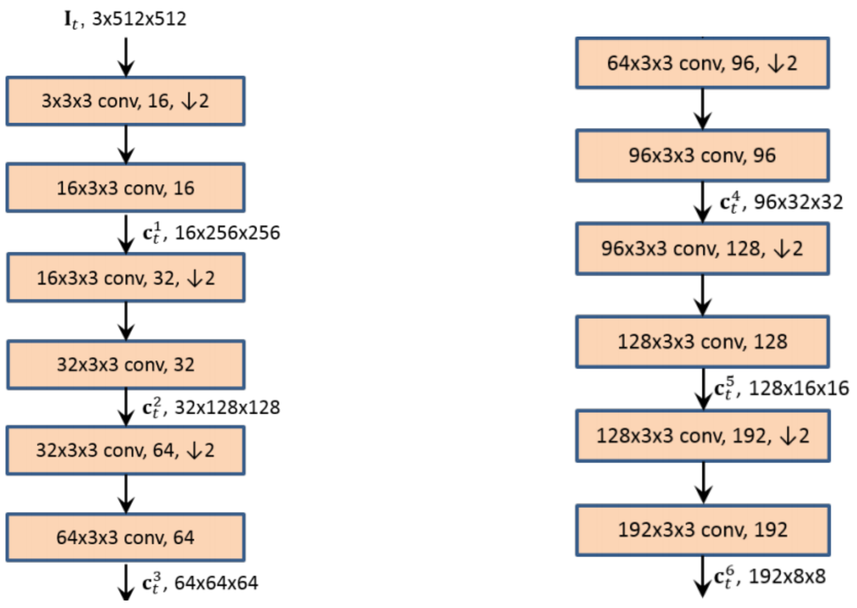


**Figure 3.2:** The feature pyramid extractor network. The first image (t = 1) and the second image (t = 2) are encoded using the same network with identical parameters and weights. Each convolution is followed by a leaky ReLU unit. $c_t^l$ denotes extracted features of image t at level l. Figure from [SYLK18].

### 3.1.2 Warping Layer

The second applied principle in PWC-Net is the warping strategy, which is an important tool in traditional coarse-to-fine approaches to connect different pyramid levels. PWC-Net realizes the warping strategy through a warping layer, which takes as inputs the $m$th level of the feature pyramid of the second image $c_2^m$, and the up-sampled flow estimation at $m$+1th level $up_2(w^{m+1})$. As mentioned above, we use symbol $w$ with $w = (u, v)^\top$ to represent the flow field. The warped feature map at $m$th level is generated using the following equation:

$$c_w^m(x) = c_2^m(x + up_2(w^{m+1})(x)), \tag{3.1}$$

where $x$ denotes the location of a pixel, and $up_2(w^{m+1})(x)$ yields the estimated flow vector of pixel $x$ extracted from the up-sampled flow estimation at $m$+1th level. In most cases, the result of $x + up_2(w^{m+1})(x)$ is not located at an exact pixel position, thus, the bilinear interpolation is applied to compute the warped feature $c_w^m(x)$. At the coarsest level, $up_2(w^L)$ is initialized as zero by default. Since the motion between the first and warped image is usually small, the search field for each pixel can be constrained to a small size, thus it is possible to use a small CNN network to estimate this motion.

### 3.1.3 Cost Volume Layer

The third used principle is cost volume. The cost volume is a widely used component in stereo matching, which stores the matching costs for associating one pixel with its corresponding pixels in another frame. Many algorithms proposed also adopt the full cost volume for solving the optical flow problem, such as in DCFlow [XRK17]. Different from stereo matching, where the search space for the matching can be restricted to a 1D space along the epipolar line, a more complex 2D search is performed for optical flow [LUT18]. Thus, building a full cost volume can be rather computationally expensive. Instead of generating a full cost volume, PWC-Net builds a partial cost volume at each pyramid level by setting limited search window for each pixel. It can help to reduce both computational costs and memory usage.

The cost volume layer is used for constructing the partial cost volume, where the matching cost is defined as the correlation between features of the first image and the warped features of the second image. The correlation is interpreted as following:

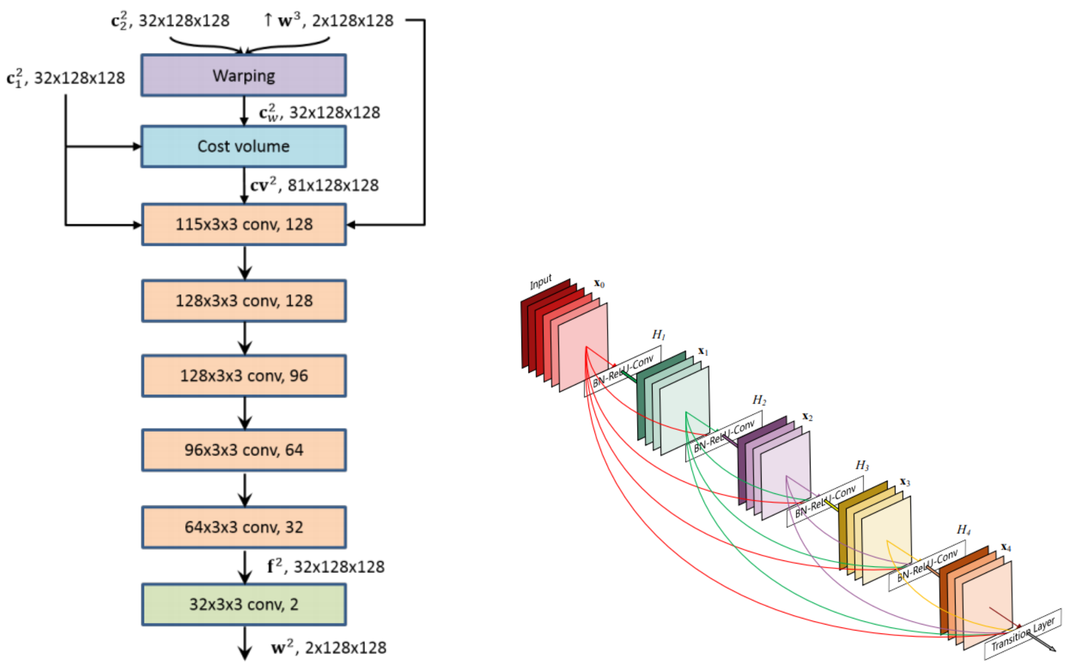
$$cv^m(x_1, x_2) = \frac{1}{N}(c_1^m(x_1))^\top c_w^m(x_2), \tag{3.2}$$

where $x_1$ and $x_2$ are two pixels, and N denotes the length of the feature vector $c_1^m(x_1)$ and $c_w^m(x_2)$.

To construct a partial cost volume, the relative distance between $x_1$ and $x_2$ is restricted to a range of $d$ pixels, i.e. $|x_1 - x_2|_\infty \leqslant d$. It is sufficient to set $d$ as a small value at each level. On the one hand, for coarser levels, a one-pixel motion corresponds to a large motion range at finer levels. On the other hand, as described above, the warping strategy enables a small motion increment at each level, thus a small search field fulfills the estimation process at each level. Instead of building a 4D partial cost volume, the associating costs for each pixel are stored as a vector to build a 3D partial cost volume, which is more appropriate to be passed as input to the estimator network. For the $m$th level, the size of the partial cost volume is $(2d + 1)^2 \times H^m \times W^m$, where $H^m$ and $W^m$ represent the height and width of the feature map at $m$th level.

### 3.1.4 Optical Flow Estimator

The optical flow estimator is a multilayer CNN network that takes as inputs the $m$th level features of the first image, $m$th level cost volume, and up-sampled flow estimation at $m$+1th level to estimate the flow field at $m$th level. Figure 3.3 (a) demonstrates the general network structure, and inputs at pyramid level 2 are taken as example. Although the network structure at different levels is identical, each network is trained individually, which allows the trained network to be more targeted on the estimation at one pyramid level. From the perspective of variational approaches, the function of optical flow estimator is similar to energy minimization which is able to find the optimal optical flow value for each pixel.



**(a)** **(b)**

**Figure 3.3:** **(a)** The optical flow estimator network with inputs at pyramid level 2. Each convolutional layer is followed by a leaky ReLU unit except the last (light green) one that outputs the optical flow [SYLK18]. **(b)** A 5-layer dense block. Each layer takes all preceding feature-maps as input [HLVW17].

Inspiring by the excellent performance of DenseNet [HLVW17] in image classification, the idea of DenseNet connections is also applied in the optical flow estimator network. As shown in Figure 3.3 (b), in a DenseNet block, the output of each convolutional layer is passed as input to all the subsequent convolutional layers. The direct connection between each two layers can realize the feature propagation and accumulation. The reuse of the features already trained allows each layer to learn fewer feature maps, which can reduce the number of trainable parameters while achieving excellent performance. The experiment results of PWC-Net indicate that the PWC-Net with DenseNet connections is 5% more accuracy than without these connections, however 40% slower, since more computation is completed during estimation.

### 3.1.5 Context Network

After estimating the flow field, a post-processing step is applied in the most of optical flow approaches. In this step, the computed flow field is refined by combining more contextual information, such as using a median filter. In PWC-Net, a context network is designed for this purpose, which makes use of dilated convolutions. Instead of only refining the final

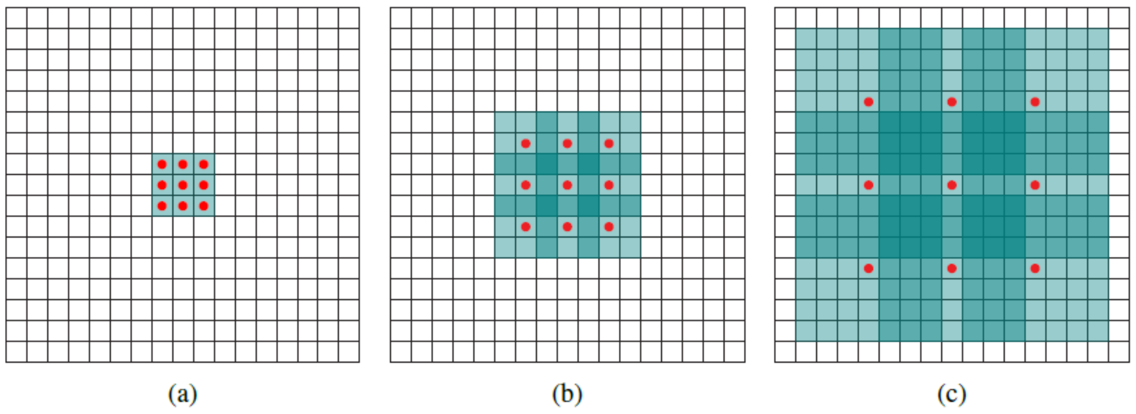


**Figure 3.4:** Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage. Red points denote the filter elements in a dilated convolutional filter. The blue square around each pixel denotes the receptive field in the original image $I_0$. (a) Original image $I_0$ and a convolutional filter with dilation factor $k = 1$; Each element in $I_0$ has a receptive field of $1 \times 1$. (b) Image $I_1$, which is produced from $I_0$ by the 1-dilated convolution and a convolutional filter with dilation factor $k = 2$; Each element in $I_1$ has a receptive field of $3 \times 3$. (c) Image $I_2$, which is produced from $I_1$ by the 2-dilated convolution and a convolutional filter with dilation factor k=4; Each element in $I_2$ has a receptive field of $7 \times 7$. Figure from [YK16].

estimation, the computed flow field at each level can be refined before passed to the next level. The use of the context network is optional in PWC-Net, however, the experiment results indicate a consistent help of applying this network as post-processing [SYLK18].

Dilated convolutions are a special category of the normal convolution, where the convolution filter is filled with different gaps. The size of the filter depends on a so-called dilation factor $k$. In the case that $k$ equals 1, the dilated convolution is identical to the normal convolution. As the factor $k$ increases, the empty space between the filter elements also increases. Systematically applying dilated convolution with different dilation factors can realize the aggregation of multi-scale contextual information by exponentially expanding the receptive field for a pixel, seeing Figure 3.4 [YK16].

Based on this characteristic, the context network possesses 7 convolution layers, as shown in Figure 3.5. The dilation factor for each layer is 1,2,4,8,16,1, and 1, respectively. The network takes as inputs the estimated flow field and the second last layer of the optical flow estimator, where the features are more complex and detailed, and generates a refined flow estimation.

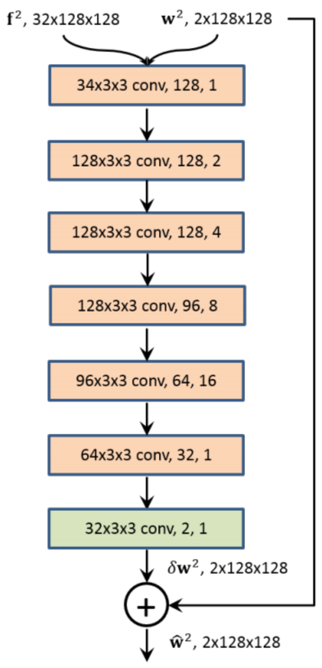


**Figure 3.5:** The context network with inputs at pyramid level 2. Each convolutional layer is followed by a leaky ReLU unit except the last (light green) one that outputs the optical flow. The last number in each convolutional layer denotes the dilation factor $k$ [SYLK18].

### 3.1.6 Training Loss

After explaining the network structure of PWC-Net, we now introduce the definition of training loss. The trainable parameters contained in PWC-Net are from the feature pyramid extractor, the optical flow estimator networks at different pyramid levels, and the context network at different pyramid levels (depending on the option). The warping and cost volume layer contain no learnable parameters. The multiscale loss function used for training is based on the AEE between the ground truth and estimated flow at each pyramid level, and is defined as following:

$$L(\theta) = \sum_{l=l_0}^{L} \alpha_l \sum_x \left| w_l^t(x) - w_l^{e,\theta}(x) \right|_2 + \gamma \, |\theta|_2, \tag{3.3}$$

where $\theta$ represents the set of trainable parameters and $\alpha_l$ is the individual weight for each pyramid level. $w_l^{e,\theta}(x)$ and $w_l^t(x)$ denote the estimated flow vector of pixel $x$ at $l$th pyramid level and its ground truth flow vector, respectively. For a fixed pyramid level $l$, both vectors correspond to the vector $(u_{i,j}^e, v_{i,j}^e)^\top$ and $(u_{i,j}^t, v_{i,j}^t)^\top$ in the AEE Equation 2.7, where $(i, j)$ also represents a pixel location. The first term in the loss function computes the weighted sum of all endpoint errors at each pyramid level, and the second term is to regularize the parameters in $\theta$ in order to avoid overfitting.

During the training process, $l_0$ is set to be 2, and a search range of $d = 4$ pixels is used for building the partial cost volume. The resulted flow field is a quarter of the original resolution, which is further up-sampled to the original resolution using the bilinear interpolation.

For fine-tuning based on different benchmarks, a more robust training loss is used:

$$L(\theta) = \sum_{l=l_0}^{L} \alpha_l \sum_x \left( \left| w_l^t(x) - w_l^{e,\theta}(x) \right| + \epsilon \right)^q + \gamma \, |\theta|_2, \tag{3.4}$$

where $\epsilon$ is a small constant. To reduce the penalty for outliers, the $L_2$ norm in the previous training loss is replaced with the $L_1$ norm and $q$ is set to be less than 1 .

