ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

BIWI Computer Vision

Self-supervised Inference of Object Movement from Video

Master's Thesis

Qi Dai
Department of Civil, Environmental and Geomatic Engineering

Advisor: Vaishakh Patil
Simon Hecker
Dengxin Dai
Supervisor: Konrad Schindler
Prof. Dr. Luc van Gool

August 10, 2019

Abstract

Many moving objects are presented when driving in the city. The knowledge of their movement, e.g. moving vector in the 3D space, provides valuable information for making the next driving decision. In this thesis, we try to infer the object movement given a video sequence, captured by a moving camera. A pipeline based on view synthesis is proposed to solve this problem in a self-supervised manner. Two networks are employed to predict the pixel-wise depth and 3D scene flow map for the input image. The evaluation on KITTI dataset demonstrates the effectiveness of our approach to estimate the 3D object movement.

Acknowledgements

I really appreciate **Simon Hecker** and **Vaishakh Patil** a lot for the instruction and help throughout the whole project. I would like to thank **Prof. Dr. Konrad Schindler**, **Prof. Dr. Luc van Gool** and **Dr. Dai Dengxin** a lot for offering me the opportunity of working on this project. I also want to express my gratitude for the support from the Computer Vision Laboratory, ETH Zurich.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A good understanding of the surrounding environment is indispensable for the driving safety. Considering a common road scene in Figure 1.1, basically we can divide this image into two areas. One is the background structure, like the buildings or the trees. These areas are rigid, which provide information for people to locate and navigate themselves during driving. The other is area which contains moving objects. These dynamic objects, like cars or pedestrains, are supposed to be treated carefully for safety or efficiency consideration.

Figure 1.1: Common driving scene

So it is very important to distinguish the dynamic objects and infer their movement(speed and moving direction). In the driving context, the drivers can determine their next driving action based on the movement of the surrounding objects. For example, we need to slow down when some cars try to change the lane and driving in front of us. Or we can accelerate when the front car moved into another lane. In SLAM(*Simultaneous localization and mapping*), the mapping result can also be further refined by the elimination of the moving object, if we are able to distinguish them from the scene.

The detection of moving objects from video has been researched for a long time. Normally the camera used to capture the video is static. In this case, the dynamic area can be determined by simply comparing two consecutive frames, and finding the area where the difference of pixel intensity is big. The intuition behind is that, the location of moving objects on the image plane between two subsequent frames is changing, while those static objects/background won't. So in static setup, areas with significant photometric difference are potentially moving objects.

However, the dynamic object detection becomes more difficult in the driving context. Suppose now the recording camera is moving, then the scene motion resulted from camera motion is entangled with the dynamic object movement. For one hand, the rigid structure is moving in the scene of two consecutive frame, due to the camera motion. On the other hand, those dynamic object is moving independently at the same time. In order to detect dynamic objects in the driving context,

we must disentangle the independent object movement from the camera motion.

One solution to address this motion confusion is to utilize the geometry of the rigid structure. The scene geometry is fully described by the dense depth map. Provided with the depth maps of two consecutive frames, their corresponding 3D models of the covered scene can be reconstructed respectively. Then these two models can be further aligned with each other based on the camera transformation. After the model alignment, those rigid objects/background is overlapped. While there will be a discrepancy between the models of moving objects. This discrepancy encapsulates the object movement, and is of vital importance for the dynamic object detection.

To solve this discrepancy, the geometry of the surrounding structure, in particular the depth map is needed. Depth estimation from images is a intensively-researched topic in computer vision. Recently Zhou et al.[32] have proposed to solve this problem in self-supervised manner based on the idea of view synthesis, without using any depth ground truth. Inspired by his work, together with the proposal from Cao[4], we try to solve the dynamic object movement in a self-supervised manner.

1.1 Focus of this Work

In this work, we focus on inferring the movement of dynamic object in the driving context. More specifically, we want to determine the object movement, given a video or image sequence taken by a moving camera.

The object movement can be numerically described as a 3D scene flow vector. Suppose now we have two consecutive frames I_t and I_{t+1}, captured by a moving camera at time *t* and *t+1* respectively. Our purpose is to estimate a pixel-wise 3D scene-flow vector map, aligned with I_t. The scene-flow map describes the moving vector between *t* to *t+1* of the point in 3D space. This point is back-projected from the pixel on I_t image plane, and the scene-flow vector is within the camera coordinate system at time *t*.

As explained before, a good depth map is the prerequisite of object movement estimation. In this work we also explore the prediction of depth map from image. We try to solve these tasks in self-supervised manner, without using any ground truth of depth or scene-flow.

1.2 Thesis Organization

This report is organized as following:

In Chapter 2, an introduction of related work is given. We mainly cover the work of supervised and unsupervised depth estimation from image, as well as solutions to compensate or explain the object movement in the scene.

In Chapter 3 we detail the theory of *View Synthesis*. The depth and scene-flow estimation can be learned in a self-supervised manner with supervised signal provided by the synthesized view. We introduce the view synthesis under the rigid-scene assumption and for scenes with moving objects.

In Chapter 4 we give a full description of our methodology. The details of challenges and its solutions are provided. We will introduce the network architecture and the formulation of loss

function.

In Chapter 5 the experiments are described and the results are presented. We mainly evaluate the depth prediction, the End-point error of the predicted 2D optical flow, and the instance-level scene-flow vector estimation.

The discussion and the conclusion are given in Chapter 6 and 7 respectively. More insights toward our result are given. The limitation as well as the future work are also presented in this part.

Chapter 2

Related Work

The inference of object movement from video is based on a good understanding of the scene geometry. The geometry is described by a dense depth map of the scene. In this part we first review works related to monocular depth estimation, where a pixel-wise depth map is estimated given a single input color image. However, some depth estimation methods, in particular those take the *monocular video* as input, assume the covered scene are rigid. Since this assumption is generally not true in the real scenario, some methods are proposed to compensate for the moving object. We also review works related to this topic.

2.1 Monocular Depth Estimation

The monocular depth estimation can either be solved in supervised or self-supervised manner.

Supervised monocular depth estimation The depth estimation is formulated as a regression problem in most supervised approaches. In this setup the difference between the predicted depth and its ground truth is minimized. The manually defined feature is used in early work. Saxena *et al.*[27] propose to estimate the single-view depth by training Markov random field(MRF) with hand-crafted features. In [20] Liu *et al.* integrate semantic labels with MRF learning. Ladicky *et al.* improve the depth estimation performance by combining the semantic labeling with the depth estimation.

The deep neural network is powerful tool and inspires many other methods. Eigen *et al.*[8] propose a deep convolutional neural network(CNN) architecture to produce dense depth map. Based on this architecture, many variant structures have been proposed to improve the prediction performance. Li *et al.*[19] improve the estimation accuracy by combining the CNNs with the conditional random filed(CRF), while Laina *et al.*[18] use the more robust *Huber loss* as the loss function. Cao *et al.*[3] formulate the depth prediction problem as a pixel-wise classification task. Besides CNNs, Kumar *et al.*[17] propose a model with recurrent neural network(RNN) to provide spatio-temporally accurate monocular depth estimation.

However, supervised approach requires large amounts of of accurate and pixel aligned depth ground truth. And ideally various scenes should be covered in the training dataset. This type of data is difficult to obtain in a large scale in the real world. The lack of ground truth depth map may hinder the performance of supervised approach. One possible solution is to use syntheic data with perfect depth ground truth. In recent work [2], synthetic images with highly accurate depth ground truth are use to train a depth estimation network, and an image style transfer network is trained to convert a real image into the synthetic domain, then the depth map can be estimated

from real image.

Another alternative is to utilize those weakly supervised training data. For example, Wu *et al.*[29] proposes to predict a coarse depth using sparse label for real-world size of object, and then refine the depth map based on CPF. Zoran *et al.*[33] introduce sparse ordinal depth as supervision, while Kundu *et al.*[25] use unpaired synthetic depth data. All these approaches requires the collection of extra ground truth annotation.

Unsupervised monocular depth estimation The photometric consistency between nearby frames makes it possible to predict the depth without the ground truth. Here a set of images are given as input. The network is trained to minimize the image reconstruction error, where the image is synthesized based on the depth prediction.

One category is to learn depth from synchronized stereo image pairs. In this setup the pose between the stereo cameras is already known. Grag *et al.*[9] trains a network to predict the depth that minimize the photometric difference between the true right view and the synthesized right view. They use Taylor expansion to approximate the cost function and derive the gradient. As a result this approximated objective is only sub-optimal. Godard *et al.*[12] choose the spatial transformer network, yielding a cost function which is differentiable without any approximation. They reconstruct the image using the predicted disparity, and enforce the left-right disparity consistency to encourage a more accurate prediction.