## 3.2 ProFlow

The second algorithm we want to analyze is ProFlow, proposed by Maurer *et al.* [MB18] in 2018. ProFlow takes as inputs three consecutive frames $t-1$, $t$ and $t+1$, and estimates the forward flow of frame $t$. ProFlow is based on the traditional optical flow pipeline, where a shallow CNN network is also embedded, as shown in Figure 3.6. Different from the normal learning-based approaches, where CNN models are pretrained on a large training dataset, ProFlow implements an self-supervised online learning process during estimation, where no ground truth or labeled data are required. The training process is performed individually for each frame of every sequence, and it also takes the location of each training sample into account. As a result, ProFlow possesses a great estimation performance as well as a high adaptability regarding the changes of content between frames and independently moving objects in the same frame.

In the following, we will explain the function of each step in the ProFlow pipeline. The explanations are based on the paper [MB18].

### 3.2.1 Initial Flow Estimation/Baseline

In the first step, a baseline approach is applied to generate the initial backward and forward flow, i.e. the flow fields from frame $t$ to frame $t-1$ and from frame $t$ to frame $t+1$, respectively. Theoretically, any approaches based on two frames can be employed in this step. To generate rather accurate estimation results, a pipeline approach that combines several advanced techniques is used as the baseline approach.

As described in Chapter 1.2, a pipeline approach consists of four steps: matching, filtering, inpainting, and variational refinement. The algorithm Coarse-to-fine PatchMatch (CPM) of Hu *et al.* [HSL16] is used in the matching step, which combines the PatchMatch ideas, i.e. neighboring propagation and random search strategy, with traditional coarse-to-fine schema, seeing Figure 3.7. This algorithm can provide an accurate flow estimation even for tiny

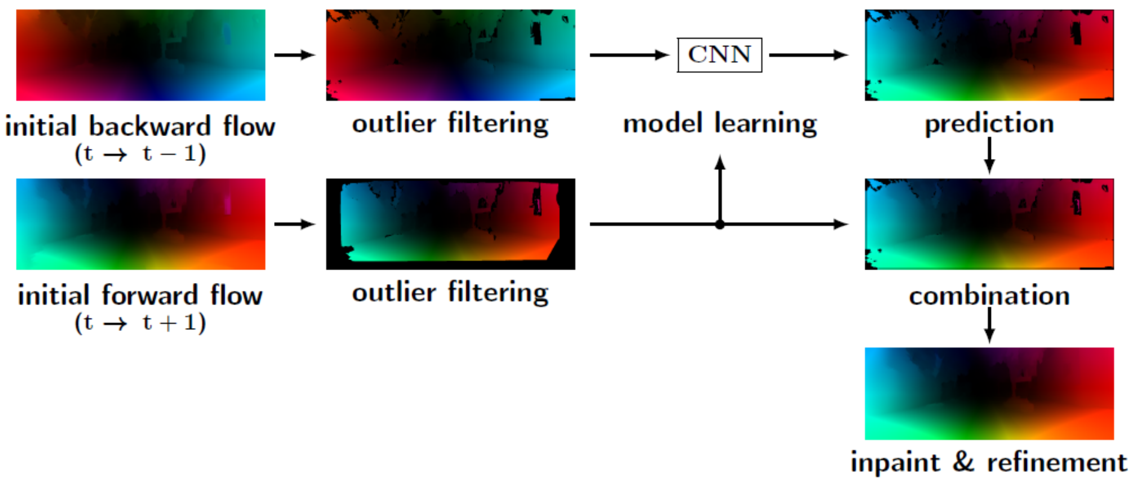


**Figure 3.6:** Overview of the ProFlow pipeline. Figure from [MB18].

structures with large displacements and possesses an implicit regularization effect [HSL16], thus is a great candidate for the first step. For outlier filtering, the standard method bidirectional checking is employed, as introduced in Section 2.3. It can eliminate considerably large amounts of outliers caused by wrong matches. To perform the bidirectional checking, matches in the reverse directions, i.e. from frame $t-1$ to frame $t$ and from frame $t+1$ to frame $t$ also need to be computed in the matching step. The inpainting step makes use of the robust interpolation technique (RIC), proposed by Hu *et al.* [HLS17], to produce dense motion fields. In RIC, the input image is segmented into several non-overlapping superpixels, and for each superpixel an affine motion model is assumed and computed based on the neighbouring blocks. The purpose of the last refinement step is to refine the estimated optical flow to gain a sub-pixel precision. To this end, the order-adaptive illumination-aware refinement (OIC) of Maurer *et al.* [MSB17] is used, which can bring great improvements to the results compared to normal refinement techniques.

### 3.2.2 Outlier filtering

After applying the baseline approach, a filtering step is performed with the aim of removing outliers contained in the initial forward and backward flow estimation. Here, the bidirectional checking is used again. The same as before, the initial flow fields in the reverse directions i.e. from frame $t-1$ to frame $t$ and from frame $t+1$ to frame $t$, need to be computed with the baseline approach. Hence, when using the bidirectional checking in the second filtering step, four flow fields will be generated in the first step. The bidirectional checking is then applied to each pair of forward/backward flow and its reverse flow to filter the corresponding outliers out. In this context, only the consistent flow vectors are remained. After filtering, many outliers can be excluded, especially in the occluded and low textured regions.

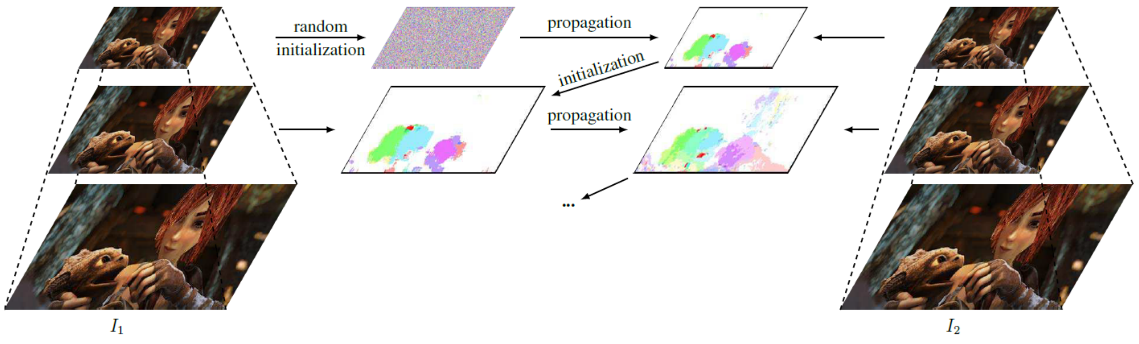


**Figure 3.7:** Overview of the CPM algorithm. The whole process runs from top to bottom. **Left:** Image pyramid of the first input image. **Right:** Image pyramid of the second input image. **Middle:** Operations at each pyramid level including generating initial correspondences, performing neighbouring propagation and random search. Figure from [HSL16].

### 3.2.3 Learning a Motion Model

In this step, a CNN network is trained to learn the vary mappings from backward flow $(t \rightarrow t-1)$ to forward flow $(t \rightarrow t+1)$. Using this network, it is possible to re-estimate the forward flow vectors that are filtered out in the filtering step, if their corresponding backward flow vectors pass the filtering step. Since the occluded pixels are the main source of outliers, the CNN model can especially bring some improvements in the estimation of occluded regions. The CNN network used in ProFlow is quite light-weighted and trained in an self-supervised way. The training process is performed during the flow estimation and only depends on the flow fields computed in the first two steps, without using any ground truth flow. Moreover, a completely new CNN model is trained for each input triplet images of the ProFlow pipeline, thus, each frame in a sequence can possess an individual motion model. In the following, we will describe the generation of the training dataset and the structure of the CNN network.

The training dataset is build based on the filtered forward and backward flow fields. To guarantee the correctness of training samples, only the locations, where both forward and backward flow pass the filtering step, are regarded as candidates of training samples. However, if all these candidates are added in the training set, in the case of very dense filtered forward and backward flow, the training set may be overlarge. Thus, a sampling operation is performed to make the training set achieve a reasonable size. To this end, a grid spacing of 10 pixels is used for sampling the training candidates equidistantly. In order to give a context for each candidate, a 7×7 patch centered with the candidate pixel is used. For each pixel in the patch, the following data are stacked together: (i) the backward flow components $u^{bw}$ and $v^{bw}$ (ii) a validity flag {0,1} indicating if this pixel is a valid training candidate. (iii) the normalized pixel coordinates $x$ and $y$ in the range of [-1,1]×[-1,1]. Thereby, the dimension of a training sample is 7×7×5, as shown in Figure 3.8 (a). The

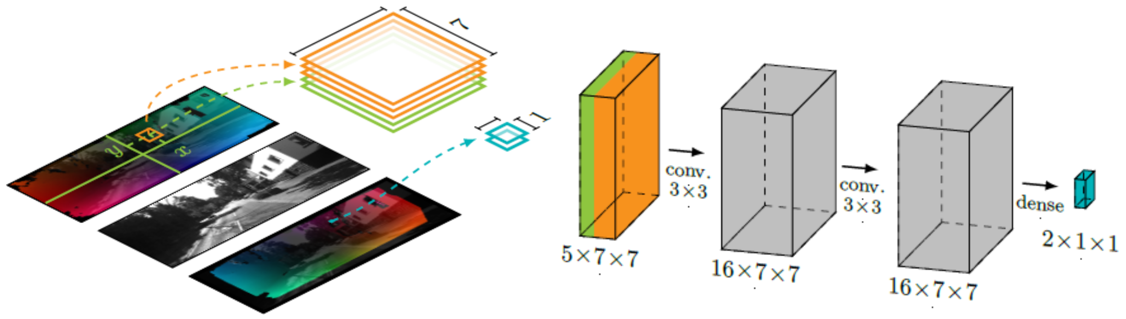


**Figure 3.8: Left:** Training sample extraction. **Right:** CNN network architecture. Both figures from [MB18].

training reference is the forward flow components $u^{fw}$ and $v^{fw}$ of the center pixel in a patch. The whole extraction process is performed automatically after the filtering of forward and backward flow.

After extracting the training data, a CNN network is trained to learn the relation between inputs and outputs. The network contains three layers, i.e. 2 convolutional layers and 1 fully connected layer, as shown in Figure 3.8 (b). In both convolutional layers, a 3×3 filter with 16 feature channels is applied to generate complex non-linear features. The fully connected layer produces a 2-dimensional vector, which corresponds to the predicted forward flow components $u^{fw}$ and $v^{fw}$. The loss function considers the absolute difference between the predicted flow vector and the actual flow vector computed by the baseline approach. Since the architecture of the network is relatively simple, the training process can be finished into 20 seconds. Normally, it takes only 3000-4000 training steps, depending on the setting of the learning rate.

As introduced in the previous paragraphs, each training sample contains not only backward flow and validity, but also the pixel location. This enables the network to learn a location dependent model, i.e. different motion patterns can be learned at different locations in a frame. The location dependent model can especially contribute to the motion estimation of individually moving objects in a frame, and improve the estimation accuracy when non-rigid deformation or perspective effect occurs [MB18].

### 3.2.4 Combination and Final Estimation

After training the neural network, the learned model is applied to all locations, where the forward flow is filtered but backward flow is reserved, to predict their forward flow. The predictions are combined with the filtered forward flow computed in the second step to produce a denser forward flow estimation. After the combination, there can still exist some locations where neither forward flow nor backward flow are available. Thereby, the inpainting algorithm RIC is employed again with the purpose of densifying the flow field. Afterwards, the dense flow is refined using the algorithm OIR. Both are the same as in the baseline approach.

# 4 Combining PWC-Net and ProFlow

After introducing the two top-performing approaches PWC-Net and ProFlow, in this chapter we conduct the combination of these two algorithms. In Section 4.1, we first integrate the PWC-Net into the ProFlow pipeline to generate a new pipeline which is then evaluated in Section 4.2. The evaluation is based on two most widely used benchmarks MPI-Sintel and KITTI 2015. By analyzing the evaluation results, we discuss the problems existing in the new pipeline and their causes. Afterwards, different modification ideas are proposed to lay a foundation for the next chapter.

## 4.1 Pipeline Update

Figure 4.1 (a) demonstrates an overview of the individual steps in the ProFlow pipeline, where the baseline approach composes of four methods, i.e. CPM, bidirectional filtering, RIC, and OIR. Since the baseline approach is used to generate dense and accurate initial flow with two consecutive frames as input, we replace it with the advanced PWC-Net. The same as with the baseline approach, PWC-Net is also used to generate both forward and backward flow and their reverse flow fields, which are necessary to perform the bidirectional filtering. The new generated pipeline is referred to as PWC-ProFlow and represented in Figure 4.1 (b).

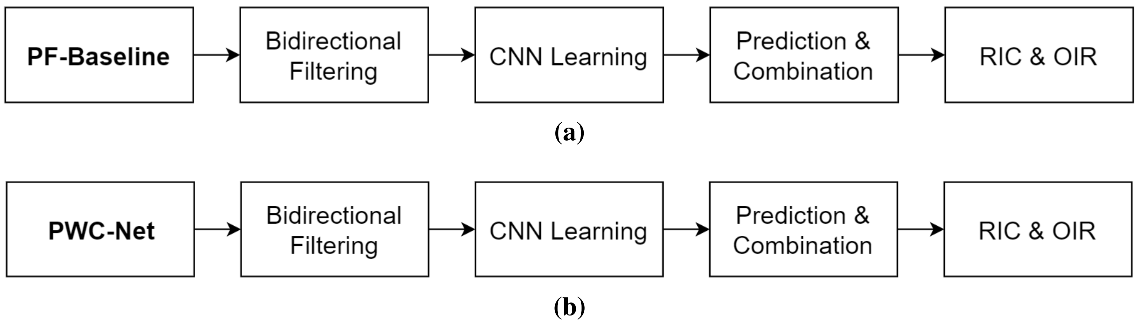


**Figure 4.1:** **(a)** Overview of the original ProFlow pipeline. **(b)** Overview of the new generated PWC-ProFlow pipeline.

## 4.2 Evaluation

After presenting the structure of the new PWC-ProFlow, we now evaluate the new pipeline through some experiments and analyze its performance.

### 4.2.1 Summary of Experiments

#### Experiments Introduction

As experiment candidates, we take the baseline approach used in the ProFlow pipeline, PWC-Net, ProFlow, and the new created method PWC-ProFlow. Further, the two most widely used benchmarks MPI Sintel [BWSB12] and KITTI 2015 [MG15] are applied to perform our evaluation. The Sintel dataset contains 23 highly challenging scenes related to large displacements, illumination changes, and significant occlusions. Each scene contains 50 consecutive frames. The KITTI dataset is taken from 200 different real road scenes, and most scenes contain cars in various sizes, shadows, lighting changes, and different backgrounds. Here, we only use the training dataset of both benchmarks, since the ground truth flow of each frame is available for evaluation. In particular, two types of ground truth are provided in KITTI dataset. The one is *noc*, which only contains the ground truth flow of non-occluded pixels. The other is *occ*, which contains the ground truth flow vectors of all pixels including both occluded and non-occluded pixels. Hence, in the following analysis, we use the name *noc* and *occ* to represent the evaluation of non-occluded pixels and all pixels of KITTI dataset, respectively.

#### Experiment Results Analysis

Table 4.1 presents the experiment results of the experiment methods on both datasets. We first compare the two baseline approaches used in ProFlow and PWC-ProFlow, i.e. the PF-baseline and PWC-Net. As one can see, PWC-Net outperforms the PF-baseline approach on both datasets, with a 17% lower AEE on the Sintel dataset, and a 45% and 55% lower AEE in the noc and occ cases of the KITTI dataset, respectively. As PWC-Net can realize a real-time estimation, it can also bring a higher runtime efficiency. Moreover, we can observe that ProFlow also outperforms its baseline approach on both datasets, especially in the occ case of the KITTI dataset, where ProFlow achieves a 24% lower AEE w.r.t. PF-baseline. This comparison demonstrates that the additional four steps in the ProFlow pipeline can bring great improvements to the estimation of the baseline approach.

Based on the above observations, it is worth expecting that PWC-ProFlow could gain a better estimation performance than PWC-Net and ProFlow, since a more advanced baseline approach is employed in the pipeline. However, as shown in Table 4.1, PWC-ProFlow ranks the third place in the estimation of the Sintel dataset, achieving a 18% higher AEE than PWC-Net and ProFlow. Further, considering the AEE, PWC-ProFlow places the second

**Table 4.1:** Average endpoint error (AEE) and the percentage of bad pixels (BP) of different methods on the training set of KITTI 2015 and MPI Sintel (clean path). PF-baseline refers to the baseline approach used in the ProFlow pipeline. One pixel with an endpoint error > 3px is defined as a bad pixel.

| **Method** | **KITTI 2015** | | | | **Sintel** |
|---|---|---|---|---|---|
| | *noc* | | *occ* | | clean |
| | AEE | BP (%) | AEE | BP (%) | AEE |
| PF-baseline | 3.49 | 12.22 | 7.30 | 20.33 | 2.36[1] |
| ProFlow | 3.16 | 11.40 | 5.56 | 17.83 | 1.98 |
| PWC-Net | 1.89 | 8.52 | 3.25 | 13.68 | 1.95[1] |
| PWC-ProFlow | 2.89 | 11.52 | 5.18 | 19.03 | 2.34 |

in both noc and occ cases of the KITTI dataset, with 9% and 7% less AEE comparing to the ProFlow, which presents certain improvements brought by applying a better baseline approach. Besides, PWC-ProFlow achieves a better runtime efficiency than ProFlow due to the real-time estimation of PWC-Net. Summarizing the experiment results, the comparison between PWC-Net and PWC-ProFlow turns out that the additional steps in the PWC-ProFlow pipeline may deteriorate the initial estimation computed by the PWC-Net. The underperformance of PWC-ProFlow can be observed on both datasets. In order to investigate the problems hidden in the PWC-ProFlow pipeline, we analyze the experiment data of each scene in both datasets. In the following, we will discuss the improvements and problem causes based on our analysis.

### 4.2.2 Improvements

After checking the AEE per scene of the Sintel dataset, we discover that PWC-ProFlow achieves a better accuracy in 13 scenes than PWC-Net and 11 scenes than ProFlow. It manifests that PWC-ProFlow is able to improve the estimation accuracy in many circumstances. Several examples confirm this observation by showing that PWC-ProFlow effectively improves the inaccurate estimations of PWC-Net and ProFlow especially in the occluded regions, seeing the right up corner in Figure 4.2 (a), the left boundary area in Figure 4.2 (b). The same improvements can be detected in the KITTI dataset, seeing the right boundary in Figure 4.3 (a). In addition to occluded regions, PWC-ProFlow also provides great improvements for the estimation under lighting changes, such as in the head of the dragon in Figure 4.3 (b). In these positive examples, the inaccurate estimations of the PWC-Net are filtered out in the filtering step, as shown in the value map of filtered forward flow. After online learning, the trained CNN model can help to predict the more accurate forward flow

[1]In order to unify the number of test frames used for methods with two and triplet images as input, the error of flow estimation of the first frame is not included.

vectors $(t \rightarrow t+1)$ in the filtered locations by virtue of existing filtered backward flow $(t \rightarrow t-1)$. After the inpainting and refinement, the estimated flow field can be further improved.

### 4.2.3 Problem Analysis

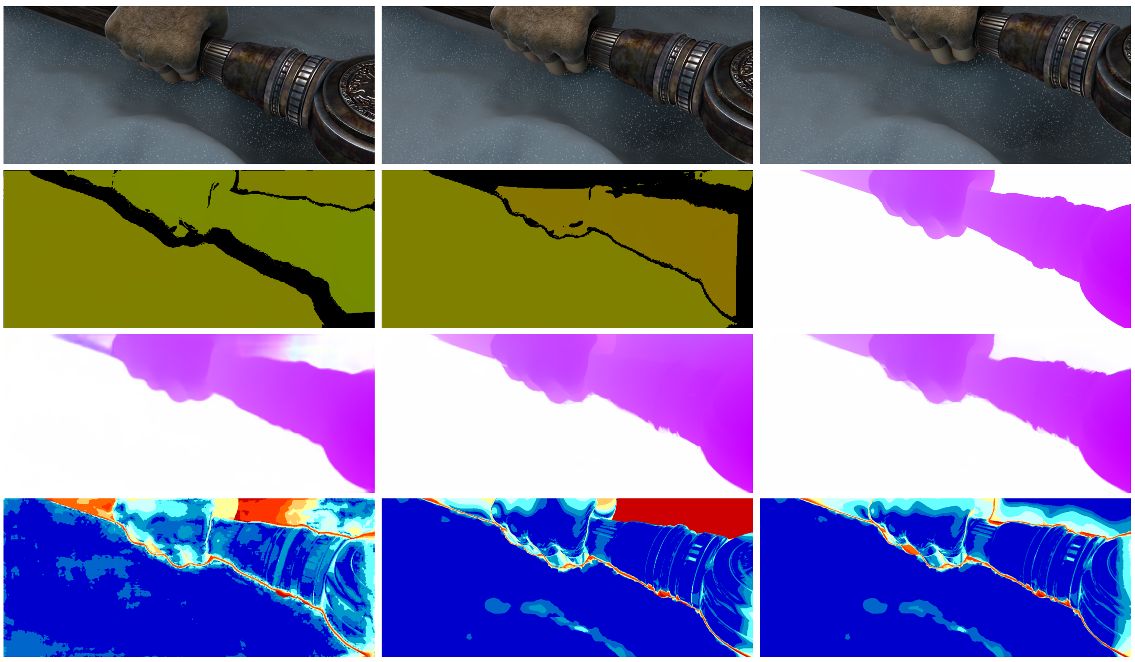
Even though many examples demonstrate the great performance of PWC-ProFlow, its high AEE on both datasets still indicates that in some situations the PWC-ProFlow achieves a very poor performance. By analyzing the negative samples in the Sintel dataset, we discover two main problems, i.e. large gaps contained in the input flow of the inpainting step and wrong predictions produced by the CNN model. Both problems can be caused by a same situation, namely in a certain region of the filtered forward flow $(t \rightarrow t+1)$, which usually corresponds to an individual local motion pattern, there exist no or very few flow vectors. In the case that the same few flow vectors exist in this region in the filtered backward flow $(t \rightarrow t-1)$, the predictions that can be made by the CNN network and thus combined with the filtered forward flow are very few. If the region takes up a large proportion of the whole flow field, it can lead to a large gap in the input flow of the inpainting step, as shown in the bottom half of Figure 4.4 (a). Since most of the inpainting approaches interpolate the empty places based on their neighboring information, large gaps can severely deteriorate the inpainting result and in turn the final estimation.

In the case that the filtered backward flow $(t \rightarrow t-1)$ contains sufficient flow vectors in this region, the problem regarding inaccurate predictions can still arise and destroy the estimation result. The reason is that the training set of the CNN model includes only a small amount of training samples from this region, since a training sample needs to possess both a valid forward $(t \rightarrow t+1)$ and backward flow $(t \rightarrow t-1)$. Thus, it is unlikely for the CNN network to correctly learn the motion pattern located in this region. For the existing backward flow vectors $(t \rightarrow t-1)$ in this region, the CNN model would possibly apply the neighboring patterns to predict the corresponding forward flow $(t \rightarrow t+1)$. As shown in Figure 4.4 (b), the predictions in the leg area are made based on the pattern located in the arm. Due to the different moving directions in these two areas, the flow vectors in the leg area are predicted with a significantly low accuracy.

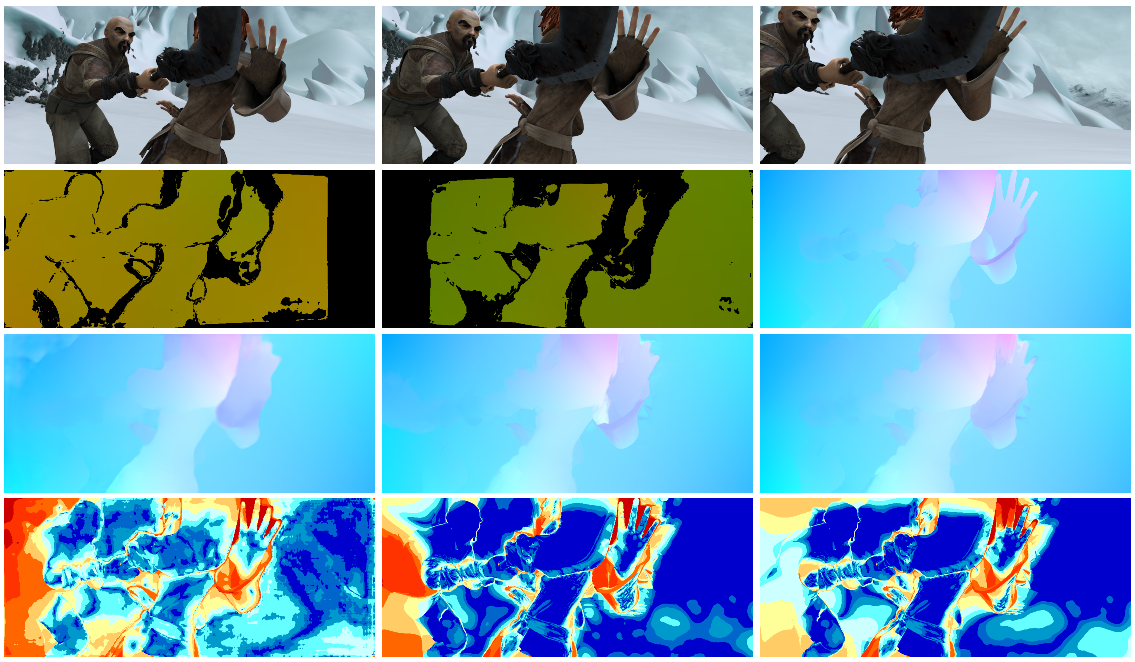
The two problems also occur in the estimation of KITTI dataset. However, by analyzing the negative samples in the KITTI set, we find another problem, i.e. the predictions at the contour of the cars normally possess a very low accuracy, especially for small cars. Figure 4.5 presents one example, seeing the upper contour of the left car. As we can see, the CNN model treats these positions as parts of the background when performing the predictions. For individual small cars, it is difficult for the CNN network to learn their patterns precisely due to the small amount of training samples from these areas. Hence, the predictions at the contour may be affected by the adjacent neighboring background flow. Since we would like to maintain the way creating and training the CNN model, this problem seems unavoidable in our pipeline approach. Thus, the focus of the modification will be mainly on the above two problems. After analysis we suggest the following modification ideas.

On the one hand, we can focus on improving the inpainting result, such as find a way to reduce the large gaps in its input flow, or find an appropriate alternative of the inpainting step which is also capable of densifying the flow field and preparing the input for the refinement step. On the other hand, we can attempt to filter less flow vectors in the filtering step, especially in the forward filtering. Compared to the baseline approach used in the ProFlow pipeline, PWC-Net is specifically pretrained for the forward flow prediction. Due to the lack of a backward training, there is no guarantee to the accuracy of the estimated backward flow $(t + 1 \rightarrow t)$. Since the bidirectional filtering makes use of flow vectors in both directions, inaccurate predictions in the backward direction $(t + 1 \rightarrow t)$ can cause the filtering of some accurate forward flow vectors $(t \rightarrow t + 1)$. Based on this characteristic of PWC-Net, we can replace the bidirectional filtering with other filtering approaches which use flow field in only one direction for the filtering to see if more flow vectors can be remained in the filtered forward flow.

Based on these ideas, different modifications of the PWC-ProFlow pipeline will be introduced in the next chapter.



**(a)**



**(b)**

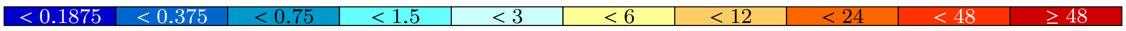
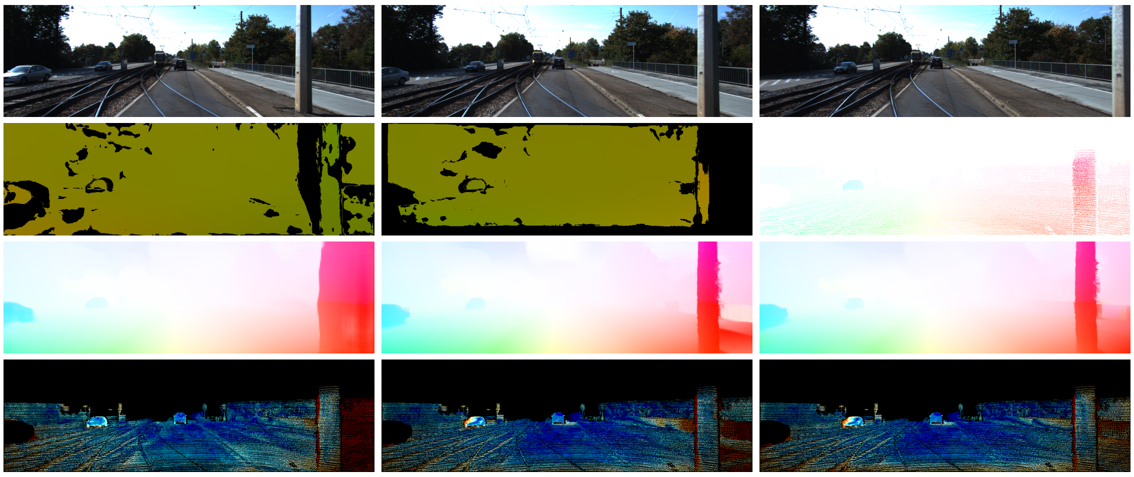
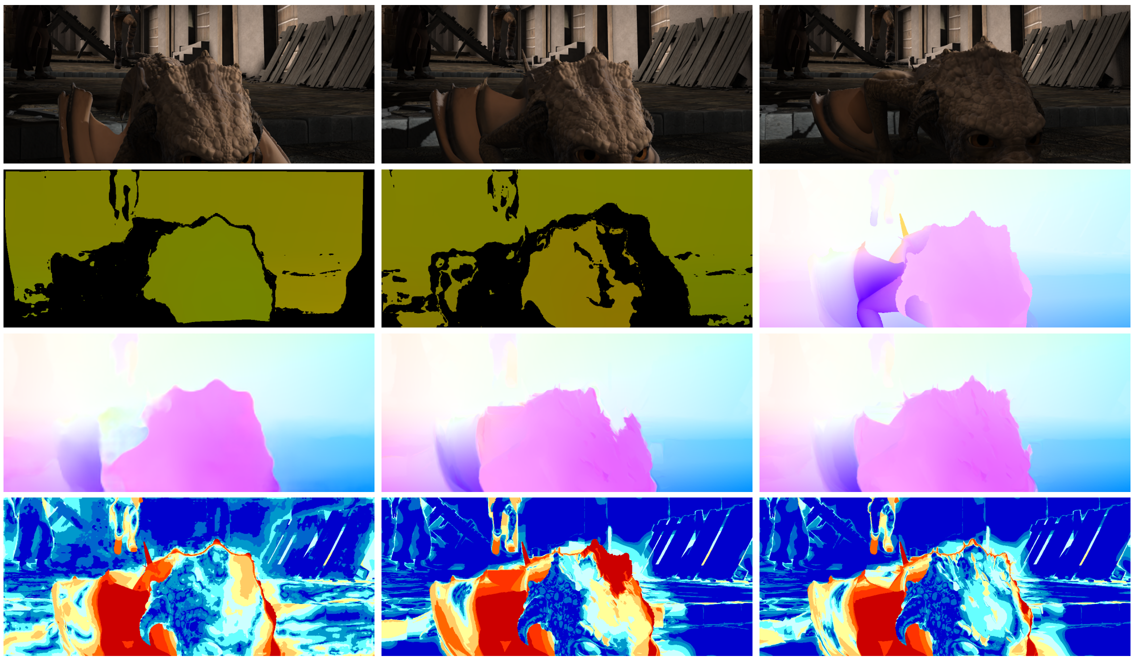


**Figure 4.2:** Visualization of improvements in occluded regions (Sintel dataset ambush_7 #41 and ambush_2 #4 [BWSB12]). **For both figures: First row:** Previous, reference, and next frame. **Second row:** Value map of filtered backward flow and forward flow, ground truth flow. **Third and fourth row:** Flow estimation of the reference frame and bad pixel visualization. **From left to right:** PWC-Net, ProFlow, and PWC-ProFlow.