Another category is to infer from single consecutive temporal image sequence. Zhou *et al.* [32] and Vijayanarasimhan [28] show that the learning of depth prediction and ego-motion at the same time is possible. The depth network and pose network are jointly trained, and the supervision signal is provided by minimizing the photometric difference between the synthesized view and true view. The ego-motion estimation from pose network makes it possible to train without the stereo image pairs. In recent works, several other constraints are introduced during the training. Yang *et al.* [30] utilize the consistency between normal and depth. Mahjourian *et al.* [22] propose a 3D point cloud alignment loss.

2.2 Compensation for non-rigid scene motion

One significant limitation of the *view-synthesis* based approach is the rigid scene assumption. In stereo setup we don't have this problem, since the stereo paired images have been synchronized with each other. For approaches which taken monocular video as input, different methods have been proposed to compensate for the non-rigid scene motion.

Besides the commonly used depth and pose network, Yin *et al.* [31] proposed to include another network to explain the object movement. This complemented network predicts a residual 2D optical flow resulted the object motion. Then image is warped by combining the rigid flow from the camera motion, and the residual flow from non-rigid scene movement. However, they reports that their proposal can only rectify small errors from rigid flow while not able to compensate big flow error, due to the gradient locality of warping loss.

Introducing more information regarding the moving objects may potentially to improve the result. Casser *et al.* [6] use pre-computed instance segmentation masks to highlight objects in the scene, and predict a 6 DoF transformation to explicitly explain the movement of each object. Instead of using segmentation mask, Cao *et al.* [5] extract objects by using a 2D bounding box.

In their work they also explicitly model the object motion by predicting the dense 3D scene flow map for each detected bounding box, which can explain both the camera motion and the object movement.

Chapter 3

Theoretical Background

We try to learn the object movement in a self-supervised manner. Without the need of ground truth, the primary supervision signal comes from photometric difference between the synthesized view and its corresponding reference view. In this chapter we introduce the theory of view synthesis under rigid-scene assumption first, and then discuss the case with considering the object movement.

3.1 View Synthesis under Rigid Scene Assumption

The problem of view synthesis can be set up as follows: Suppose a rigid scene is recorded, as shown in Figure 3.1. Two views are captured for this scene from different viewpoints, say the *reference view* I_{ref} at P_1, and the *source view* I_{src} at P_2. Now we want to synthesize the I_{ref} from the I_{src}. This means for each pixel on the synthesized view $\hat{I}_{ref}$, an intensity is chosen or interpolated from the intensity of I_{src}. The ground truth of $\hat{I}_{ref}$ is actually already provided, as I_{ref}.

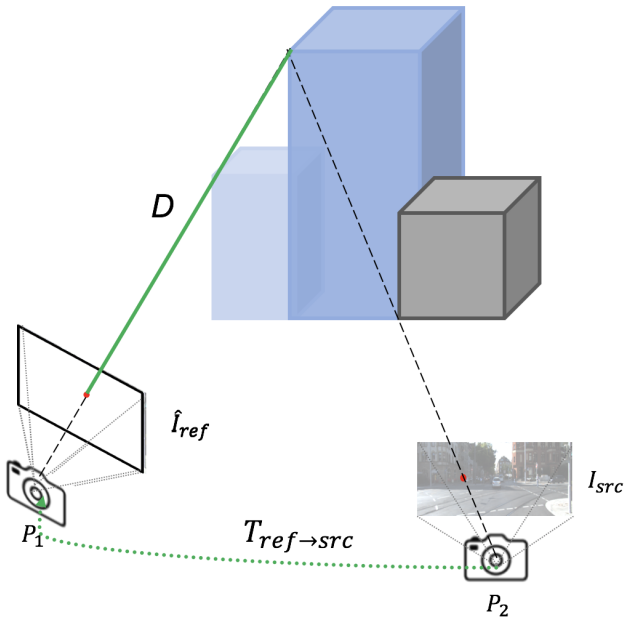

Figure 3.1: Process of view synthesis

To synthesize a view, a dense optical flow map from I_{ref} to I_{src} is required. This flow map provides the point correspondence of each pixel on the I_{ref}. The process of view synthesis can be generalized as three steps: pixel projection into the 3D space, camera coordinates system transformation, and projection onto the image plane of I_{src}.

Firstly pixel on the I_{ref} image plane is projected into the 3D space. This process can described as Equation 3.1:

$$P^{ref} = D_{ref}(p_t)K^{-1}h(p_t) \tag{3.1}$$

K is the camera intrinsics while $D_{ref}(p_t)$ is a scalar which indicates the depth of pixel p_t. $h(p_t)$ is the homogeneous pixel coordinate. After this projection the 3D location of p_t in the reference camera coordinate system C^{ref} is determined, denoted as P^{ref}. In order to find its corresponding pixel p_s on I_{src}, a coordinate transformation is needed to transform P^{ref} to P^{src}, which is with respect to the source camera coordinate system C^{src}:

$$P^{src} = T_{ref\to src}\, P^{ref} \tag{3.2}$$

Here the $T_{ref\to src}$ is the transformation matrix from C^{ref} to C^{src}. $T_{ref\to src}$ has 6 *Degree-of-freedom*, which are corresponding to rotation and translation respectively. After the transformation, the 3D point is projected onto the source image plane with the camera intrinsics, then point correspondence between I_{ref} and I_{src} is established. The whole synthesized process can be described by Equation 3.3 :

$$h(p_s) \sim KT_{ref\to src}\, D_{ref}(p_t)K^{-1}h(p_t) \tag{3.3}$$

Provided with the dense point correspondence between I_{ref} and I_{src}, the I_{ref} can be synthesized by filling the intensity of its corresponding pixel I_{src}. Normally the correspondence is a continuous value while pixels are discrete as grids. Then the bi-linear sampling strategy is adopted to interpolate the intensity from the four neighbouring pixels, as illustrated as Figure 3.2:

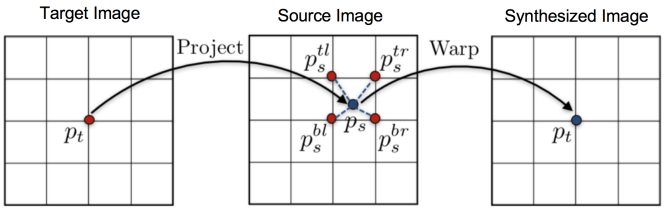

Figure 3.2: Bilinear sampling, adapted from Zhou et al. (2017)

The equation to interpolate the intensity of p_s is formulated as 3.4, where w_{ij} is linearly proportional to the spatial proximity between p_s and p_s^{ij}.

$$\hat{I}_s(p_t) = I_s(p_s) = \sum_{i\in\{t,b\},j\in\{l,r\}} w^{ij} I_s(p_s^{ij}) \tag{3.4}$$

In summary, three pieces of information are needed to synthesize I_{ref} from I_{src}:

1. The dense depth map of the reference view $D(I_{ref})$. The depth scalar can determine the 3D point location in the C^{ref} coordinate system during the 2D to 3D transformation.

2. The transformation matrix $T_{ref \to src}$.

3. The camera intrinsics K, which is used in the conversion from image plane into 3D space, or vice versa.

3.2 View Synthesis for Dynamic Scene

View synthesis under *rigid-scene assumption* presupposes the consistency between P^{ref} and P^{src}. That is during the capture of I_{ref} and I_{src}, no object movement occurs. However, this assumption is generally not true in the driving scene, where many moving objects like vehicles or pedestrains are presented.

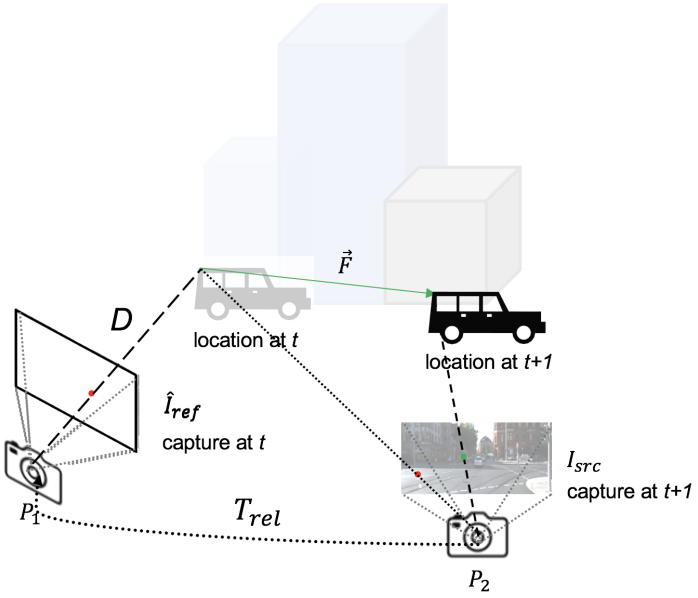

Figure 3.3: View synthesis for dynamic scene

To explain the dynamic scene, a 3D scene flow map $F(I_{ref})$ for the reference view is introduced during synthesis. $F(I_{ref})$ records the pixel-wise moving vector during the capture of I_{ref} and I_{src}. The scene flow vector is defined with respect to the reference camera coordinate system C^{ref}. This synthesis process is illustrated in Figure 3.3 and can be formulated as Equation 3.5:

$$h(p_s) \sim KT_{ref \to src} \left[D_{ref}(p_t) K^{-1} h(p_t) + F_{ref}(p_t) \right] \tag{3.5}$$

Similar with the view synthesis under *rigid-scene assumption*, the synthesis for dynamic scene can also be divided into three steps, as explained in the previous section. However, after the projection of pixel from image plane into the 3D space, the 3D point is further transformed according to the scene flow vector $F_{ref}(p_t)$. This 3D vector describes the object movement between time *t* to *t+1*. The remaining synthesis process is the same with the one for rigid scene.

Chapter 4

Methodolodgy

In this chapter the methodolodgy to estimate the object movement from temporal image sequence is presented. We first set up the problem, then detail those challenges and proposed solutions during the estimation in the second section. The design of network architecture is explained in the third section. We conclude this chapter with the formulation of loss function in the fourth section.

4.1 Problem Setup

Our system takes the temporal sequence of calibrated stereo image pairs as input, denoted as $\{I_t^l, I_t^r\}$ and $\{I_{t+1}^l, I_{t+1}^r\}$. The ultimate purpose is to estimate the dense scene-flow map $F_{t\to t+1}^l$ for the left temporal image sequence. The $F_{t\to t+1}^l$ describes the pixel-wise 3D moving vector in the left camera coordinate system, at time *t*.

In the previous chapter we mention that besides the dense scene-flow map, another three pieces of information are required for view synthesis: the dense depth map $D[I_{ref}]$, the transformation matrix $T_{ref\to src}$, and the camera intrinsics *K*. In our proposal the $D[I_{ref}]$ is predicted by a depth network. The transformation matrix of camera from *t* to *t+1* is pre-computed using the open-sourced visual odometry library [11]. The camera intrinsics is also provided in the dataset and only required to be re-scaled accordingly with the input images. In summary, our system estimates the depth map as well as the dense scene-flow map from the input.

4.2 Challenges and Proposals

4.2.1 Ambiguity between Depth and Scene Flow

As explained in Chapter 3, during synthesizing a view of a dynamic scene, the pixel depth is required to project an image point into the 3D space, then the scene-flow vector is required to further translate this 3D point. The photometric consistency between the synthesized view $\hat{I}$ and its reference view I provides the primary supervision signal.

However, using the photometric difference between $\hat{I}$ and I alone is not sufficient to supervise the depth and scene-flow prediction, due to their innate coupling in the 3D space. This ambiguity is illustrated in Figure 4.1 and 4.2. In these two figures, the combined depth and scene flow prediction of both d_1, F_1 and d_2, F_2 are able to establish the point correspondence between p_t and p_s. This indicates that a minimized photometric difference doesn't necessarily suggests an accurate depth and scene flow prediction.

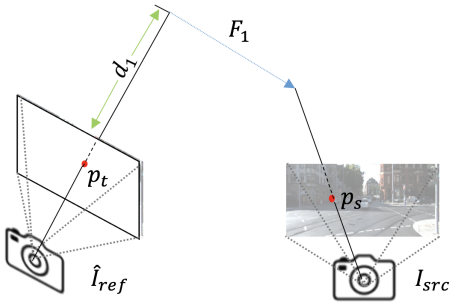

Figure 4.1: Synthesized with d_1 & F_1

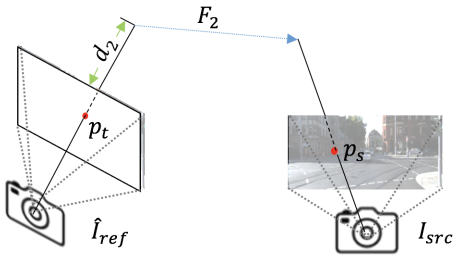

Figure 4.2: Synthesized with d_2 & F_2

This ambiguity can either be fixed by introducing extra constraints. In our system the photometric consistency between the calibrated stereo image pairs is introduced, besides the consistency across the temporal image sequence. The idea behind is, the stereo images pairs have been synchronized with each other, thus there are no object movement during the capture of the left and right image. In other word, the left-right photometric consistency is free of the above ambiguity thanks to the exclusion of object movement (also the scene flow vector). The left-right photometric consistency alone is sufficient to supervise the depth prediction, and can thus correct the bias caused by the depth and scene flow ambiguity.

Another solution is to learn the scene flow vector based on an accurate depth prediction. As mentioned in Chapter 2, the unsupervised monocular depth estimation can already provide an accurate depth result. The idea is to fix the model parameters of the depth network and train the scene-flow network alone. Although with introducing the left-right photometric consistency it's possible to jointly optimize the depth and scene-flow network, a good depth network can provide a good starting point to learn the scene-flow map.

To acquire a good depth estimation, the synchronized stereo image pairs are used to train the depth network in the first place. We don't use the depth estimation from the monocular video, since they are limited for the rigid scene.

4.2.2 Sparsity of Moving Object

Another challenge is the sparsity of the moving object in the scene. As shown in Figure 4.3, those moving objects only constitute a small proportion, while most part of the scene are actually rigid background.

The photometric consistency provides the primary supervision in our system. Normally this loss term is computed by first computing the photometric difference between the synthesized and its reference view, and average over the whole image plane.

However if the photometric loss is computed in this way, then the difference resulted from the moving object would be totally overwhelmed by the more dominant rigid background. In this way, the supervision signal for scene-flow vector is reduced, which makes the training of scene-flow network rather difficult.

To fix this unbalanced pixel distribution of static and dynamic area, a RCNN-based architec-

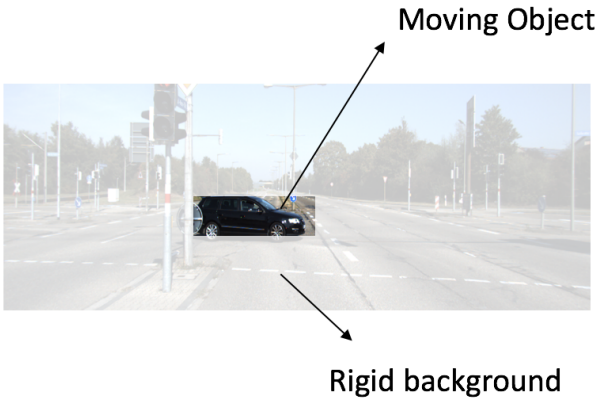

Figure 4.3: Sparsity of Moving Object

ture is adopted for the scene-flow network. In particular, the scene-flow map is only predicted for specific *Region-of-Interest* (ROI), instead of for the whole image plane. With the ROI scene-flow prediction, the constraint of photometric consistency can be directly imposed on this area.

In our system these ROI is extracted by the *Single Shot Multibox Detector* (SSD) [21], a *start-of-the-art* object detection system which extracts object in the form of 2D bounding box. The object detection is conducted over all images in the training dataset. All extracted ROIs with arbitrary size are resized into a fixed size, and feed into the scene-flow network. It is important to note that the scene covered by the extracted ROI is not necessarily dynamic. The ROIs are selected only because some objects are contained in these areas.

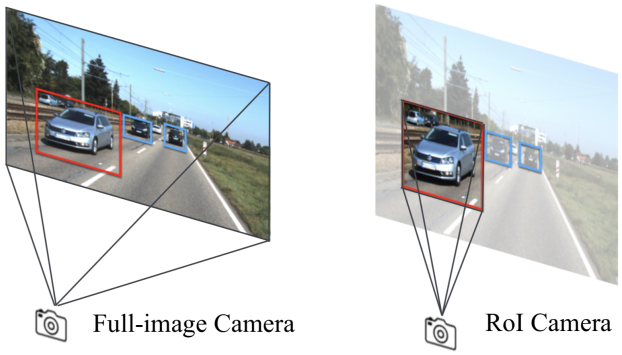

Figure 4.4: ROI Crop and Resize, from Zhe et al.(2019)

The images of ROI should be synthesized as well in order to impose the photometric consistency. This synthesis process basically follows the steps described in Chapter 3. However due to the ROI extraction, the camera intrinsics for each ROI is different with the one for the whole image. The process of cropping and resizing ROI is illustrated in Figure 4.4.