**(a)**



**(b)**

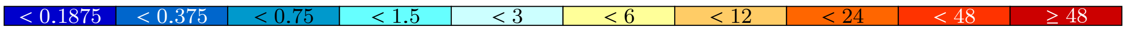
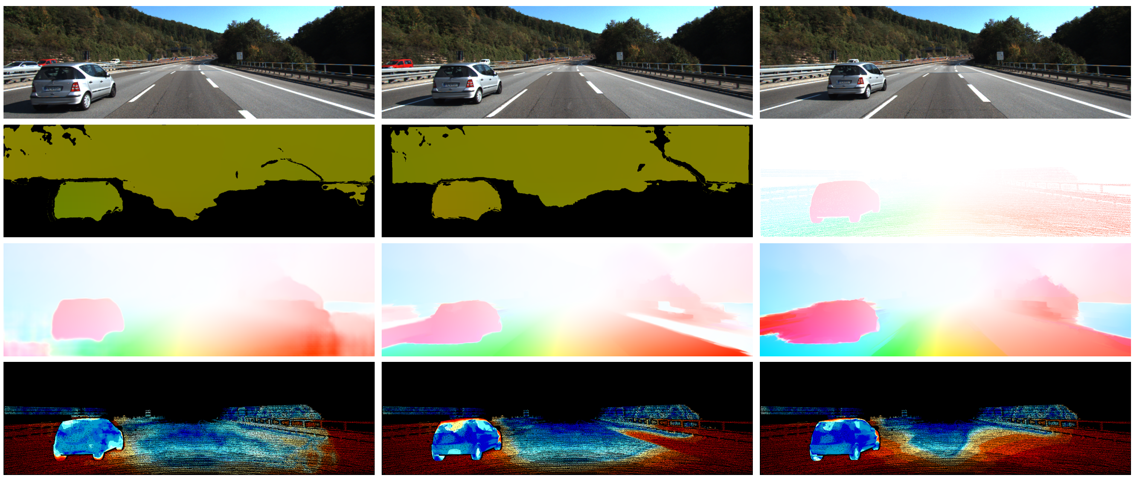
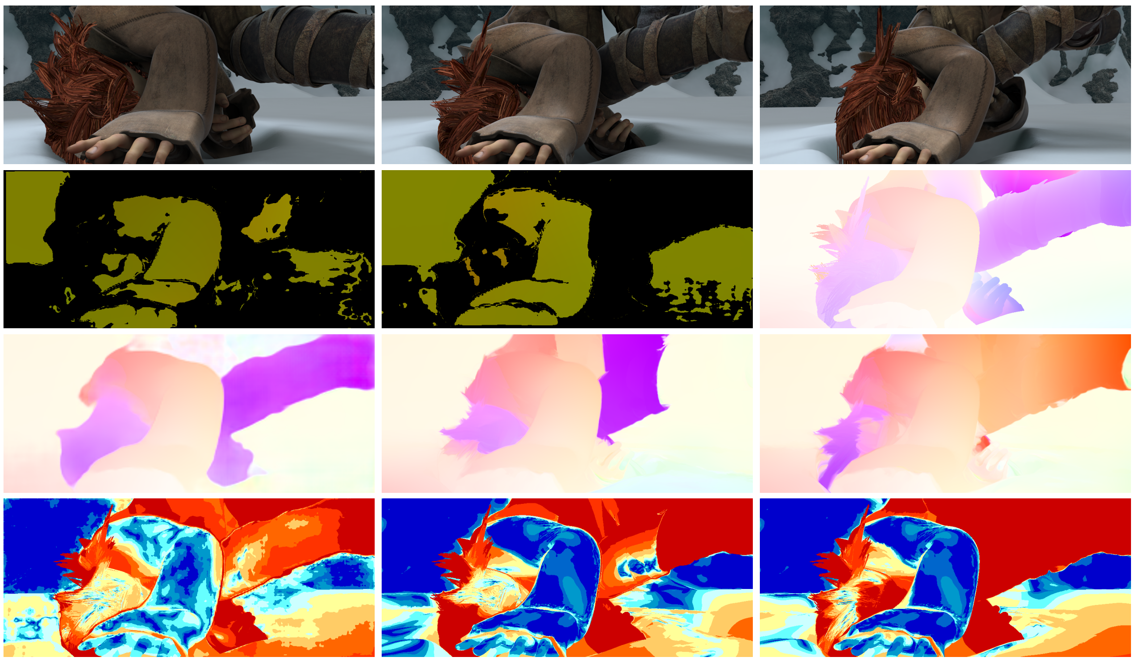


**Figure 4.3:** Visualization of improvements in occluded regions (KITTI dataset occ #145 [MG15]) and lightning changes (Sintel dataset market_5 #8 [BWSB12]). **For both figures: First row:** Previous, reference, and next frame. **Second row:** Value map of filtered backward flow and forward flow, ground truth flow. **Third and fourth row:** Flow estimation of reference frame and bad pixel visualization. **From left to right:** PWC-Net, ProFlow, and PWC-ProFlow.



(a)



(b)

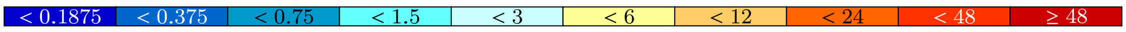


**Figure 4.4:** Visualization of negative performance due to large gaps in the input flow of the inpainting step (KITTI dataset occ #188 [MG15]) and due to wrong predictions (Sintel dataset ambush_4 #7 [BWSB12]). **For both figures: First row:** Previous, reference, and next frame. **Second row:** Value map of filtered backward flow and forward flow, ground truth flow. **Third and fourth row:** Flow estimation of reference frame and bad pixel visualization. **From left to right:** PWC-Net, ProFlow, and PWC-ProFlow.

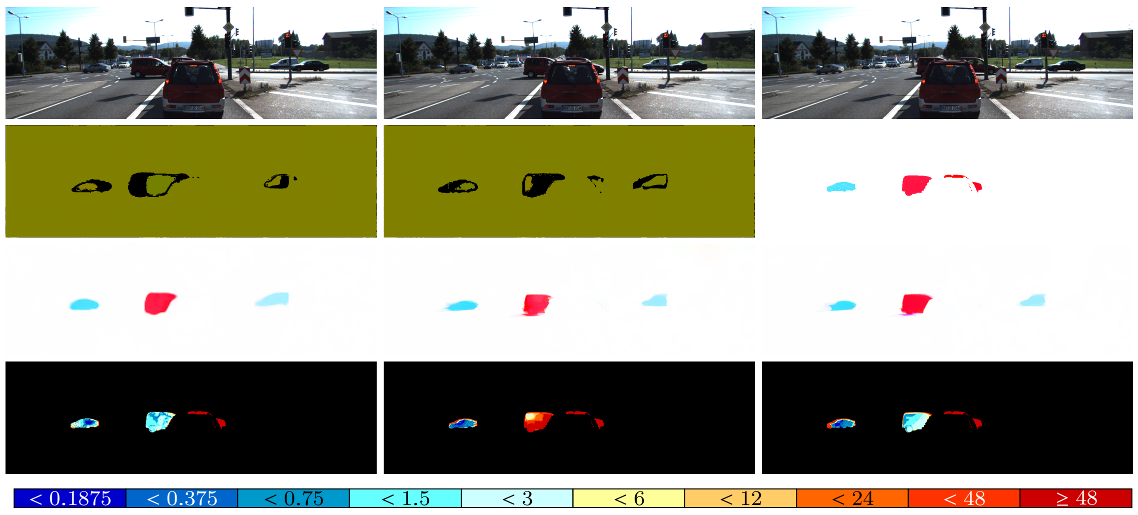


**Figure 4.5:** Visualization of negative performance due to wrong predictions at the contour (KITTI dataset occ #114 [MG15]). **First row:** Previous, reference, and next frame. **Second row:** Value map of filtered backward flow and forward flow, ground truth flow. **Third and fourth row:** Flow estimation of reference frame and bad pixel visualization. **From left to right:** PWC-Net, ProFlow, and PWC-ProFlow.

# 5 Modification

In the previous chapter, we analyze the reasons that can lead to an underperformance of the PWC-ProFlow pipeline. Based on the analysis, six different modifications are created in this chapter with the purpose of increasing the estimation accuracy. In Section 5.1 and 5.2, the first two modifications are designed based on the idea of improving the inpainting result. Section 5.3 introduces the other four modifications, where new filtering algorithms are adopted to replace the original bidirectional filtering. All the new generated pipelines are evaluated on the Sintel and KITTI datasets. Based on the experiment results, we further analyze their estimation performance and compare their estimation results with PWC-ProFlow, PWC-Net, and ProFlow.

## 5.1 Modifying the Input of Inpainting

### 5.1.1 Model Generation

As analyzed in Section 4.2, large gaps contained in the input flow of the inpainting step can deteriorate the inpainting result due to the lack of neighbouring information, and thus lead to a poor final estimation. The most intuitive way to deal with this problem is to sample flow vectors from the initial forward flow computed by the PWC-Net in the first step to the current input flow. By inserting more flow vectors, we intend to provide more available information for the inpainting process. Based on the good performance of PWC-Net, we assume that inpainting using some of the filtered flow vectors of PWC-Net can still be hypothetically better than inpainting using flow vectors very far away. The first modification is referred to as PWC-ProFlow-S and the new generated pipeline is shown in Figure 5.1.

For the purpose of sampling, we employ a grid checking process. In this process, a grid with a predefined size slides over the filtered forward flow. Meanwhile, the number of flow vectors inside the grid is counted. If the number is less than a threshold, the initial forward flow will be sampled equidistantly to fill the blank area in the grid. In the case that

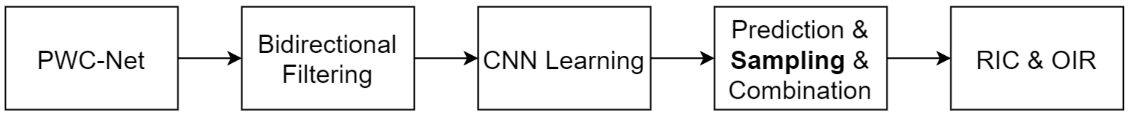


**Figure 5.1:** Overview of the pipeline of PWC-ProFlow-S.

**Table 5.1:** Average endpoint error (AEE) and the percentage of bad pixels (BP) of different methods on the training set of KITTI 2015 and MPI Sintel (clean path). One pixel with an endpoint error > 3px is defined as a bad pixel.

| **Method** | **KITTI 2015** | | | | **Sintel** |
|---|---|---|---|---|---|
| | *noc* | | *occ* | | clean |
| | AEE | BP (%) | AEE | BP (%) | AEE |
| ProFlow | 3.16 | 11.40 | 5.56 | 17.83 | 1.98 |
| PWC-Net | 1.89 | 8.52 | 3.25 | 13.68 | 1.95 |
| PWC-ProFlow | 2.89 | 11.52 | 5.18 | 19.03 | 2.34 |
| PWC-ProFlow-S | 2.16 | 9.10 | 3.65 | 15.92 | 1.82 |

a sampled value and a predicted value, generated by the CNN model for this position, exist simultaneously, a weighted sum of both is calculated as the result for this position. The weight of the predicted value depends on the total number of flow vectors in the current grid. Less flow vectors correspond to a lower potential of a correct prediction, thus less weight will be given to the predicted value. The sum of both weights remains to be 1. We also tried to directly take the predicted value as the result instead of calculating the weighted sum. However, we find using the weighted sum can help to reduce the negative impact of the wrong predictions. A later example can demonstrate this advantage.

### 5.1.2 Evaluation

In the experiments, we define the grid size as $200 \times 100$ for the Sintel dataset and $300 \times 100$ for the KITTI dataset in order to keep a similar ratio as their training images. The threshold is set to be 0.5, meaning that when less than a half flow vectors exist in the grid, the sampling process will be triggered. The sampling distance in both horizontal and vertical directions is 3. The evaluation results are presented in Table 5.1.

**In comparison with PWC-ProFlow.** As shown in Table 5.1, the new pipeline PWC-ProFlow-S achieves a better estimation accuracy than PWC-ProFlow on both datasets, with a 22% lower AEE on the Sintel dataset, a 26% and 30% lower AEE in the noc and occ cases of the KITTI dataset, respectively. The test outcome indicates that appropriate handling of large gaps before inpainting can effectively improve the estimation results, especially in some scenes of the Sintel dataset, e.g. the AEE of ambush_4 decreases from 11.3 to 7.6. Figure 5.2 (a) and (b) present the results of PWC-ProFlow-S regarding the negative estimation examples of PWC-ProFlow mentioned in Chapter 4. Here, we can observe significant improvements in the estimation of the leg area in subfigure (a) and the bottom edge area in subfigure (b), where large gaps exist in the input flow of the inpainting step in PWC-ProFlow. Besides, the wrong predictions in the leg area in subfigure (a) are also treated in a proper way during sampling. The same great progress can also be obtained when compared to the ProFlow. As the quality of the inpainting results enhances, the advantages of using an advanced baseline approach are demonstrated.