Suppose the location and the original size of the extracted ROI j is $[x, y, w, h]$. Then the ROI is resized into $[w_r, h_r]$. The transformation of camera intrinsics is formulated as equation 4.1.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad K^j = \begin{bmatrix} f_x w_r/w & 0 & (c_x - x)w_r/w \\ 0 & f_y h_r/h & (c_y - y)h_r/h \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

The scene-flow prediction for the full image is also assembled from each ROI. This is done by resizing the ROI prediction into its original size, and filling the full scene-flow map with prediction from corresponding location. For ROIs overlapped with each other, only one prediction will be chosen for filling. Which ROI to be chosen depends on the order in the generated bounding box.

4.2.3 Geometric Constraint: Supervise the Scene-Flow Prediction

When synthesizing an image, the depth of the reference view (where the image is synthesized to), as well as its scene-flow map are required. The depth of the source view (where the image is synthesized from) is not needed. However, by utilizing the source depth map together with the reference depth, a supervision for the scene-flow prediction can be established.

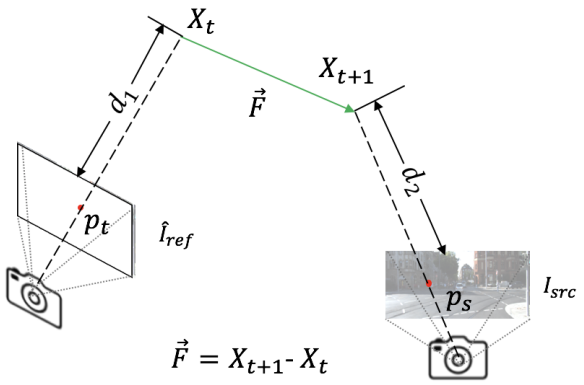

Figure 4.5: Geometric Constraint for Scene-flow Prediction

In our system this supervision is named as *geometric constraint*. The geometric constraint is illustrated in Figure 4.5. Here X_t and X_{t+1} are locations of the same point at different time instant in the 3D space. If the coordinates of X_t and X_{t+1} in the reference camera coordinate system are known, then its scene-flow vector can be formulated as $\mathbf{F} = X_t - X_{t+1}$. This geometric constraint provides a direct pixel-wise supervision on the scene-flow prediction.

To impose the geometric constraint, the depth map of both the reference and the source view are needed. The corresponding image points are projected into the 3D space with the depth information. Then X_{t+1}, which is originally in the source image coordinate system, is transformed into the reference system. After the transformation the geometric constraint for this scene-flow vector is available.

When computing the geometric constraint, the point correspondence (the 2D optical-flow map) between the reference and the source view is also required. That is, for each pixel p_t on I_{ref}, its corresponding pixel p_s on I_{src} are expected to be known in advance. This is because we have to make sure the projected 3D point X_t and X_{t+1} are exactly the same points. Thus in our system,

we also include a network to provide the optical-flow map. This network is only used in the early training stages and is excluded later. The idea is, after training several iterations, the scene-flow network can provide a relatively accurate prediction. Then the optical-flow map can be synthesized with the reference depth and the scene-flow map. We can use this point correspondence to supervise the scene-flow prediction.

By imposing the geometric constraint, the information of the source depth map is used. The integration of source depth fixes the ambiguity of the end vertex of the scene-flow prediction. As illustrating in Figure 4.5, the starting vertex of scene-flow vector X_t is determined by the reference depth d_1. If the source depth d_2 is not used, then the end vertex of scene-flow vector X_{t+1} can be arbitrarily predicted alone the projection ray of p_s on I_{src}, without violating the photometric consistency. The introduction of geometric constraint will penalize the inaccurate prediction of the end vertex, and encourage an accurate scene-flow prediction.

4.3 Pipeline and Network

Our pipeline is illustrated in Figure 4.6. Two components are included: depth and scene-flow prediction. The upper part of this figure refers to the depth prediction, where a depth map for the whole input image is predicted. The bottom part refers to the scene-flow prediction, where the scene-flow map for each ROI is predicted.

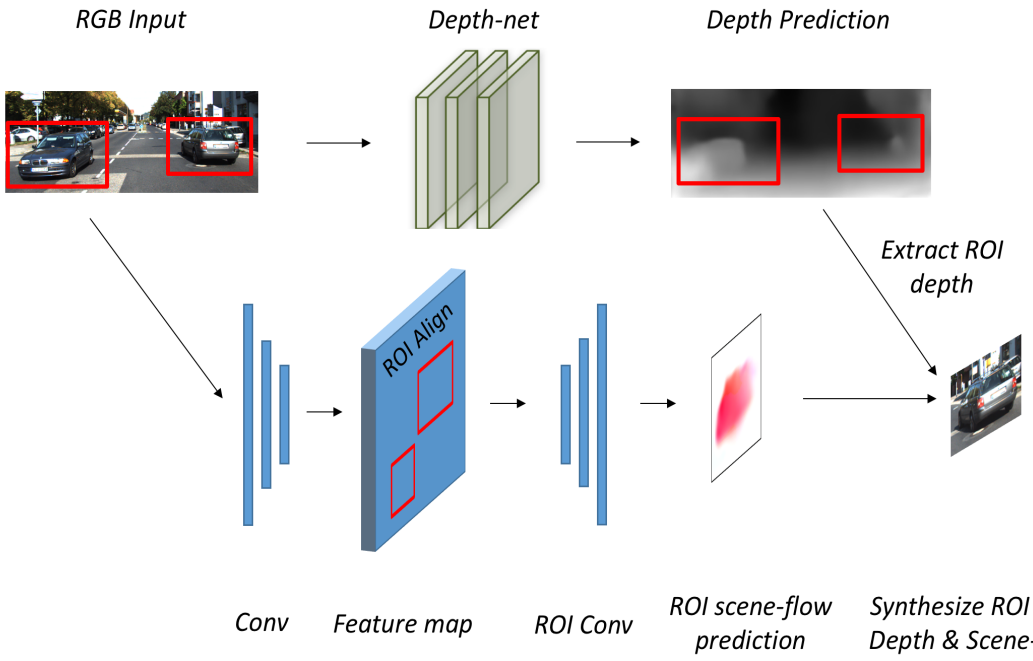

Figure 4.6: Pipeline Overview

For both depth and scene-flow network, an encoder-decoder architecture is adopted. In encoder the network inputs are convoluted to produce feature maps. While in decoder those down-scaled feature maps are convoluted and scaled into a higher resolution, and finally provide the depth and scene-flow prediction. To overcome the gradient locality, predictions at multi-scale are provided. Both photometric and geometric consistency are imposed on these multi-scale depth and scene-flow predictions. The skip connections are also adopted between the encoder and decoder.

In our system the standard Resnet-50 [14] structure is used for the encoder. The details of the encoder is illustrated in Figure 4.7. Each residual block contains several residual units whose structure is shown in Figure 4.8. There are four residual blocks in the encoder part and containing 3, 4, 6, 3 residual units respectively. The spatial size of feature map is halved after each residual block. And the skip connections for the decoder are provided from the first convolutional layer (Conv 1), the first pooling layer (Pool 1), as well as the first three residual blocks.