**In comparison with PWC-Net.** As shown in Table 5.1, the new pipeline PWC-ProFlow-S outperforms the PWC-Net in the Sintel dataset with a 7% lower AEE. It manifests that the additional steps in the PWC-ProFlow-S pipeline can provide a general improvement for the estimation of PWC-Net on the Sintel dataset. Figure 5.3 (a) and (b) present two examples to show the improvements, seeing the lower-right area in subfigure (a) and the right edge area in subfigure (b). However, in the case of KITTI dataset, PWC-ProFlow-S performs worse than the PWC-Net. By analyzing the negative samples, we find that one possible reason is still due to the inaccurate predictions at contours of the small cars, as explained in Chapter 4. Another reason also stems from the predictions of very small cars. While grid checking, the areas of some small cars may contain no flow vectors, however, because of their small sizes, the total number of non-existing flow vectors inside the grid is still lower than a half. Hence, the sampling process will not be initiated. Figure 5.4 demonstrates one of the examples, where no flow vectors are sample to the second car on the left, thus, the inpainting result and the final estimation in this area are inaccurate.

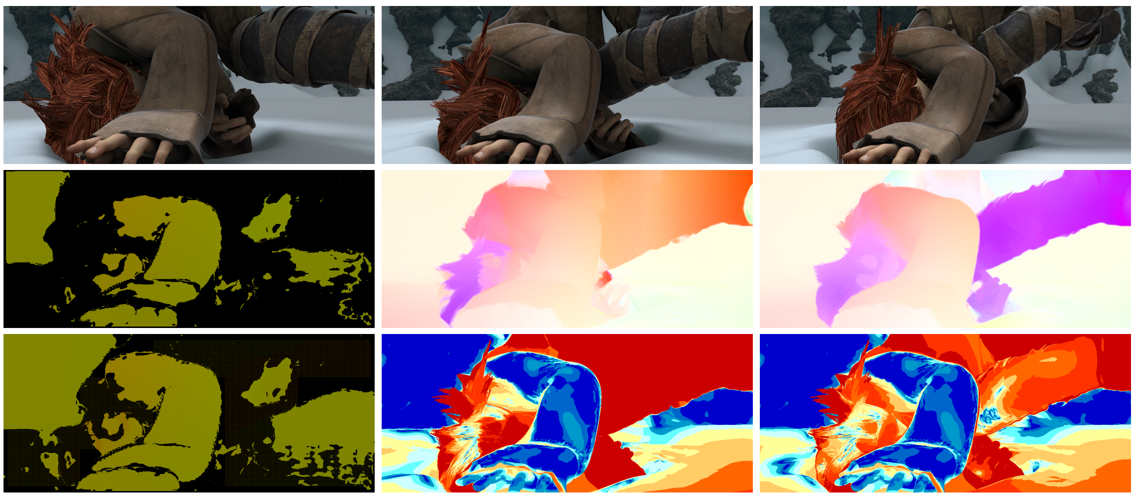
## 5.2 Replacing Inpainting with Dense Filling

### 5.2.1 Model Generation

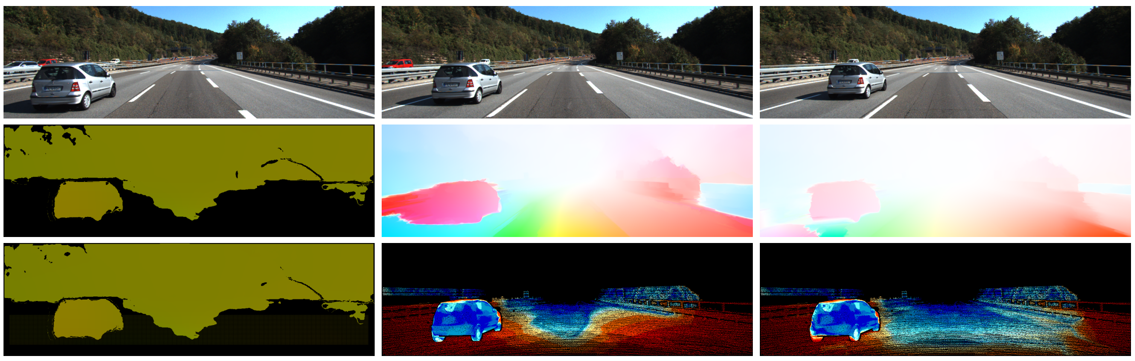
The idea of the second modification is also based on improving the inpainting results. However, instead of remedying the input flow of the inpainting step, we replace the inpainting step with another approach that is also able to densify the flow field. Considering the great performance of PWC-Net, we apply a method called dense filling, which is implemented as following: after combining the CNN predictions, the initial forward flow vectors are sampled to all blank positions, where no forward flow exists. The problem regarding to large gaps is trivially tackled in this modification. The new pipeline with dense filling is referred to as PWC-ProFlow-D and shown in Figure 5.5. The second to fourth step together can be regarded as a corrector of the potentially erroneous flow vectors contained in the initial forward flow. The experiment results are presented in Table 5.2.

### 5.2.2 Evaluation

**In comparison with PWC-ProFlow.** As shown in Table 5.2, the new generated pipeline PWC-ProFlow-D outperforms the PWC-ProFlow on both datasets, with a 22% lower AEE on Sintel dataset, a 37 % and 38 % lower AEE in the noc and occ cases of the KITTI dataset, respectively. Actually, we can find a general decrease of AEE per scene in both datasets. The test outcome proves that the dense filling is a great alternative of the inpainting step



(a)



(b)

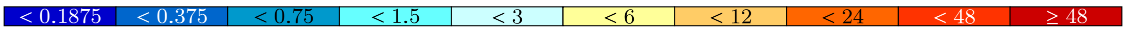


**Figure 5.2:** Visualization of estimation improvements comparing with PWC-ProFlow (Sintel ambush_4 #7 [BWSB12] and KITTI occ #188 [MG15]). **For both figures: First row:** Previous, reference, and next frame. **Second row:** Value map of the input flow of the inpainting step in PWC-ProFlow, flow estimation of PWC-ProFlow and PWC-ProFlow-S. **Third row:** Value map of the input flow of the inpainting step in PWC-ProFlow-S, bad pixel visualization of PWC-ProFlow and PWC-ProFlow-S.

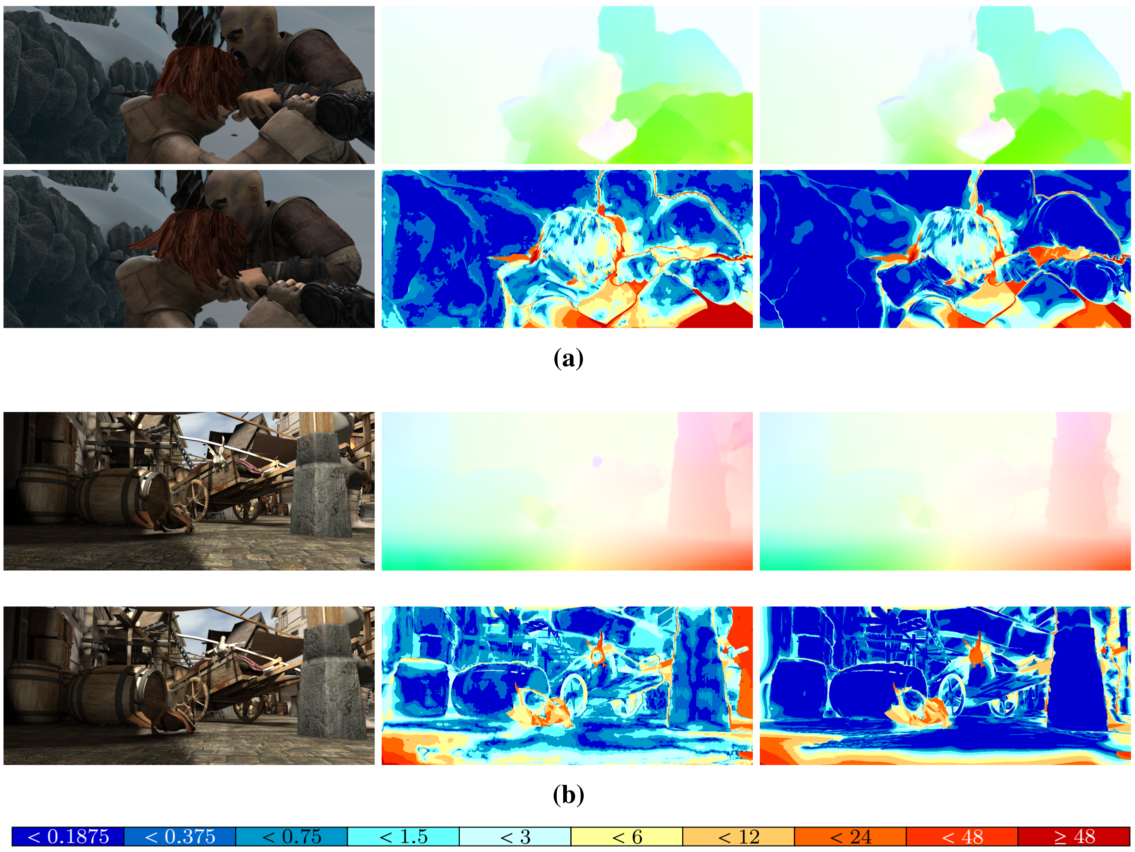


**Figure 5.3:** Visualization of estimation improvements comparing with PWC-Net (Sintel ambush_5 #30 and market_6 #10 [BWSB12]). **For both figures: First row:** Reference frame, estimation results of PWC-Net and PWC-ProFlow-S. **Second row:** The next frame and bad pixel visualization of PWC-Net and PWC-ProFlow-S.

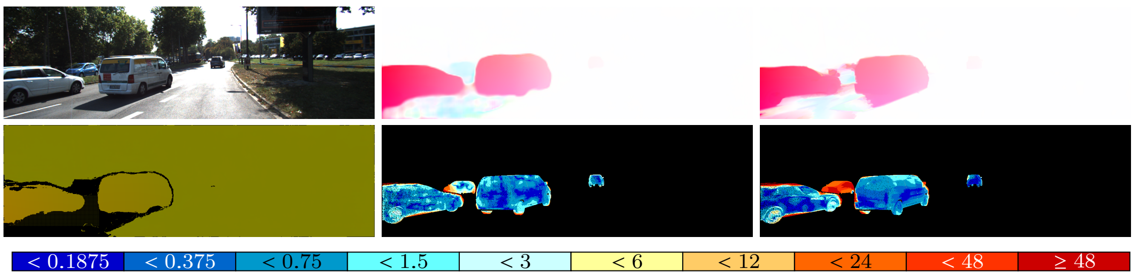


**Figure 5.4:** Visualization of negative estimation samples in KITTI dataset due to non-sampling of small cars (KITTI benchmark occ # 93 [MG15]). **First row:** Reference frame, estimation results of PWC-Net and PWC-ProFlow-S. **Second row:** Value map of the input flow of the inpainting step in PWC-ProFlow-S, bad pixel visualization of PWC-Net and PWC-ProFlow-S.

**Table 5.2:** Average endpoint error (AEE) and the percentage of bad pixels (BP) of different methods on the training set of KITTI 2015 and MPI Sintel (clean path). One pixel with an endpoint error > 3px is defined as a bad pixel.

| **Method** | **KITTI 2015** | | | | **Sintel** |
|---|---|---|---|---|---|
| | *noc* | | *occ* | | clean |
| | AEE | BP (%) | AEE | BP (%) | AEE |
| ProFlow | 3.16 | 11.40 | 5.56 | 17.83 | 1.98 |
| PWC-Net | 1.89 | 8.52 | 3.25 | 13.68 | 1.95 |
| PWC-ProFlow | 2.89 | 11.52 | 5.18 | 19.03 | 2.34 |
| PWC-ProFlow-D | 1.83 | 8.51 | 3.23 | 14.38 | 1.82 |

and is capable of providing a better input for the final refinement step. Figure 5.6 (a) and (b) demonstrate the estimation results of PWC-ProFlow-D regarding the negative samples of PWC-ProFlow listed in Chapter 4. Again, the estimation in the leg area in subfigure (a) and the bottom edge area in subfigure (b) are effectively improved. However, as one can observe, in this modification, the wrong predictions in the leg area in subfigure (a) still deteriorate parts of the estimation result. It indicates that different from the first modification, there is no chance to correct the wrong predictions in PWC-ProFlow-D.

**In comparison with PWC-Net.** Except the BP in the occ case of the KITTI dataset, PWC-ProFlow-D performs better than PWC-Net on both datasets, with a 7% lower AEE on the Sintel dataset and slight improvements on the KITTI dataset. The improvements brought by this modification confirm that the middle steps in the pipeline can effectively improve some of the inaccurate estimations of PWC-Net. Figure 5.7 present some of the examples, where estimation improvements can be detected in the upper left corner of subfigure (a), the upper right edge area of subfigure (b), and left edge area of subfigure (c). The problems that cause the underperformance of the first modification PWC-ProFlow-S in KITTI dataset are automatically solved in the process of dense filling. Hence, an improvement in estimation accuracy on the KITTI dataset can be found when comparing the PWC-ProFlow-D with PWC-ProFlow-S.

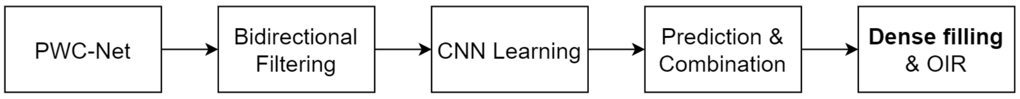
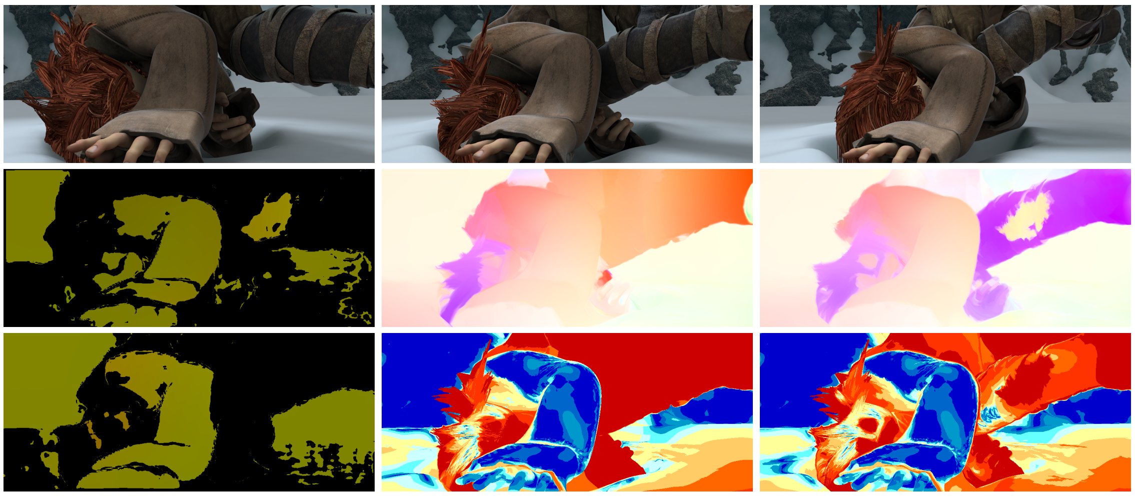
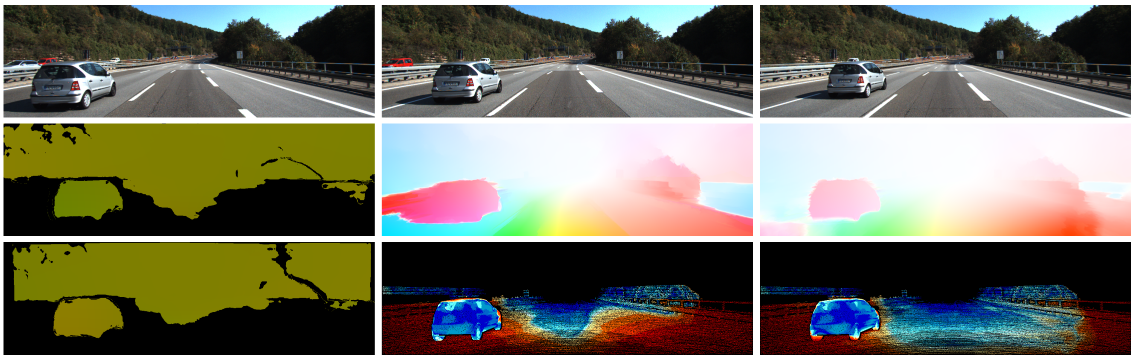


**Figure 5.5:** Overview of the pipeline of PWC-ProFlow-D.



**(a)**



**(b)**

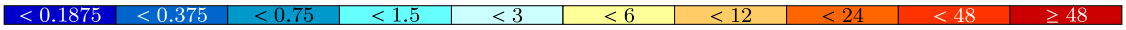
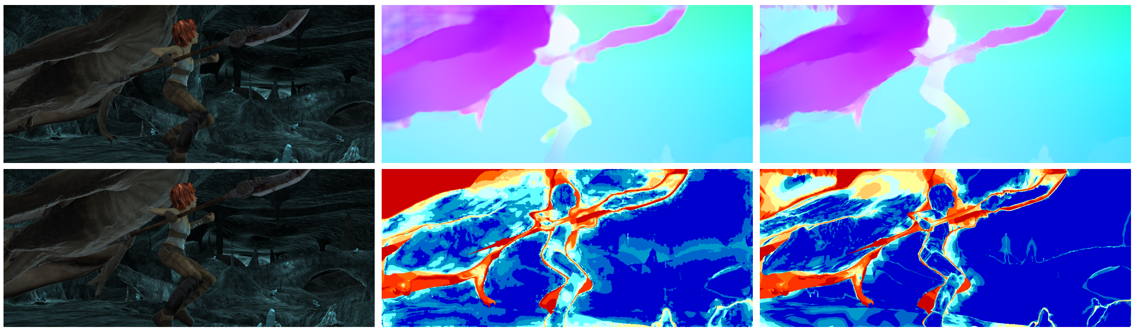


**Figure 5.6:** Visualization of estimation improvements comparing with PWC-ProFlow (Sintel dataset ambush_4 #7 [BWSB12] and KITTI dataset occ #188 [MG15]). **For both figures: First row:** Previous, reference, and next frame. **Second row:** Value map of filtered backward flow, estimated flow of PWC-ProFlow and PWC-ProFlow-D. **Third row:** Value map of filtered forward flow, bad pixel visualization of PWC-ProFlow and PWC-ProFlow-D.

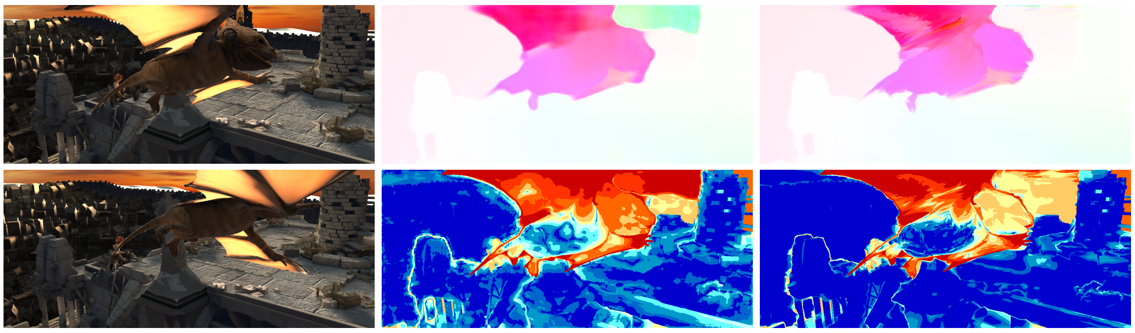
## 5.3 Applying New Filtering Algorithms

### 5.3.1 Model Generation

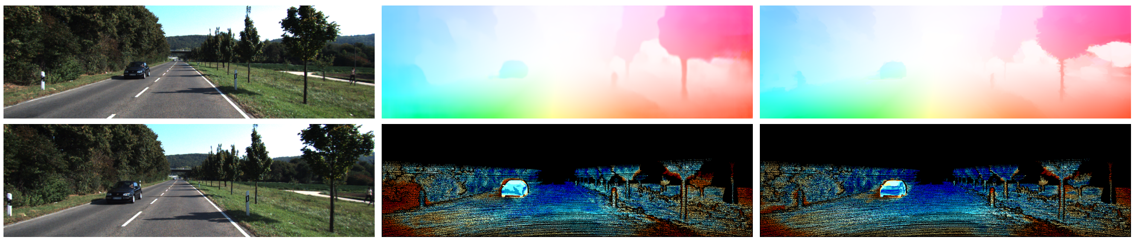
In the first two modifications, we increase the estimation accuracy by improving the inpainting results. However, the root cause of an inaccurate inpainting result and wrong predictions of CNN model is because the filtered forward flow contains few flow vectors in some local regions after the bidirectional filtering. As described in Chapter 4, the unstable performance of the backward flow estimation $(t+1 \longrightarrow t)$ of PWC-Net can be one possible



**(a)**



**(b)**



**(c)**

| < 0.1875 | < 0.375 | < 0.75 | < 1.5 | < 3 | < 6 | < 12 | < 24 | < 48 | ≥ 48 |
|---|---|---|---|---|---|---|---|---|---|

**Figure 5.7:** Visualization of estimation improvements comparing with PWC-Net (Sintel dataset cave_2 19, temple_2 #25 [BWSB12] and KITTI dataset occ #183 [MG15]). **For all three figures: First row:** Reference frame and estimated flow of PWC-Net and PWC-ProFlow-D. **Second row:** The next frame and bad pixel visualization of PWC-Net and PWC-ProFlow-D.

reason that leads to this problem. Hence, another starting point for the modification is to adopt a different filtering method which only takes flow predictions in one direction as input. In this section, we implement two different filtering algorithms. Combined with the normal pipeline steps containing the inpainting step and the new pipeline steps containing the dense filling, the last four new pipelines are created. In the following, we first introduce the two filtering algorithms and then the structures of the new pipelines.

### Uniqueness Filtering

Since occluded pixels are the main source of outliers, we adopt the uniqueness checking, introduced in Section 2.3, to create the first filtering algorithm. The uniqueness filtering takes a flow field as input to generate the filtered flow field. The whole computation process is mainly based on the discrete version of Equation 2.6, where $(i, j)$ denotes a pixel location:

$$o(i,j,t) = \begin{cases} 0 & \text{if } m(i + u^{fw}_{i,j}, j + v^{fw}_{i,j}, t+1) = 1 \\ 1 & \text{else} \end{cases} \tag{5.1}$$

In the case of forward filtering, the input flow refers to the initial forward flow computed in the first step. Further, the initial forward flow vectors are substituted into the equation as the value of $u^{fw}$ and $v^{fw}$. During the implementation, we first build a 2-dimensional array with the same size of the flow field. Each value of the array unit accumulates the total number of pixels that fall into this unit location after the flow vector is added. Normally, bilinear interpolation is applied to deal with the issue that the target location, i.e. the location where a pixel lands after adding the flow vector, deviates from an exact pixel location. In this process, all the four neighboring units around the target location are assigned to a certain value based on the weights computed by the bilinear interpolation. After generating the 2D array, we can check if one pixel is the unique pixel that falls in its target position. In the case that other pixels exist, the flow vector in this pixel location is filtered out. Here, we set the threshold for the uniqueness checking to be 1.5 due to the employing of bilinear interpolation. In the case of backward filtering, the input flow field refers to the initial backward flow.

### Uniqueness and RGB Filtering

Figure 5.8 demonstrates the filtered results of different filtering algorithms. Since our main purpose in the filtering step is to filter the occluded pixels, the ground truth occlusion map is here regarded as the reference result. As one can see, the results of uniqueness filtering contain significantly more flow vectors than the results of the bidirectional filtering. However, compared with the reference occlusion maps, some occluded pixels still remain in the results of uniqueness filtering. To remove more occluded pixels, we create a second

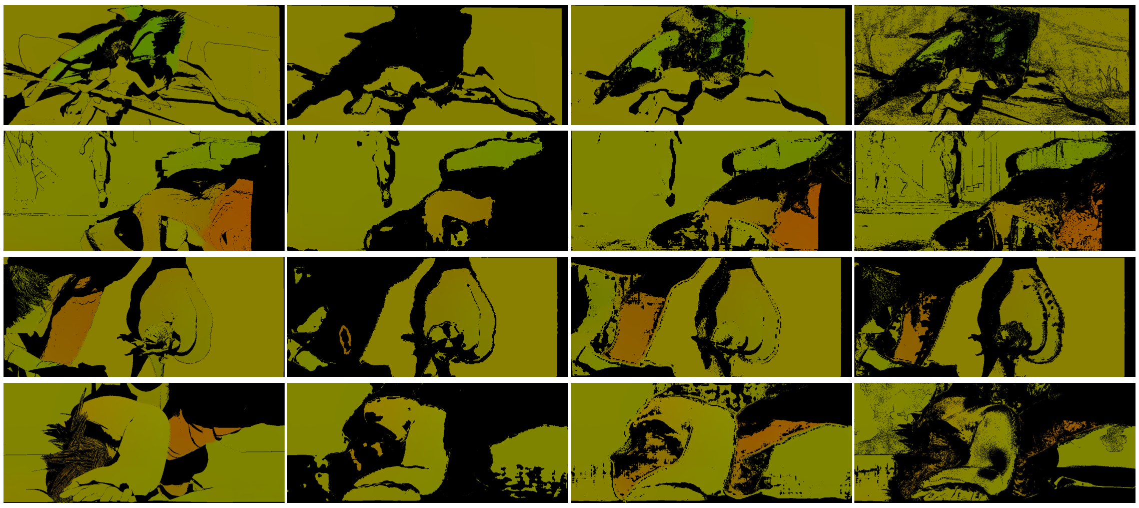


**Figure 5.8:** Forward filtering results using different filtering algorithms. **From top to bottom:** Figures are cave_4 #44, market_5 #18, temple_3 #37, and ambush_4 #7 of the Sintel dataset [BWSB12]. **From left to right:** Ground truth occlusion map, bidirectional filtering, uniqueness filtering, and uniqueness and RGB filtering.

filtering algorithm by combining an extra RGB checking with the uniqueness checking. The uniqueness and RGB filtering takes as inputs the reference image, the target image, and a flow field. The filtering algorithm is implemented as following: first the uniqueness checking is applied to the input flow field. If one flow vector fails in the uniqueness checking, it is filtered out as in the uniqueness filtering. Otherwise, the Euclidean distance of the RGB vectors between the starting pixel in the reference image and the nearest landing pixel in the target image is computed. If the distance is larger than a threshold, the flow vector will be filtered out. For forward filtering, the target image refers to the subsequent frame, and for backward filtering, it is the previous frame.

### Comparing different Filtering Algorithms

As one can see in Figure 5.8, more occluded pixels are filtered out in the results of the uniqueness and RGB filtering, even so, the filtered results still contain more flow vectors than the bidirectional filtering. As conclusion, compared to the reference occlusion maps, the uniqueness filtering would remove slightly fewer occluded pixels in some regions, while the uniqueness and RGB filtering would remove slightly more pixels in some regions. However, in the most cases, both algorithms are able to produce denser filtered results than the bidirectional filtering.

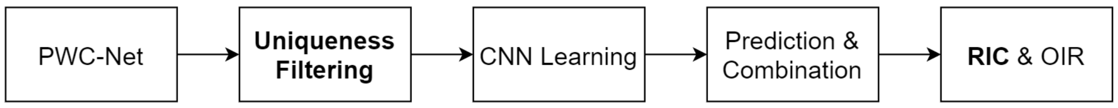
#### Pipeline Generation

After introducing the two filtering algorithms, we now generate four new pipelines based on them. One advantage of using the new filtering algorithms is that it is not necessary to generate the reverse forward $(t+1 \rightarrow t)$ and backward $(t-1 \rightarrow t)$ flow fields in the first step, which can save computational costs. As shown in Figure 5.9, in the first two pipelines, we replace the bidirectional filtering with the uniqueness filtering for both forward and backward filtering. While the first pipeline PWC-ProFlow-U-I applies inpainting as the tool of densification, the second pipeline PWC-ProFlow-U-D uses dense filling for the same purpose. As described above, the filtering results of the uniqueness filtering usually contain some occluded pixels. Hence, in the other two modifications we apply the uniqueness and RGB filtering to the backward filtering and continue to use the uniqueness filtering for the forward filtering. The idea is that the CNN model takes the existing flow vectors in the filtered backward flow $(t \rightarrow t-1)$ as input and performs the forward flow predictions. The backward flow vectors in the occluded regions can lead to wrong predictions, thus an inaccurate final estimation. Hence, in the filtered backward flow, we prefer to leave as less outliers as possible. On the contrary, we slightly increase the tolerance to the number of outliers in the filtered forward flow, since large gaps in the filtered forward flow can bring great difficulties to the subsequent steps. The last two new pipelines are referred to as PWC-ProFlow-U-RGB-I and PWC-ProFlow-U-RGB-D.

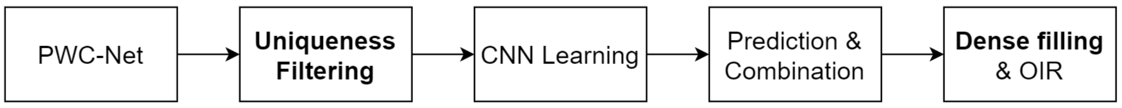
### 5.3.2 Evaluation

All of the four pipelines generated are evaluated on the Sintel and KITTI datasets. The experiment results are demonstrated in Table 5.3.