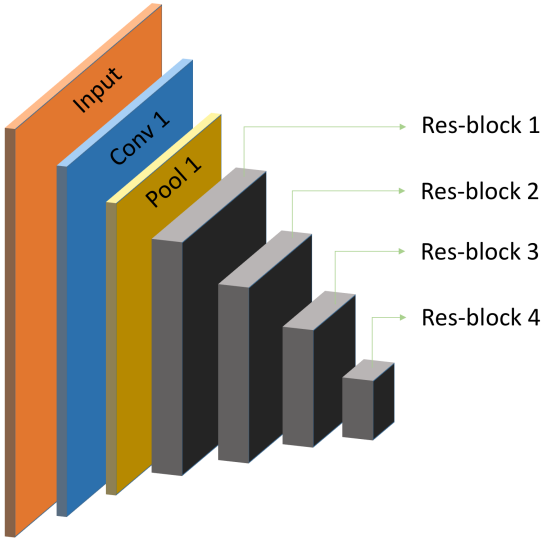

Figure 4.7: Network encoder

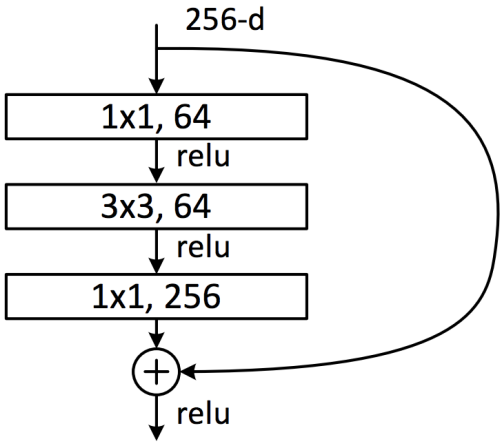

Figure 4.8: Residual unit [14]

The depth network takes RGB images as input and predict the multi-scale depth maps ($\frac{1}{8}, \frac{1}{4}, \frac{1}{2}$ and the full spatial size) for the whole image. For scene-flow network, multiple maps are concatenated alone the channel dimension and used as network input. In our system we concatenate the reference and source RGB image, the synthesized reference image under rigid-scene assumption (synthesizing with depth map only), the depth prediction for the reference and the source view, etc. It is important to note that the input of scene-flow network contains the prediction from depth network.

The decoder of scene-flow network takes feature maps aligned with ROIs as input. The feature maps from encoder are cropped and resized into a fixed size according to the normalized ROI coordinates, then concatenated along the batch dimension. The skip connections from a higher resolution are also cropped and resized into a fixed size, then concatenated with the up-convoluted feature maps along the channel dimension. Similar with the depth network, the scene-flow network also provide multi-scale scene-flow predictions. The scene-flow predictions with size $128 \times 128, 64 \times 64, 32 \times 32, 16 \times 16$ are provided for each ROI, regardless of their original size.

4.4 Formulation of Loss Function

In our system multiple losses are employed, including the photometric loss across temporal image sequence, the left-right photometric loss between the stereo image pairs, the geometric loss of the scene-flow prediction, and the smoothness of disparity and scene-flow prediction. Their supervisions towards depth and scene-flow network are summarized in Table 4.1.

Table 4.1: Summary of supervision of different losses

	Depth Network	Scene-flow Network
Photometric loss	✓	✓
Left-right photometric loss	✓	×
Geometric loss	✓	✓
Disparity smoothness	✓	×
Scene-flow smoothness	×	✓

The scene-flow map is predicted for each ROI and is assembled together to form the full scene-flow map. Those losses which are related to the ROI scene-flow prediction, is imposed on both ROI region and on the full image. These relevant losses include the photometric loss, the geometric loss and the scene-flow smoothness loss. Each of them is separated as ROI-based loss and full loss.

Take the example of photometric loss. The full loss is the average of the photometric difference between the full synthesized view and its reference view. The full synthesized view is based on the depth prediction and the fully-assembled scene-flow map. The robust image similarity measurement is used [12], which is a combination of an L1 and single scale SSIM term:

$$L_{photo} = \alpha \frac{1 - SSIM(I_{ref}, \hat{I})}{2} + (1 - \alpha)\|I_{ref} - \hat{I}\|_1 \tag{4.2}$$

In our implementation the α is set as 0.85. To compute the ROI-based photometric loss, the ROI depth is first cropped from full depth map prediction, and resized into the same size as its scene-flow prediction. Then the ROI is synthesized following the common process of the view synthesis, except the camera intrinsics for the reference view is changed according to the ROI size and location.

The left-right photometric loss L_{lr_photo} is also computed according to Equation 4.2. However the left or right image is synthesized according to the disparity map, which is the direct output of the depth network. The previously mentioned view synthesis is not used in formulating L_{lr_photo}. The used stereo image pairs have already been synchronized and rectified with each other, so the horizontal offset provided by the disparity map is sufficient to synthesize the left view from the right one, or vice versa.

The geometric loss is also separated as the ROI-based one and the full one. The geometric constraint $\mathbf{F}_{geom}$ is computed first. The $\mathbf{F}_{geom}$ is computed based on the depth prediction of the reference and the source view, and the optical-flow map of these two images. Then the absolute difference $\mathbf{F}_{diff}$ between the geometric constraint and the scene-flow prediction $\mathbf{F}_{pred}$ is computed as equation 4.3. The geometric loss L_{geom} is the average of the $\mathbf{F}_{diff}$ either over the ROI or over the whole image plane.

$$\mathbf{F}_{diff} = \|\mathbf{F}_{geom} - \mathbf{F}_{pred}\| \tag{4.3}$$

The smoothness loss for depth and scene-flow prediction are introduced to penalize a fluctuated prediction. They are denoted as L_{disp} and L_{sf} respectively. It is important to note that the depth smoothness loss is actually imposed on the disparity prediction, and L_{sf} is also separated into the ROI and the full loss. We suppose the depth and the scene-flow discontinuities occurs at areas with high image gradient, which is normally the boundary of objects. When formulating

the smoothness loss, the depth or scene-flow discontinuities are weighted by image gradient. The lower the gradient is, the more penalization is given:

$$L_{disp} = |\partial_x d| e^{\|-\partial_x I\|} + |\partial_y d| e^{\|-\partial_y I\|} \tag{4.4}$$

The computation of L_{disp} is formulated in 4.4, where the $\partial_x d$ and $\partial_y d$ are the discontinuities of disparity prediction alone x and y dimension respectively, while $\partial_x I$ and $\partial_y I$ are the image gradient along the horizontal and vertical direction. The L_{sf} is computed in a similar manner.

The final loss is a weighted average of all loss mentioned above. The weight is carefully chosen so that one loss won't be totally overwhelmed by another.

$$L = L_{photo} + L_{lr\_photo} + L_{geom} + L_{disp} + Lsf \tag{4.5}$$

Chapter 5

Experiments and Results

We conduct some experiments to demonstrate the effectiveness of our proposed pipeline. In this chapter the experiment setting and the result is given. We detail the data source as well as data preprocessing in the first and second section, then display various evaluation results in the third part. Some result analysis is also given.

5.1 Data Sources

5.1.1 KITTI

We train and evaluate our system on the widely used KITTI dataset [10]. Stereo images pairs in KITTI have been rectified and synchronized with each other. Relevant camera parameters like the intrinsics or the baseline length are also available.

Two categories of dataset are used, including the KITTI raw dataset and the scene-flow 2015 dataset. Raw dataset contains various scenes, like the scene of city center which contains many moving objects, or the residential part which is generally rigid. In contrast, scenes covered by the scene-flow 2015 dataset are highly dynamic, with many moving objects. In our experiment, the raw dataset is used to train the depth network and initialize the training of the scene-flow network, while the scene-flow dataset is used to fine tune the scene-flow network further.

Some preprocessing steps are necessary to feed KITTI images into our system:

- Image Resize: The typical original size of image in KITTI dataset is 1242×375. In order to fit the GPU memory, images are resized into 640×192 in advance. The camera intrinsics are supposed to be resized accordingly, as shown in Equation 5.1. Here the r_x is the ratio between the width of the resized image and its original width. The same goes for r_y.

$$K = \begin{bmatrix} f_x \cdot r_x & 0 & c_x \cdot r_x \\ 0 & f_y \cdot r_y & c_y \cdot r_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

- Concatenation: The view synthesis requires a temporal image sequence. In our system the number of frame inside a sequence is set as three. These three images (after resized) are concatenated horizontally, and ordered from left to right as $t-1$, t and $t+1$. Here the t refers to the capture time of its corresponding frame.

The scene-flow 2015 dataset is composed of two subsets, training and testing set. Both sets contains 200 temporal image sequences (the sequence length is 2), while the training set contains the annotations of disparity, 2D optical flow, etc. Thus the training set is adopted for evaluation, where the dense 3D scene-flow map can be generated from the ground truth annotation. The multi-view version of the test set is used for fine tuning the scene-flow network.

To achieve a fair evaluation, all scenes covered by the scene-flow training set is excluded from the raw dataset when training the scene-flow network. We also exclude scenes where no object(ROI) is detected. After the concatenation and the removal of relevant frames, there are in total 7089 sequences for the scene-flow dataset, and 22038 sequences for the raw dataset.

5.1.2 Cityscapes

The Cityscapes dataset [7] is also used for improve the performance of depth-network. Compared with KITTI, scenes in Cityscapes contains more moving objects. As indicated in the work of Godard [12], the performance of depth network is improved by first training on KITTI raw dataset,then training on the Cityscapes dataset. The same preprocessing steps are also applied on images in this dataset.

5.2 Generation of ROI and Camera Pose

5.2.1 Generation of ROI

In Chapter 4 we mention that a R-CNN based architecture is adopted to address the sparsity of moving objects in the scene. Thus the ROIs of the image which contains the potentially moving objects are supposed to be extracted in advance. In my thesis the SOTA object detection system *Single Shot Multibox Detector* (SSD) [21] is used. SSD takes RGB image as input, and output the normalized coordinates of a 2D bounding box on the image. The 2D bounding box highlights the object in the scene.

The published SSD model can detect around 20 different kinds of object. We choose four of them which are able to move, including bicycle, bus, car and pedestrians. The employed model has been trained on the Pascal VOC 2007, 2012 and COCO, without fine-tuning on the KITTI dataset.

However, different images have different number of ROIs. In order to feed those ROIs into the system, those normalized coordinates, which is originally stored as text file, should be converted into the image(PNG) file format which always have a fixed spatial size. In particular, an 2D matrix with the same size of the RGB image is created. The value of elements located in background area is zero, while value in ROI area is a positive integer, depends on which ROI it belongs to. This 2D matrix is stored as PNG file.

During the conversion from normalized coordinates into PNG image, the spatial size of each ROI is enlarged a bit. That is, the width and the height of the ROI is extended by 10% per side. The enlargement of ROI is aimed to include the moving objects in different frames, e.g: I_t and I_{t+1}.

5.2.2 Generation of Camera Pose

The camera pose describes the transformation between the camera coordinate system of different frames inside a temporal image sequence. The pose can either be represented as a 4×4 rotation-translation matrix, or a vector with 6 degree-of-freedom: $[x, y, z, \phi, \omega, \kappa]$. The first three elements of the vector describe the translation, while ϕ, ω, κ are rotation angles of X, Y, Z axis respectively. In our system the rotation order of Z-Y-X is adopted.

$$R = R_x \times R_y \times R_z \tag{5.2}$$

We use LIBVISO2 [11] to solve the camera motion in our system. LIBVISO2 is a C++ library for computing the 6 DOF motion of a moving mono- or stereo camera. The stereo version is adopted due to its better accuracy and flexibility. The camera motion is solved by minimizing the re-projection error of sparse feature matches, based on the RANSAC algorithm.

LIBVISO2 takes stereo rectified image pairs as input, say $\{I_t^l, I_t^r\}$ and $\{I_{t+1}^l, I_{t+1}^r\}$, respectively. Then the rotation-translation matrix of the left camera coordinate system is returned. The camera intrinsics, as well as the baseline length should also be provided.

5.3 Implementation Details

Our system is implemented using TensorFlow [1] and trained using a single NVIDIA Titan-X GPU. Both networks in our system, the depth and scene-flow network, are trained end-to-end using Adam optimizer [16]. The rectified liner units (ReLU) [24] is adopted as the activation function, except for the prediction layer. The batch normalization is also applied for both two networks.

The color data augmentation is performed on the fly, at a chance of 50%. The random gamma, brightness and color shifts are performed by randomly sampling from a uniform distributions, in the ranges [0.8,1.2] for gamma, [0.5,2.0] for brightness, and [0.8,1.2] for each color channel separately. In early work augmentations like random scaling and cropping are also performed. We exclude them in our system, because the ROI detection is performed in advance and can not be changed accordingly with the scaling and cropping of image.

A stage-wise training strategy is adopted for optimizing these two networks separately, with training the depth network first and then scene-flow network. The training details are explained in the following part.

5.3.1 Depth Training

Basically the training of depth network is separated into four stages, where different loss functions and training dataset are used. For all stages, the batch size is set as 4. The penalty for the smoothness of disparity prediction is also imposed across all these four stages. Since a multi-scale disparity maps are output and typically their value will differ by a factor of two between each scale (due to the down-scaling of the image size), the weight for the disparity smoothness loss is scaled accordingly, to get equivalent smoothing at each level.

In the first stage, the photometric loss of the rigid warped image is used. In particular, this rigid warped image $\hat{I}_t$ is synthesized from I_{t+1} (or I_{t-1}), based on the depth prediction. No photometric consistency between the stereo images is considered. The employed losses are the rigid warped photometric loss and the disparity smoothness loss, with their weight setting as 1.0 and

0.5 respectively. The encoder of depth network is initialized from the a model of Resnet-50, pre-trained on ImageNet. While the decoder is initialized using the Xavier initializer. The KITTI raw dataset is used as training dataset. The learning rate for Adam optimizer is 2e-4. Then the network is trained for 100K iterations, taking around 14 hours.

However, the depth prediction from the first stage is not accurate enough, because the input image doesn't satisfy the rigid-scene assumption. In the second stage, the rigid-warped photometric loss is replaced by the left-right photometric loss L_{lr_photo}. L_{lr_photo} is the averaged difference between $\hat{I}_l$ and I_l, $\hat{I}_r$ and I_r. Here I_l and I_r are the rectified stereo image pairs while $\hat{I}_l$ and $\hat{I}_r$ are synthesized images from the disparity prediction. The weight for L_{lr_photo} and L_{disp} are 1.0 and 20.0 respectively. The learning rate is 1e-4. In the following third and fourth stages the weight for L_{disp} and the learning rate are kept as the same value. Then the network is trained for 200K iterations.

In the third stage Cityscapes dataset is used for training the depth network, as we aim to improve the generalization performance of the network. The depth network is trained on the Cityscapes dataset for 300K iterations with L_{lr_photo} and L_{disp}, then fine-tuned on the KITTI raw dataset for another 200K steps in the fourth stage.

5.3.2 3D Scene-flow Training

Three losses are employed for optimizing the scene-flow network, including the dynamic warped photometric loss L_{photo}, the geometric loss L_{geom}, and the smoothness loss of the predicted scene-flow map L_{sf}. Their corresponding weight are 1.0, 0.5 and 0.5 respectively. Different with the disparity smoothness in the depth network, the weight for L_{sf} should not be scaled for different level. This is because the multi-scale scene-flow predictions are in the unit of meter and is relevant with its own spatial size.

The non-linearity used in the prediction layer is the sigmoid function. Its output is multiplied with 10 and then adding a bias of -5. This scaling and bias indicating that we assume the maximum speed of the moving object is 360 kilometers per hour.

The formulation of geometric constraint requires an optical flow map from the reference to the source frame. A flow network is introduced to provide a better flow map. In our system the *direct flow network* from Geonet [31] is used. This network takes the concatenation of reference and source view as input, then output their optical flow. The direct flow network only provides the flow map and won't be trained in our system. It is also important to note that when the network is able to provide a reasonable scene-flow prediction, the optical flow can be synthesized based on the network output. Then the flow network can be excluded.

When training the scene-flow network the batch size is 2, with one of them is the left temporal image sequence while the other is the right sequence. The learning rate is 1e-4. The network is directly trained on the KITTI scene-flow 2015 dataset. We first train the scene-flow network for 300K iterations with using the flow network to produce the flow map. Then the flow network is excluded and the scene-flow network is trained for another 700K iterations.

5.4 Results

Basically four evaluation metrics are used to display the performance of our approach. We first evaluate the instance-level moving object speed and direction, which can be directly inferred from the prediction of scene-flow network. Secondly the depth map is also predicted and compared with other approaches. We also evaluate the optical flow map from I_t to I_{t+1}, which is synthesized based on the depth and scene-flow prediction.

The scene-flow and optical flow prediction is evaluated on the training subset of KITTI scene-flow 2015 dataset, where the ground truth annotation for disparity and optical flow map is available. While for depth evaluation, two different test splits are used to enable a complete comparison with existing works.

5.4.1 Evaluation on Moving Object Speed and Direction

The output of scene-flow network is the dense 3D scene-flow map from I_t to I_{t+1}. Each pixel corresponds to one 3D vector which encapsulates its movement between time t to $t + 1$. The magnitude of this vector is the moving speed, while its unit vector indicates its direction.

The KTIIT sceneflow 2015 dataset provides the annotation for disparity, optical flow map as well as the masks which highlight moving objects in the scene. The scene-flow ground truth can be synthesized from these annotations, followed the procedure explained in 4.

Our network provides the instance level scene-flow prediction in form of ROI. Then the scene-flow map for the full image is assembled by resizing and filling all ROIs. To evaluate the moving speed and direction, a single dominant 3D moving vector for each object is computed. Then the magnitude difference between ground truth and predicted moving vector is the error of moving speed, while the angle difference is the error of direction.

The dominant moving vector is computed by averaging over an specific area of scene-flow map. This area is provided by logically overlapping the object mask and ROI area. The object mask is provided by the KITTI sceneflow 2015 dataset, while the ROI is actually stored as a normalized coordinates of a 2D bounding box for each image. When computing the moving vector of different objects inside one single image, we iterate over all available ROIs and conduct the logical-and operation with the object mask. Finally, the speed and direction error of all available instances (the overlapping of mask and ROI) are collected and averaged, to provide the evaluation metrics.

Table 5.1: Evaluation of object motion

	Dir E. Mean	Dir E. Med	Speed E. Mean	Speed E. Med
Geonet [31] + Monodepth [12]	33.192	10.465	0.633	0.387
B2F [15] + Monodepth [12]	33.609	**6.573**	0.845	0.324
Our 3D scene flow	**23.803**	8.448	**0.435**	**0.291**

The average/median prediction error of the object movement is shown in Table 5.1. To enable a comparison, the 3D scene flow is computed based on the optical flow and depth prediction from existing works. The employed depth model comes from Godard [12], which is able to provide depth prediction without scale ambiguity. For optical flow prediction, the models from *SOTA* unsupervised approach are choosen, namely Geonet [31] and Back2Future [15]. The mean and the

median error of the moving direction and speed prediction are displayed. The direction metric is in the unit of degree.

It can be seen that our approach achieves the best performance in almost all metrics, except for the median direction prediction. While our mean direction angle prediction is much smaller than other two approaches, which is only 23.803 degree. We also have a significant better performance in terms of the mean speed prediction, which halves its counterpart from B2F and Monodepth. The median error of the speed prediction is also the best.

Table 5.2: Inlier of object motion prediction

	Dir E. $\leq 15^\circ$	Dir E. $\leq 30^\circ$	Speed E. ≤ 0.15	Speed E. ≤ 0.3
Geonet + Monodepth	59.2%	71.5%	25.2%	42.6%
B2F + Monodepth	63.1%	70.4%	**27.2%**	48.5%
Our 3D scene flow	**63.3%**	**75.6%**	24.7%	**52.2%**

Table 5.2 shows the percentage of object movement prediction error which below a certain threshold. A higher percentage indicates a more accurate estimation. It can be seen that our approach still achieves the best performance in almost all metrics. More than 75% of the moving objects have an angular direction error below 30°, and more than 50% have an speed error below 0.3 meters.

It is important to note that the previous work from Cao [5] also provide some evaluation results of scene flow prediction. However he doesn't explain his evaluation procedure nor provide his prediction. Thus it's not possible to compare our prediction with his.

5.4.2 Evaluation on depth prediction

Two test splits are used for evaluating the depth prediction in previous work. One is the eigen split [8], which uses 697 images for evaluation. The ground truth depth map is generated by re-projecting the 3D Velodyne laser point onto the image plane. To enable comparison with existing works the generated depth map is cropped in the same manner as in [8], where the sky and the bottom ground part is excluded from the evaluation. Both the ground truth depth map and its prediction are capped by 80 meters when computing the error.

The other set is called KITTI split. There are 200 images with ground truth disparity available. These disparity maps are annotated in a semi-automatic process and is theoretically more accurate than depth map generated from Velodyna laser points. However the Eigen split is more frequently used in terms of depth evaluation. We provide the evaluation on both splits.

All our depth predictions have the same size as the network input (640×192) and are supposed to be resized into the same size of the ground truth, which is typically 1242×375. The bi-linear interpolation method is adopted for resizing the depth prediction.

Table 5.3: Depth evaluation over different training stages (Eigen)

	Abs Rel	Sq Rel	RMSE	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Rigid L_{photo} Only	0.1313	1.0652	5.4663	83.36%	94.10%	97.47%
With L_{lr_photo}	0.1287	0.9456	5.1016	84.35%	94.75%	**97.81%**
CS fine-tuned	**0.1237**	**0.9357**	**5.0358**	**85.44%**	**94.96%**	97.69%

We first show the improvement achieved by training through different stages. Table 5.3 shows the evaluation of models from different stages on the Eigen split. It can be seen that by using the L_{lr_photo}, the performance has been improved a lot, with the RMSE dropped from 5.4663 to 5.1016. It shows that the left-right photometric consistency can provide a more accurate supervision signal than the rigid view synthesis. The row of 'CS fine-tuned' shows the evaluation after trained on the Cityscapes (CS) dataset. Some improvement have been further achieved by training on the more dynamic CS image, as its performance indicators are almost among the best.

Table 5.4: Depth evaluation over different training stages(KITTI)

	Abs Rel	Sq Rel	RMSE	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
With L_{lr_photo}	0.1209	1.8306	5.963	88.1%	95.7%	98.1%
CS fine-tuned	0.1126	1.8075	5.598	89.2%	96.2%	98.3%

Benefits gained by training on the CS dataset is more obvious by checking the evaluation on the KITTI split. As shown in Table 5.4, the results of CS fine-tuned model are superior to the predictions of model which trained only on the KITTI dataset. The RMSE drops from 5.963 to 5.598, and the percentage of accurate depth prediction (the last three columns of the table) are also higher. This is probably because the scenes covered by Cityscapes dataset are predominant in city, where more moving objects are presented. While test images in KITTI split are also all dynamic scenes. The similarity of covered scene improve the model performance as well as it generalization ability.

Table 5.5: Comparison of depth result with existing works (Eigen)

	Abs Rel	Sq Rel	RMSE	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
GeoNet	0.153	1.328	5.737	80.2%	93.4%	97.2%
Monodepth	0.124	1.076	5.311	84.7%	94.2%	97.3%
Monodepth2	**0.115**	**0.903**	**4.863**	**87.7%**	**95.9%**	**98.1%**
Ours	0.123	0.935	5.035	85.4%	94.9%	97.6%

The comparison of depth prediction with existing works is shown in Table 5.5. It can be seen that our depth model has achieved the *SOTA* accuracy. To enable to fair comparison all these chosen models have been trained on KITTI raw dataset and then fine-tuned on Cityscapes dataset. It is important to note that the depth prediction of Geonet [31] and Monodepth2 [13] are only up-to-scale. In order to evaluate the performance a scaling of depth prediction is needed. This scaling is done by multiplying with the median ground truth. The scale ambiguity comes from the fact that their approach use pose network to estimate the camera pose. Without the absolute scale information, the pose predicted by Pose network is defined only up-to-scale.

By contrast, the depth predicted by our model and Monodepth [12] is free of scale ambiguity. In our system the camera pose is pre-computed for view synthesizing, while in Monodepth the left-right photometric consistency is directly imposed, thus its disparity prediction is already the absolute one.

Table 5.6: Comparison of depth result with existing works (KITTI)

	Abs Rel	Sq Rel	RMSE	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth	**0.1042**	**0.9501**	**5.191**	87.3%	95.9%	**98.5%**
Ours	0.1126	1.8075	5.598	**89.2%**	**96.2%**	98.3%

Table 5.6 shows the comparison of evaluation on KITTI split. It can be seen that our result has

achieved a similar accuracy.

The following pages show some visual results of our depth prediction.

Figure 5.1: Visual result of depth prediction

5.4.3 Evaluation of Optical Flow

Although we don't provide a direct optical flow prediction, the flow map can be generated based on the depth and scene flow prediction. We provide the evaluation of these generated flow maps in this section. The employed evaluation metric is the averaged end-point error. The end-point error

is the L2 norm of the difference between the flow ground truth and prediction. The evaluation is conducted on the KITTI scene-flow 2015 dataset, where 200 images with ground truth flow annotation is used for testing.

Table 5.7: Comparison of optical flow result with existing works

Method	Noc	All
GeoNet (Direct) [31]	**6.77**	12.21
UnflowC [23]	-	**8.80**
Geonet [31]	8.05	10.81
Ours	8.57	12.14

Table 5.7 compares our flow prediction results with existing works. It can be seen that our flow prediction is not perfect compared with other works. More analysis will be given in the next chapter.

Chapter 6

Discussion

In this part we explore more details of the results. As shown in Chapter 5 the evaluation on optical flow is not perfect. We visualize the error of optical flow prediction and analyze these results in the first section. We also jointly optimize the depth and scene flow network but don't manage to achieve an improvement. Details regarding the joint optimization is given in the second section. Some future works are proposed in the third section.

6.1 Exploration of Optical Flow Prediction

To explore the performance gap between our optical flow prediction and previous work, we visualize the error of flow map. The map is visualized by first cropped the end-point error value between 0 to 10. Then the map is normalized by dividing 10 and re-scaled by multiplying with 255. After this scaling pixel with a end-point error $\geq$ 10 has a intensity of 255.

Figure 6.1: Visualization of end-point error 1

Figure 6.1 and 6.2 shows some examples of end-point error visualization. Pixel in brighter area has a higher end-point error. It can be observed that the optical flow generated from the depth and scene-flow prediction could explain the object movement to some extent. However, this is not

true for all scenes. For example, the predicted scene flow can not compensate for the scene of the last row in Figure 6.2.

Figure 6.2: Visualization of end-point error 2

One reason for this is probably due to the network (direct version of Geonet [31], short for DirFlow-net) employed for providing optical flow is not accurate enough. Figure 6.3 is the visualization of flow prediction error from DirFlow-net. It can be seen that this prediction is not that good. This means the geometric constraint computed from this flow map is not accurate, which may provide the wrong supervision signal for our scene-flow network.

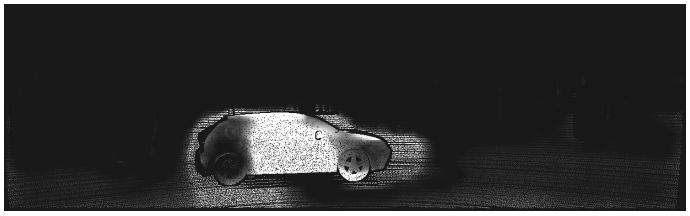

Figure 6.3: Flow prediction from DirFlow-net

6.2 Exploration of Joint optimization

The introduction of left-right photometric consistency makes it possible to jointly optimize the depth and scene-flow network. Theoretically the joint optimization can not only improve the scene-flow prediction, but also further improve the depth network performance.

However we don't gain an improvement by jointly optimizing depth and scene-flow network. Instead, both the depth and scene-flow network start to provide bad results when they are trained at the same time. One possible reason is that the depth prediction result is part of the input of scene-flow network. During the joint optimization, the output of depth network will be changed, while the scene-flow network can not adjust itself accordingly with this change. Thus the scene-flow network starts to fail. As a result, a wrong supervision signal is provided to depth network as well, since the photometric loss is computed from both depth and scene-flow prediction.

6.3 Future Work

To further improve the performance of optical flow prediction, the following works can be tried:

- Supervise the scene-flow network with a better optical flow prediction. Currently the employed geometric constraint is computed from the flow which is predicted by the direct flow version of Geonet [31]. The optical flow evaluation result of this model is not perfect as shown in previous chapter. By supervising with a better optical flow map, potentially the result will be improved.

- Training on Other dataset. The bad generalization ability is the other reason for the bad optical flow prediction. Some improvement may be gained by training on other dataset. One alternative is to train on the Cityscapes dataset, where more dynamic scenes are available.

As for joint optimization, the following methods may potentially improve the result:

- Normalization of depth input of the scene-flow network. Currently the depth prediction is directly feed into the scene-flow network, without any pre-processing. The depth prediction ranges from 0.2 to 100 in our system. However when convoluting the RGB image, it's a routine to re-scale the unsigned 8 bit input into float value between 0 and 1. We can try to normalize the depth map before feeding into the network.

- Recent work from Ranjan [26] using the competitive collaboration framework to jointly train the depth, pose, optical flow and moving object segmentation. This framework works in a similar way of *Expectation Maximization* and achieved a good result. This framework can be introduced into our system to jointly optimize depth and scene-flow network.

Chapter 7

Conclusion

In this work, we try to infer the object movement from video. Based on the idea of view synthesis, our system is able to learn the dense scene flow map of two consecutive frames in a self-supervised manner. Two networks have been designed to provide the depth and scene flow prediction respectively. A R-CNN architecture is adopted to compensate for the sparsity of the moving objects in the scene. The geometric constraint computed from 2D optical flow and depth is utilized to solve the scene flow ambiguity.

The experiment results on KITTI dataset shows the effectiveness of our system. The accuracy of our scene-flow prediction is better than the results generated from depth and optical flow from previous work. Moreover our depth prediction also achieves the *SOTA* performance.

In the future some more constraints can be introduced to get a better result, in particular to improve the performance of optical flow generated from the scene flow prediction. A network which predicts a better optical flow map can be introduced, to provide a better supervision on the scene flow map.

Appendix A

Depth Performance Indicator

In Chapter 5 6 numerical indicators are used to evaluate the depth prediction performance. These indicators are formulated as following. We use D_{pred} and D_{gt} to denote the ground truth and prediction of depth map. All computations are pixel-wise.

Abs Rel: short for *absolute relative error*, formulated as $\frac{1}{N}\sum|D_{pred} - D_{gt}| \ / \ D_{gt}$.

Sq Rel: short for *squared relative error*, formulated as $\frac{1}{N}\sum(D_{pred} - D_{gt})^2 \ / \ D_{gt}$.

RMSE: short for *root mean squared error*, formulated as $\sqrt{\frac{1}{N}\sum(D_{pred} - D_{gt})^2}$.

δ refers to the relative difference between D_{pred} and D_{gt}. This indicator can be formulated as $\delta = max(D_{pred}/D_{gt}, D_{gt}/D_{pred})$. So the value in table for columns of $\delta < 1.25$, $\delta < 1.25^2$ and $\delta < 1.25^3$ are actually the percentage of δ whose magnitude is below 1.25, 1.25^2 and 1.25^3 respectively.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] Amir Atapour-Abarghouei and Toby P Breckon. Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 18, page 1, 2018.

[3] Yuanzhouhan Cao, Zifeng Wu, and Chunhua Shen. Estimating depth from monocular images as classification using deep fully convolutional residual networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(11):3174–3182, 2018.

[4] Zhe Cao, Abhishek Kar, Christian Häne, and Jitendra Malik. Learning independent object motion from unlabelled stereoscopic videos. 2019.

[5] Zhe Cao, Abhishek Kar, Christian Hane, and Jitendra Malik. Learning independent object motion from unlabelled stereoscopic videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[6] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. *arXiv preprint arXiv:1811.06152*, 2018.

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

[8] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[9] Ravi Garg, Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016.

[10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

[11] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV)*, 2011.

[12] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2, page 7, 2017.

[13] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. Digging into self-supervised monocular depth estimation. *arXiv preprint arXiv:1806.01260*, 2018.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Joel Janai, Fatma G"uney, Anurag Ranjan, Michael J. Black, and Andreas Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *European Conference on Computer Vision (ECCV)*, volume Lecture Notes in Computer Science, vol 11220, pages 713–731. Springer, Cham, September 2018.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Arun CS Kumar, Suchendra M Bhandarkar, and P Mukta. Depthnet: A recurrent neural network architecture for monocular depth prediction. In *1st International Workshop on Deep Learning for Visual SLAM,(CVPR)*, volume 2, page 2, 2018.

[18] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016.

[19] Bo Li, Chunhua Shen, Yuchao Dai, Anton Van Den Hengel, and Mingyi He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1119–1127, 2015.

[20] Beyang Liu, Stephen Gould, and Daphne Koller. Single image depth estimation from predicted semantic labels. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1253–1260. IEEE, 2010.

[21] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[22] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5667–5675, 2018.

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

[24] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[25] Jogendra Nath Kundu, Phani Krishna Uppala, Anuj Pahuja, and R Venkatesh Babu. Adadepth: Unsupervised content congruent adaptation for depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2656–2665, 2018.

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

[27] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. Learning depth from single monocular images. In *Advances in neural information processing systems*, pages 1161–1168, 2006.

[28] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017.

[29] Yiran Wu, Sihao Ying, and Lianmin Zheng. Size-to-depth: A new perspective for single image depth estimation. *arXiv preprint arXiv:1801.04461*, 2018.

[30] Zhenheng Yang, Peng Wang, Wei Xu, Liang Zhao, and Ramakant Nevatia. Unsupervised learning of geometry with edge-aware depth-normal consistency. *arXiv preprint arXiv:1711.03665*, 2017.

[31] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1983–1992, 2018.

[32] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, volume 2, page 7, 2017.

[33] Daniel Zoran, Phillip Isola, Dilip Krishnan, and William T Freeman. Learning ordinal relationships for mid-level vision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 388–396, 2015.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Self-supervised Inference of Object Movement from Video

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Dai	Qi

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, 09. August, 2019	Q. Dai

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.