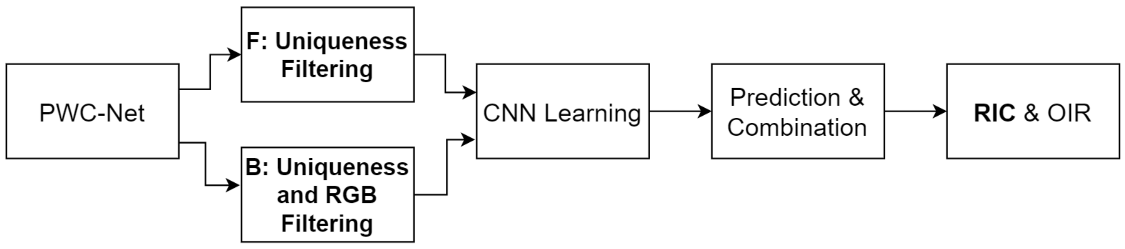
**In comparison with PWC-ProFlow.** As shown in Table 5.3, all the four new pipelines perform better than PWC-ProFlow on both datasets. In particular, the pipelines PWC-ProFlow-U-I and PWC-ProFlow-U-RGB-I outperform PWC-ProFlow with a more than 19% lower AEE on the Sintel dataset, and a more than 10% and 9% lower AEE in the noc and occ cases of the KITTI dataset. This experiment outcome demonstrates the general improvements brought by the new applied filtering algorithms and confirms our idea that allowing more flow vectors to retain in the filtered flow can increase the estimation accuracy. Figure 5.10 (a) and (b) show the results of the four modifications regarding the negative estimation examples of PWC-ProFlow mentioned in Chapter 4. In subfigure (a), we can detect great improvements in the leg area in the estimation results. Inspecting both the filtered backward flow and the filtered forward flow of uniqueness filtering in Figure 5.8, we can observe that some locations in the leg area possess both forward and backward flow after filtering, which is not the case in PWC-ProFlow. The information in these locations can be added to the training dataset, thus, the CNN model has a chance to learn parts of the motion pattern in the leg area. This helps the CNN model to predict the forward flow in this area more accurately.



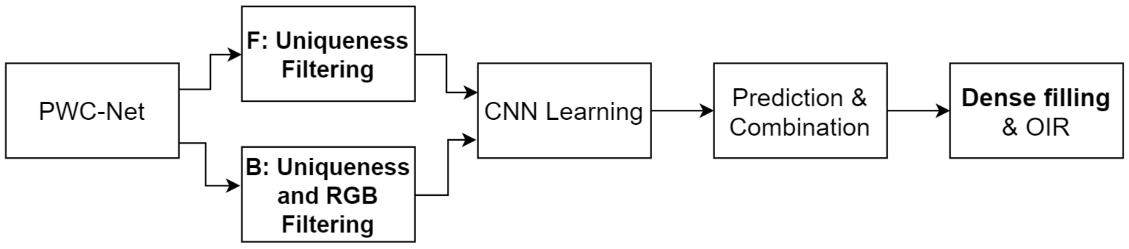
**(a)**



**(b)**



**(c)**



**(d)**

**Figure 5.9: From (a) to (d):** Overview of the pipeline of PWC-ProFlow-U-I, PWC-ProFlow-U-D, PWC-ProFlow-U-RGB-I, and PWC-ProFlow-U-RGB-D.

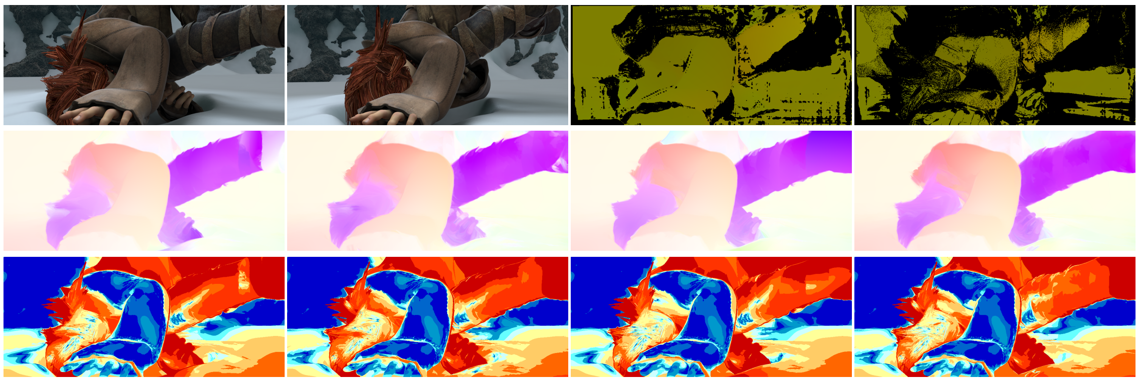
**In comparision with each other.** By comparing the four modifications with each other, we discover that the pipelines with dense filling generally outperform the pipelines with inpainting, as previously detected in Section 5.2. Between different filtering algorithms, the cooperation of uniqueness filtering and uniqueness and RGB filtering achieves better accuracy on both datasets than only employing uniqueness filtering for both forward and backward filtering. It indicates that less outliers in the filtered backward flow can help to reduce the amount of wrong predictions. Actually, we have also built two pipelines, where the uniqueness and RGB filtering is applied for both forward and backward filtering. However, the experiment results do not show an improvement compared to the pipelines with above two filtering combinations. As a conclusion, the pipeline PWC-ProFlow-U-RGB-D performs the best among the four new pipelines created in this section.

**In comparison with the PWC-Net.** As shown in Table 5.3, all the four variations outperform PWC-Net in the estimation of Sintel dataset, especially the method PWC-ProFlow-U-RGB-D, which achieves the best accuracy on Sintel dataset among all the experiment

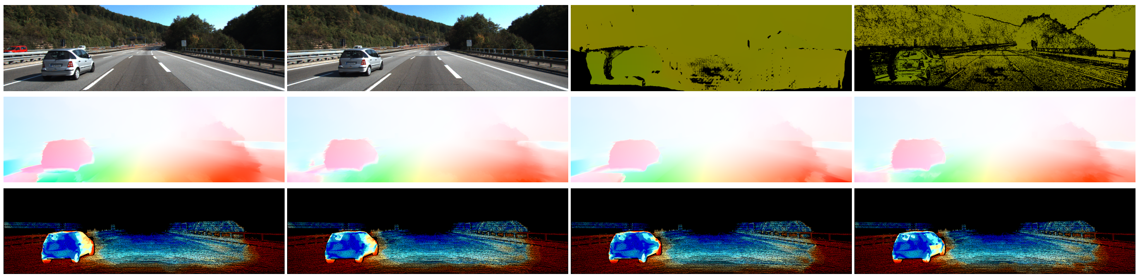
**Table 5.3:** Average endpoint error (AEE) and the percentage of bad pixels (BP) of different methods on the training set of KITTI 2015 and MPI Sintel (clean path). One pixel with an endpoint error > 3px is defined as a bad pixel.

| **Method** | **KITTI 2015** | | | | **Sintel** |
|---|---|---|---|---|---|
| | *noc* | | *occ* | | clean |
| | AEE | BP (%) | AEE | BP (%) | AEE |
| ProFlow | 3.16 | 11.40 | 5.56 | 17.83 | 1.98 |
| PWC-Net | 1.89 | 8.52 | 3.25 | 13.68 | 1.95 |
| PWC-ProFlow | 2.89 | 11.52 | 5.18 | 19.03 | 2.34 |
| PWC-ProFlow-S | 2.16 | 9.10 | 3.65 | 15.92 | 1.82 |
| PWC-ProFlow-D | 1.83 | 8.51 | 3.23 | 14.38 | 1.82 |
| PWC-ProFlow-U-I | 2.60 | 9.56 | 4.98 | 17.37 | 1.95 |
| PWC-ProFlow-U-D | 2.08 | 9.03 | 4.06 | 15.89 | 1.82 |
| PWC-ProFlow-U-RGB-I | 2.61 | 9.61 | 4.62 | 16.84 | 1.88 |
| PWC-ProFlow-U-RGB-D | 1.95 | 8.70 | 3.56 | 14.82 | 1.75 |

candidates. However, for the KITTI dataset, PWC-Net possesses a higher estimation accuracy in both noc and occ cases. We can also see an outperformance when comparing the pipeline PWC-ProFlow-D with PWC-ProFlow-U-D and PWC-ProFlow-U-RGB-D. Since the general effectiveness of the new filtering algorithms is proved in the comparison with PWC-ProFlow, this outcome indicates that in some frames of the KITTI dataset the new filtering algorithms perform significantly worse than the bidirectional filtering. After analyzing the experiment results, we discover some negative samples as shown in Figure 5.11. Subfigure (a) and (b) present two examples in the noc case. It is clearly to see that the filtered forward flow of bidirectional filtering contain more flow vectors. Due to the lack of training samples, the front regions of the cars in subfigure (a) and subfigure (b) are predicted with a very low accuracy in the pipelines PWC-ProFlow-U-D and PWC-ProFlow-U-RGB-D. In this context, one possible reason is that the inaccurate initial flow vectors computed by the PWC-Net can cause multiple pixels from one object landing in the same area in the target image, for example the car area in subfigure (b). After bilinear interpolation, some valid positions in this area may contain a value higher than the threshold, thus are filtered out. For this problem, we can consider to slightly increase the threshold defined for the uniqueness checking for the KITTI dataset to leave more flow vectors in the filtered forward flow. Besides the problem observed in the noc case, we also find that in the occ case, the predictions in the bottom edge area in some frames are slightly deteriorated by the neighbouring flow, as shown in the subfigure (c). These wrong predictions can also decrease the overall estimation accuracy.



(a)



(b)

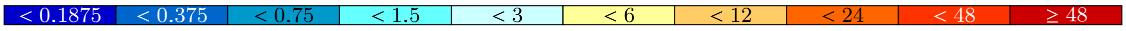
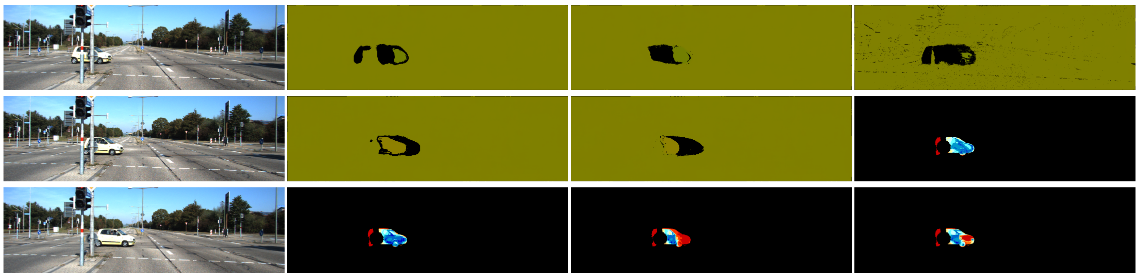
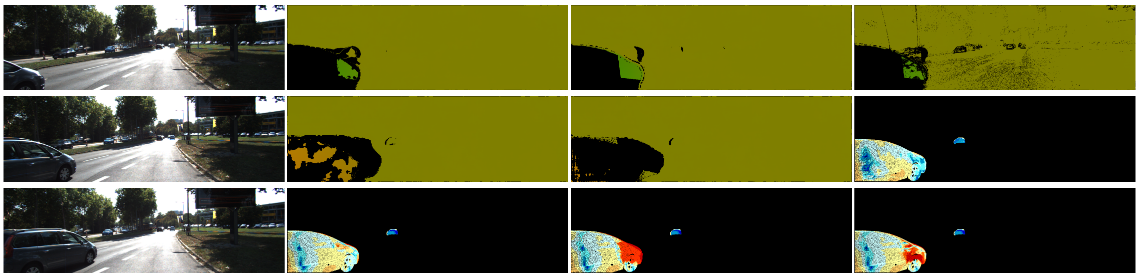


**Figure 5.10:** Visualization of estimation improvements comparing with PWC-ProFlow (Sintel dataset ambush_4 #7 [BWSB12] and KITTI dataset occ #188 [MG15]). **For both figures: First row:** Reference frame, subsequent frame, value map of filtered backward flow using uniqueness filtering and uniqueness and RGB filtering. **Second row and third row:** Estimation results and bad pixel visualization of four modifications. **From left to right:** PWC-ProFlow-U-I, PWC-ProFlow-U-D, PWC-ProFlow-U-RGB-I, and PWC-ProFlow-U-RGB-D.

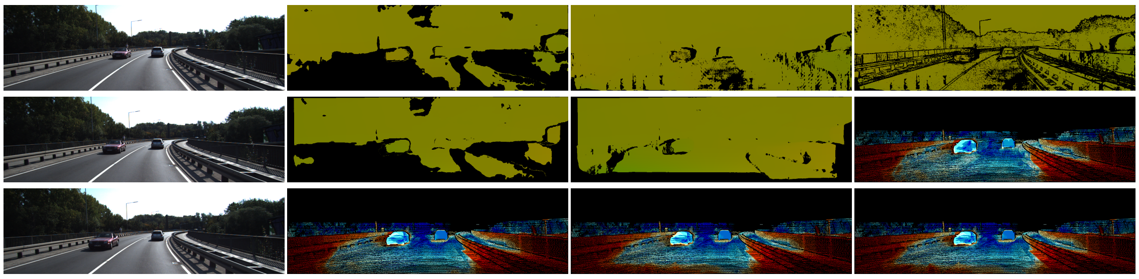
Summarizing all the modifications created in this chapter, the experiment results in Table 5.3 demonstrate that all the modifications perform better than the original PWC-ProFlow created in Chapter 4 on both datasets. In particular, the pipeline PWC-ProFlow-U-RGB-D achieves the best estimation performance on the Sintel dataset, while PWC-ProFlow-D yields the best estimation performance in the noc and occ cases of the KITTI dataset.



**(a)**



**(b)**



**(c)**

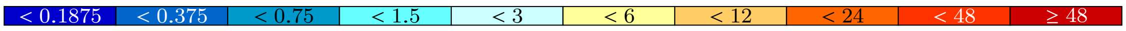


**Figure 5.11:** Comparison of estimation results in KITTI dataset (noc #43, noc #91, and occ #176 [MG15]). **For both figures: First row:** Previous frame, value map of filtered backward flow using bidirectional filtering, uniqueness filtering, and uniqueness and RGB filtering. **Second row:** Reference frame, value map of filtered forward flow using bidirectional filtering and uniqueness filtering, bad pixel visualization of PWC-Net. **Third row:** Next frame, bad pixel visualization of PWC-ProFlow-D, PWC-ProFlow-U-D, and PWC-ProFlow-U-RGB-D.

# 6 Conclusion

In this thesis, we combined two top-performing optical flow algorithms PWC-Net and ProFlow with the purpose of generating a new algorithm that can outperform these two algorithms and combine their advantages. Through the evaluation on the MPI Sintel and KITTI 2015 benchmarks, we discovered that the intuitive combination of both algorithms did not achieve the expected performance on either dataset. To improve the estimation performance, we explored the causes that lead to the underperformance of the new algorithm. We found that the sparse filtered forward flow produced by the bidirectional filtering can on the one hand lead to large gaps in the input flow of the inpainting step, on the other hand lead to wrong predictions of the CNN model. Both problems can severely reduce the estimation accuracy. Based on these findings, we proposed six different modifications in this thesis. In the first modification, the large gaps contained in the input flow of the inpainting step are partially filled via sampling so that more information can be provided for the inpainting process. In the second modification, the inpainting step is completely replaced with a so-called dense filling method which densifies the flow field by directly sampling the initial forward estimation of PWC-Net. Thanks to the generally good estimation of PWC-Net, it turns out that the dense filling achieves a more stable performance compared to the inpainting. In the last four modifications, new filtering algorithms are embedded in the pipeline. By applying new filtering algorithms, we expected to leave more flow vectors in the filtered forward flow. Through experiments, it confirms that with proper setting of the threshold, the new filtering algorithms can produce denser filtered flow than the bidirectional filtering. By analyzing the experiment results, we discussed advantages and disadvantages of each modification with positive and negative samples. As a conclusion, compared to PWC-ProFlow and ProFlow, all the modifications created demonstrate significant improvements in the estimation accuracy on both MPI Sintel and KITTI 2015 datasets. Compared to PWC-Net, our modifications also present great improvements on the Sintel dataset. In particular, the pipelines PWC-ProFlow-D and PWC-ProFlow-U-RGB-D achieve the best estimation accuracy among all the experiment candidates on the KITTI and Sintel dataset, respectively.

In the future, we can further modify the pipelines generated in this thesis. For example, the threshold in the uniqueness filtering can be fine-tuned for the KITTI dataset to see if the pipeline PWC-ProFlow-U-RGB-D can outperform the PWC-Net on the KITTI dataset. Since the RGB checking may not be fully appropriate to lightning changes, we can further integrate other filtering algorithms in the pipeline. Beyond that, it is also worth exploring how important the last refinement step is by comparing the estimation performance with and without refinement. Moreover, we can also fine-tune the parameters contained in the variational refinement to see if some further improvements can be brought to the estimation results.

# Bibliography

[BBPW04] T. Brox, A. Bruhn, N. Papenberg, J. Weickert. “High accuracy optical flow estimation based on a theory for warping.” In: *Proceedings European Conference on Computer Vision*. 2004, pp. 25–36 (cit. on p. 13).

[BLKU16] M. Bai, W. Luo, K. Kundu, R. Urtasun. “Exploiting semantic information and deep matching for optical flow.” In: *Proceedings European Conference on Computer Vision*. 2016, pp. 154–170 (cit. on p. 9).

[Bru18] A. Bruhn. *Lecture notes in Correspondence Problems in Computer Vision*. 2018 (cit. on pp. 8, 10, 12–14, 16).

[BSL11] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, R. Szeliski. “A database and evaluation methodology for optical flow.” In: *International Journal of Computer Vision* 92.1 (2011), pp. 1–31 (cit. on p. 16).

[BVS17] C. Bailer, K. Varanasi, D. Stricker. “CNN-Based Patch Matching for Optical Flow with Thresholded Hinge Embedding Loss.” In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition* (2017) (cit. on p. 9).

[BWSB12] D. J. Butler, J. Wulff, G. B. Stanley, M. J. Black. “A naturalistic open source movie for optical flow evaluation.” In: *Proceedings European Conference on Computer Vision*. 2012, pp. 611–625 (cit. on pp. 10, 15, 33, 37–39, 44, 45, 47, 48, 50, 54, 62, 63).

[CGN14] H. Chao, Y. Gu, M. Napolitano. “A Survey of Optical Flow Techniques for Robotics Navigation Applications.” In: *Journal of Intelligent & Robotic Systems* 73.1 (2014), pp. 361–372 (cit. on p. 7).

[DFI15] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, T. Brox. “Flownet: Learning optical flow with convolutional networks.” In: *Proceedings IEEE International Conference on Computer Vision*. 2015, pp. 2758–2766 (cit. on p. 9).

[FBK15] D. Fortun, P. Bouthemy, C. Kervrann. “Optical flow modeling and computation: a survey.” In: *Computer Vision and Image Understanding* 134.1 (2015), pp. 21–46 (cit. on pp. 7, 8, 12–14, 16, 18, 19).

[GW16] D. Gadot, L. Wolf. “PatchBatch: A batch augmented loss for optical flow.” In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4236–4245 (cit. on p. 9).

[HLS17] Y. Hu, Y. Li, R. Song. “Robust interpolation of correspondences for large displacement optical flow.” In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 481–489 (cit. on pp. 10, 28).

[HLVW17] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger. “Densely connected convolutional networks.” In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4700–4708 (cit. on p. 24).

[HS81] B. K. P. Horn, B. G. Schunck. “Determining Optical Flow.” In: *Artificial Intelligence.* 17 (1981), pp. 185–203 (cit. on pp. 8, 13).

[HSL16] Y. Hu, R. Song, Y. Li. “Efficient coarse-to-fine patchmatch for large displacement optical flow.” In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 5704–5712 (cit. on pp. 27–29).

[HZZ17] Y. Han, P. Zhang, T. Zhuo, W. Huang, Y. Zhang. “Video Action Recognition Based on Deeper Convolution Networks with Pair-Wise Frame Motion Concatenation.” In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017, pp. 1226–1235 (cit. on p. 7).

[IMS17] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, T. Brox. “Flownet 2.0: Evolution of optical flow estimation with deep networks.” In: *Proceedings IEEE Conference on Computer Cision and Pattern Recognition*. 2017, pp. 2462–2470 (cit. on pp. 9, 20).

[LDY18] X. Liu, Z. Deng, Y. Yang. “Recent progress in semantic image segmentation.” In: *Artificial Intelligence Review* (2018), pp. 1–18 (cit. on p. 8).

[LHS18] Y. Li, Y. Hu, R. Song, P. Rao, Y. Wang. “Coarse-to-fine PatchMatch for dense correspondence.” In: *IEEE Transactions on Circuits and Systems for Video Technology* 28.9 (2018), pp. 2233–2245 (cit. on p. 10).

[LK81] B. D. Lucas, T. Kanade. “An Iterative Image Registration Technique with an Application to Stereo Vision.” In: *Proceedings International Joint Conference on Artificial Intelligence*. 1981, pp. 674–679 (cit. on p. 13).

[Low04] D. G. Lowe. “Distinctive image features from scale-invariant keypoints.” In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110 (cit. on p. 9).

[LUT18] C. Lu, H. Uchiyama, D. Thomas, A. Shimada, R.-i. Taniguchi. “Sparse Cost Volume for Efficient Stereo Matching.” In: *Remote Sensing* 10.11 (2018) (cit. on p. 23).

[MB18] D. Maurer, A. Bruhn. “ProFlow: Learning to Predict Optical Flow.” In: *Proceedings British Machine Vision Conference*. 2018 (cit. on pp. 27, 28, 30).

[MG15] M. Menze, A. Geiger. “Object Scene Flow for Autonomous Vehicles.” In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3061–3070 (cit. on pp. 10, 33, 38–40, 44, 45, 47, 48, 54, 55).

[MHG15] M. Menze, C. Heipke, A. Geiger. "Discrete optimization for optical flow." In: *Proceedings German Conference on Pattern Recognition*. 2015, pp. 16–28 (cit. on p. 10).

[MSB17] D. Maurer, M. Stoll, A. Bruhn. "Order-adaptive and illumination-aware variational optical flow refinement." In: *Proceedings British Machine Vision Conference*. 2017, pp. 9–26 (cit. on pp. 9, 10, 28).

[ON94] M. Otte, H.-H. Nagel. "Optical flow estimation: advances and comparisons." In: *Proceedings European Conference on Computer Vision*. 1994, pp. 49–60 (cit. on p. 17).

[Onk13] N. Onkarappa. "Optical flow in Driver Assistance Systems." PhD thesis. Computer Vision Center, University Autònoma de Barcelona, 2013 (cit. on p. 7).

[RB17] A. Ranjan, M. J. Black. "Optical flow estimation using a spatial pyramid network." In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4161–4170 (cit. on pp. 9, 20).

[RWHS15] J. Revaud, P. Weinzaepfel, Z. Harchaoui, C. Schmid. "Epicflow: Edge-preserving interpolation of correspondences for optical flow." In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1164–1172 (cit. on p. 10).

[SB12] N. Sharmin, R. Brad. "Optimal filter estimation for Lucas-Kanade optical flow." In: *Sensors* 12.9 (2012), pp. 12694–12709 (cit. on p. 17).

[SRLB08] D. Sun, S. Roth, J. Lewis, M. J. Black. "Learning optical flow." In: *Proceedings European Conference on Computer Vision*. 2008, pp. 83–97 (cit. on p. 9).

[SSJB16] L. Sevilla-Lara, D. Sun, V. Jampani, M. J. Black. "Optical Flow with Semantic Segmentation and Localized Layers." In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 3889–3898 (cit. on p. 9).

[SYLK18] D. Sun, X. Yang, M.-Y. Liu, J. Kautz. "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume." In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8934–8943 (cit. on pp. 9, 20–22, 24–26).

[VP89] A. Verri, T. Poggio. "Motion field and optical flow: qualitative properties." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11.5 (1989), pp. 490–498 (cit. on p. 7).

[WC11] A. Wedel, D. Cremers. *Stereo scene flow for 3D motion analysis*. Springer Science & Business Media, 2011 (cit. on p. 8).

[WRHS13] P. Weinzaepfel, J. Revaud, Z. Harchaoui, C. Schmid. “DeepFlow: Large displacement optical flow with deep matching.” In: *Proceedings IEEE International Conference on Computer Vision*. 2013, pp. 1385–1392 (cit. on p. 10).

[WSB17] J. Wulff, L. Sevilla-Lara, M. J. Black. “Optical flow in mostly rigid scenes.” In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4671–4680 (cit. on p. 9).

[XJM12] L. Xu, J. Jia, Y. Matsushita. “Motion detail preserving optical flow estimation.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.9 (2012), pp. 1744–1757 (cit. on p. 15).

[XRK17] J. Xu, R. Ranftl, V. Koltun. “Accurate Optical Flow via Direct Cost Volume Processing.” In: *IEEE Conference on Computer Vision and Pattern Recognition* (2017) (cit. on pp. 9, 23).

[YK16] F. Yu, V. Koltun. “Multi-Scale Context Aggregation by Dilated Convolutions.” In: *Computing Research Repository* (2016) (cit. on p. 25).

[ZZXW19] Z.-Q. Zhao, P. Zheng, S.-T. Xu, X. Wu. “Object Detection With Deep Learning: A Review.” In: *IEEE Transactions on Neural Networks and Learning Systems* (2019), pp. 1–21 (cit. on p. 8).

# Appendix A Experiment Results per Scene of Sintel Benchmark

**Table A.1:** Average Endpoint Error (AEE) of different experiment methods on each scene of the Sintel dataset [BWSB12].

| Sintel | PWC-Net | ProFlow | PWC-ProFlow | PWC-ProFlow-S | PWC-ProFlow-D |
|---|---|---|---|---|---|
| Alley_1 | 0.26290 | 0.15685 | 0.14982 | 0.15102 | 0.15316 |
| Alley_2 | 0.26994 | 0.17888 | 0.20523 | 0.20618 | 0.18818 |
| Ambush_2 | 4.68224 | 5.81265 | 5.83254 | 4.79243 | 5.57515 |
| Ambush_4 | 7.68907 | 7.84383 | 11.30393 | 7.60571 | 7.82731 |
| Ambush_5 | 2.95132 | 3.57785 | 3.52813 | 2.59937 | 2.74928 |
| Ambush_6 | 5.19208 | 5.46332 | 6.42998 | 4.76556 | 4.79748 |
| Ambush_7 | 0.39274 | 0.48187 | 0.32670 | 0.33738 | 0.29575 |
| Bamboo_1 | 0.50773 | 0.29201 | 0.33607 | 0.33862 | 0.32860 |
| Bamboo_2 | 1.34362 | 0.99685 | 1.48103 | 1.38246 | 1.28434 |
| Bandage_1 | 0.68759 | 0.45553 | 0.47506 | 0.47154 | 0.47190 |
| Bandage_2 | 0.42328 | 0.22578 | 0.23779 | 0.23786 | 0.23814 |
| Cave_2 | 5.54194 | 5.24823 | 6.74364 | 5.37731 | 5.14732 |
| Cave_4 | 3.26355 | 3.12966 | 4.69484 | 3.26329 | 3.27692 |
| Market_2 | 0.59580 | 0.50632 | 0.46170 | 0.45825 | 0.44410 |
| Market_5 | 6.74354 | 7.58080 | 8.84742 | 6.83291 | 6.84383 |
| Market_6 | 1.80960 | 1.68491 | 1.74238 | 1.59443 | 1.53395 |
| Mountain_1 | 0.22047 | 0.18466 | 0.20293 | 0.20054 | 0.16302 |
| Shaman_2 | 0.28201 | 0.16209 | 0.16266 | 0.16264 | 0.16283 |
| Shaman_3 | 0.19419 | 0.13486 | 0.13162 | 0.13178 | 0.13160 |
| Sleeping_1 | 0.11692 | 0.07014 | 0.07018 | 0.07018 | 0.07028 |
| Sleeping_2 | 0.12577 | 0.04792 | 0.04795 | 0.04795 | 0.04799 |
| Temple_2 | 1.74801 | 2.15232 | 1.84612 | 1.67112 | 1.55613 |
| Temple_3 | 3.75267 | 4.39615 | 4.87433 | 3.74509 | 3.66427 |
| **AEE** | 1.95478 | 1.97869 | 2.34281 | 1.82399 | 1.82152 |

**Table A.2:** Average Endpoint Error (AEE) of different experiment methods on each scene of the Sintel dataset [BWSB12].

| **Sintel** | **PWC-ProFlow-U-I** | **PWC-ProFlow-U-D** | **PWC-ProFlow-U-RGB-I** | **PWC-ProFlow-U-RGB-D** |
|---|---|---|---|---|
| Alley_1 | 0.15596 | 0.15078 | 0.15394 | 0.15081 |
| Alley_2 | 0.19845 | 0.19086 | 0.19882 | 0.18687 |
| Ambush_2 | 5.60160 | 4.85103 | 3.52503 | 4.30483 |
| Ambush_4 | 8.31790 | 7.87171 | 7.76622 | 7.56828 |
| Ambush_5 | 3.31510 | 3.02773 | 3.11496 | 2.73931 |
| Ambush_6 | 4.65903 | 4.52097 | 4.52820 | 4.47340 |
| Ambush_7 | 0.41854 | 0.34818 | 0.43254 | 0.33591 |
| Bamboo_1 | 0.32569 | 0.32480 | 0.32498 | 0.32332 |
| Bamboo_2 | 1.26679 | 1.28383 | 1.27124 | 1.21105 |
| Bandage_1 | 0.46976 | 0.47636 | 0.47203 | 0.47572 |
| Bandage_2 | 0.23799 | 0.24039 | 0.23761 | 0.23943 |
| Cave_2 | 5.66957 | 5.39526 | 5.61940 | 5.23465 |
| Cave_4 | 3.21453 | 3.07781 | 3.12479 | 2.99103 |
| Market_2 | 0.49876 | 0.46415 | 0.50892 | 0.46084 |
| Market_5 | 7.38419 | 6.70790 | 7.57562 | 6.55440 |
| Market_6 | 1.79156 | 1.63367 | 1.74575 | 1.54246 |
| Mountain_1 | 0.16060 | 0.14889 | 0.15855 | 0.14883 |
| Shaman_2 | 0.16252 | 0.16275 | 0.16257 | 0.16276 |
| Shaman_3 | 0.13103 | 0.13177 | 0.13089 | 0.13149 |
| Sleeping_1 | 0.07018 | 0.07028 | 0.07018 | 0.07028 |
| Sleeping_2 | 0.04796 | 0.04798 | 0.04796 | 0.04799 |
| Temple_2 | 1.72068 | 1.56444 | 1.78683 | 1.51854 |
| Temple_3 | 3.91506 | 3.75033 | 3.94485 | 3.67557 |
| **AEE** | 1.94905 | 1.82464 | 1.88437 | 1.75424 |

## Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature