Deep Learning for Depth, Ego-Motion, Optical Flow Estimation, and Semantic Segmentation

By

Tuo Feng

A thesis submitted for the degree of

Doctor of Philosophy

School of Computer Science and Electronic Engineering

University of Essex

June 2021

Abstract

Visual Simultaneous Localization and Mapping (SLAM) is crucial for robot perception. Visual odometry (VO) is one of the essential components for SLAM, which can estimate the depth map of scenes and the ego-motion of a camera in unknown environments. Most previous work in this area uses geometry-based approaches. Recently, deep learning methods have opened a new door for this area. At present, most research under deep learning frameworks focuses on improving the accuracy of estimation results and reducing the dependence of enormous labelled training data.

This thesis presents the work for exploring the deep learning technologies to estimate different tasks, such as depth, ego-motion, optical flow, and semantic segmentation, under the VO framework. Firstly, a stacked generative adversarial network is proposed to estimate the depth and ego-motion. It consists of a stack of GAN layers, of which the lowest layer estimates the depth and ego-motion while the higher layers estimate the spatial features. It can also capture the temporal dynamics due to the use of a recurrent representation across the layers. Secondly, digging into the internal network structure design, a novel recurrent spatial-temporal network (RSTNet) is proposed to estimate depth and ego-motion and optical flow and dynamic objects. This network can extract and retain more spatial and temporal features. The dynamic objects are detected by using optical flow difference between full flow and rigid flow. Finally, a semantic segmentation network is proposed, producing semantic segmentation results together with depth and ego-motion estimation results.

All of the proposed contributions are tested and evaluated on open public datasets. The comparisons with other methods are provided. The results show that our proposed networks outperform the state-of-the-art methods of depth, ego-motion, and dynamic objects estimations.

Acknowledgment

I want to express my honest gratitude and appreciation for my supervisor, Professor Dongbing Gu, whose guidance, support, and encouragement have been invaluable throughout my Ph.D. study. His words and suggestions make people feel like a spring breeze, conveying warmth and hope, whether in study or life. At the beginning of my study, he guided me step by step into the latest academic frontiers in robotics and recommended Dr. Ruihao Li to help me with commencing paper: "Unsupervised Learning of Depth and Ego-Motion from Video". Thence, I am incredibly grateful for Dr. Ruihao Li, who shared his research papers and code when I just started. As my supervisor, Prof. Dongbing Gu continuously encouraged and was always willing and enthusiastic to assist in any way he could throughout my research. During the difficult period of the COVID-19, he still kept caring about my physical and mental health. I am very fortunate to have met and respected such excellent seniority who is a mentor and friend.

I would like to say a special thank you to my colleague Mr. Robin Dowlin. He supported my experiments, especially in GPU configuration for the deep learning development environment. For the very new hardware model in my office, he helped me configure the required software environment.

At last, I would like to thank my family. My parents raised me from childhood with immense love. I cannot study abroad to pursue my Ph.D. degree without their support and encouragement, with special thanks to my mother for supporting my later studies alone. In addition, I would like to say a big thank you to Miss Ji Qi from the bottom of my heart. She came to the UK to carry out Master's and Doctoral studies because of me. The cost of studying abroad at her own expense depends on her previous savings and a part-time job. Her concern and accompany let me go through this critical time. We encourage each other to make progress together, whether in the past, present, or future.

Contents

List of Figures

Notations

R	rotation
t	translation
t_x	x axis translation
t_y	y axis translation
t_z	z axis translation
α	x axis rotation angles
β	y axis rotation angles
γ	z axis rotation angles
$\mathbf{P}$	point P in 3D space
$\mathbf{a}$	vector a in 3D space
$\mathbf{a}^{'}$	vector $a^{'}$ through Euler transformation from vector a in 3D space
f	camera focal length
b	baseline of camera
xl	pixel point x-axis coordinate of the left camera
yl	pixel point y-axis coordinate of the left camera
xr	pixel point x-axis coordinate of the right camera
yr	pixel point y-axis coordinate of the right camera
pt	homogeneous coordinates of a pixel in the target view
ct	camera coordinates
pst	projected coordinates in the source view
$\hat{T}_{t\rightarrow s}$	projected target to source ego-motion
$\hat{D}(pt)$	projected target depth
$\hat{I}_s$	warped source image
I_s	original source image

I_t	original target image
Q	point Q in 3D space
C_1	source camera location
C_2	target camera location
$I_1^{'}$	source image
$I_2^{'}$	target image
$P_1^{'}$	point P passing position in source image
$P_2^{'}$	point P passing position in target image
$Q_1^{'}$	point Q passing position in source image
$Q_2^{'}$	point Q passing position in target image
L	deep network learned transformation relation from source to target
x_t	current input
h_t	previous states
$\mathbf{s}_i$	auto-encoder input
$\mathbf{s}_o$	auto-encoder output
$E()$	Encoder
$D()$	Decoder
$\mathbf{f}$	features in the middle of auto-encoder
E_i	right reconstruction loss function
$I_l^{'}$	synthesize left image
$I_r^{'}$	synthesize right image
E_p	3d point cloud warping loss
$\widehat{P_2}$	depth d2 corresponding point clouds
$P_{1\to 2}^{'}$	depth d2 transformed from depth d1 corresponding point clouds
A_l	error image in l layer
E^{l-1}	error unit
R^l	recurrent representation layer
G^l	generator
D^l	discriminator
S^l	initial states
d	depth estimator

p	ego-motion estimator
V	view reconstructor
$\hat{A}^l$	estimated error image
$\hat{I}^t$	predicted image
K	camera intrinsic matrix
$d_t(i,j)$	estimated disparity
$\hat{T}_{t-1,t}$	the camera coordinate transformation matrix
λ_l	weight at l layer
λ_t	temporal weight factor
n_l	total number of ReLU units
∇	solving gradient operation
x'	random interpolation samples
ε	a random number from uniform distribution between 0 and 1
M	using monocular image sequence
S	using stereo image sequence
B	batch size dimension
H	image height dimension
W	image width dimension
C	image channel dimension
$Image_{1,2,3}$	three monocular consecutive images
$Object_1$	Dynamic Object
L_p	photometric reprojection loss function
d_t^*	mean-normalized disparity
δ_x	image x axis gradient
δ_y	image y axis gradient
μ	auto-masking weight
μ_d	dynamic masking weight
F_d	depth features
F_i	image features
F_f	optical flow features

Abbreviations

SLAM	Simultaneous Localisation and Mapping
VSLAM	Visual Simultaneous Localisation and Mapping
VO	Visual Odometry
2D	Two Dimension
3D	Three Dimension
6-DoF	Six Degrees of Freedom
GPS	Global Positioning System
VINS	Vision-aided Inertial Navigation System
RCNN	Recurrent Convolutional Neural Network
PCA	Principal Component Analysis
DSO	Direct Sparse Odometry
PSPNet	Pyramid Scene Parsing Network
RSTNet	Recurrent Spatial-Temporal Network
SGANVO	Stacked Generative Adversarial Network Visual Odometry
MonoSLAM	Monocular Simultaneous Localisation and Mapping
RGB	Red Green Blue
RGB-D	Red Green Blue Depth
IMU	Inertial Measurement Unit
ORB	Oriented Fast and Rotated BRIEF
BoW	Bag-of-Words
PTAM	Parallel Tracking and Mapping
DTAM	Dense Tracking and Mapping
TSDF	Truncated Signed Distance Function
ICP	Iterative Closest Point

CPU	Central Processing Unit
GPU	Graphic Processing Unit
LSD-SLAM	Large-Scale Direct Monocular SLAM
DSO	Direct Sparse Odometry
SVO	Semi-Direct Monocular Visual Odometry
SP	Supervised
UnSP	Unsupervised
SemiSP	Semi-Supervised
DCNF	Deep Convolutional Neural Field
CRF	Conditional Random Field
CNN	Convolutional Neural Network
FCN	Fully Convolutional Network
SfM	Structure-from-Motion
RNN	Recurrent Neural Network
GAN	Generative Adversarial Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
ConvLSTM	Convolutional Long Short-Term Memory
ConvGRU	Convolutional Gated Recurrent Unit
SVM	Support Vector Machine
GANs	Generative Adversarial Nets
KL	Kullback-Leibler
JS	Jensen-Shannon
ReLU	Rectified Linear Unit
ATE	Absolute Trajectory Error
RMSE	Root-Mean-Square Error
SSIM	Structural Similarity Index Measure

Chapter 1

Introduction

The popularity of deep learning technology has led to tremendous research progress in computer vision and robotics. Visual-based SLAM is an overlapped area between these two fields, directly benefiting from applying deep neural networks to achieve numerous robotic tasks. In order to explore how this data-based learning method can achieve and improve various tasks of visual odometry in SLAM, this thesis has made several innovative attempts and contributions. This chapter introduces the research motivation, the challenging tasks, the used methodologies, and the novel contributions. Also included is the thesis's outline.

1.1 Motivation

Localization is such a problem that faces a robot that needs to figure out where it is on a given map using sensor data in robotics. Simultaneous Localization and Mapping (SLAM) is a complex problem. The robot needs to track where it is while building a map from scratch without prior knowledge introduced in [9]. This technology can endow mobile robots with persistent autonomy in an unknown environment and is widely used in autonomous driving and virtual reality applications. For the past decades, the autonomous navigation of a robot has depended on feeding a large number of sensors' data such as infrared, radar, gyroscope, GPS, and other superior sensors. The more precise the positioning, the more expensive sensor resources are needed. Robot perception has become a common practice in many industrial applications. With the continuous hardware development, computer vision provides an inevitable trade-off for navigation technology by only using images or videos. Designing a more reliable and accurate SLAM system using cheap sources like cameras has

become a competitive target among quite a lot of research works in robotics. Researchers named this SLAM system that only uses cameras as visual SLAM (VSLAM) in [10].

The classic VSLAM system generally consists of two parts: front-end feature extraction with loop closer detection and back-end mapping optimization as summarized in [11]. Because robots are hard to directly utilize the raw sensor data such as images from camera or beams from laser to express the SLAM state, the front-end as a preceding module extracts relevant pixel location features of different points from the view. The front-end module is named visual odometry (VO) in [12] which is in charge of data association through associating each sensor's measurement to a specific landmark. The back-end module optimizing visualization map relies on initial triangulating the positions of landmarks from multiple views. Therefore, the property of visual odometry directly affects the performance of SLAM. Following the discussion of [11], this thesis raises a further question of "how good the visual odometry problem is solved," which is also difficult to answer because there are many aspects in visual odometry that need to be specified, such as using monocular or stereo data, planar or three-dimensional environments, and single or multiple estimation tasks. Monocular and planar data leverages the cheapest resources and the most inadequate information. Despite the limited resources, previous visual odometry can quickly fail in some challenging environments, such as high-speed estimation and highly dynamic environments. These resource awareness conditions demand that researchers design more efficient and versatile visual odometry with decent accuracy and robustness. Thence, the motivation of this thesis is to develop some innovative monocular visual odometry systems to conduct various estimation tasks, which could be used for back-end processing.

1.2 Tasks and Challenges

Visual odometry primarily comprises four estimation tasks: depth (disparity), ego-motion, optical flow, and semantic segmentation. While these tasks are different, the challenges to face are almost identical, such as improving the estimation accuracy and robustness. This thesis will set out to accomplish the following tasks and address their faced challenges:

1. **Disparity Estimation**

   The depth of target objects in the view field from the camera is one of the most critical parameters in visual odometry. Civera [13] proposed a new parametrization for point

features: inverse depth within monocular SLAM which permits the efficient and accurate representation of uncertainty. This inverse depth which is the disparity mentioned below is an essential parameter that its accuracy directly affects the accuracy of the map and the scale recovery.

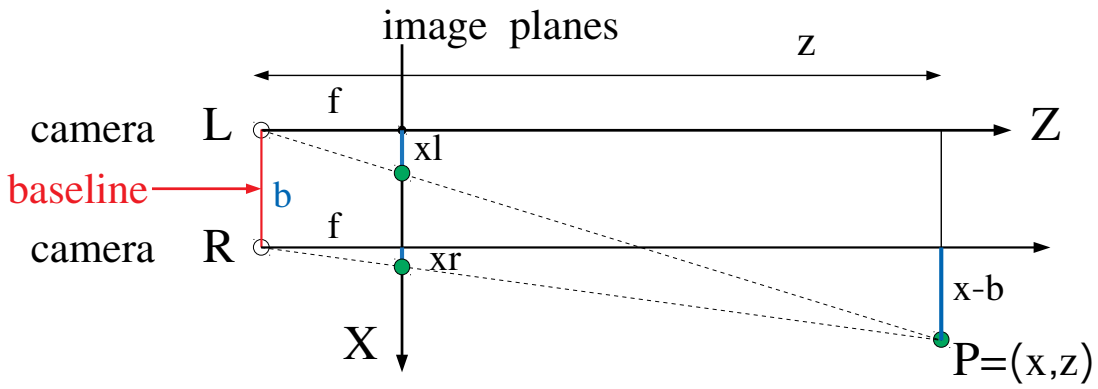

Figure 1.1: Ideal stereo camera imaging model to infer depth.

The challenge of disparity estimation is achieving accurate performance according to different hardware platforms such as monocular or stereo cameras. A stereo camera can quickly solve the depth estimation problems like an individual's eyes through knowledge of similar triangle geometry as shown in Figure 1.1. We can get three equations:

$$\frac{z}{f}=\frac{x}{xl}\ ;\quad \frac{z}{f}=\frac{x-b}{xr}\ ;\quad \frac{z}{f}=\frac{y}{yl}=\frac{y}{yr} \tag{1.1}$$

We can get z of point P by replacing x, which is the depth of the space point P from the camera:

$$z=\frac{f*b}{xl-xr} \tag{1.2}$$

Y-axis is perpendicular to the page. It can be found that if we want to calculate the depth, the premise assumes that we must know: 1. Camera focal length f, left and right camera baseline b. These parameters can be obtained through prior information or camera calibration. 2. Parallax $xl-xr$. Need to know the correspondence between each pixel (xl,yl) of the left camera and the corresponding point (xr,yr) of the right camera. The

above is the core problem of stereo vision.

Nevertheless, when people close one eye and observe an object with only one eye, they can also distinguish which object is close and which is distant. Does it mean that monocular cameras can also get depth information? The positive answer comes with many subsidiary conditions. It is authentic that people can obtain a certain depth of information through one eye. However, there are some easily neglected factors in action: people have an excellent prior knowledge of understanding this world. Therefore, there is a fundamental prediction of the size of everyday objects based on the years of visual training from the child. According to the common sense of objects, the individual's brain can indeed infer which object is far and which object is near in the view. The second factor is that the human's eye is shaking when observing an object which is equivalent to a moving monocular camera. Similar to the principle of the Structure from Motion (SfM), the single moving eye can get depth information by comparing multiple views. Thus, these thoughts inspire monocular depth estimation.

2. **Camera's Ego-Motion Estimation**

The most intuitive task of visual odometry is the camera's ego-motion estimation. Ego-motion is a continuous process where 2D image sequences captured by a camera are used to estimate 3D camera movement within a rigid scene as defined in [14]. The camera movement is a rigid body movement, which ensures that the length and angle of the same vector in each coordinate system will not change. This transformation becomes the Euclidean transformation. Ego-motion is 6-DoF camera rigid transformation quantity which consists of three dimensional Euler angles to establish rotation matrix R and three dimensional translation $t = [t_x, t_y, t_z]$. Rotation matrix R can be uniquely determined by the rotation axis x, y, z and the corresponding rotation angles: α, β, γ. For a three-dimensional point $P(x, y, z)$ rotating around the z axis by an angle θ, it can be expressed by the following rotation matrix:

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.3}$$

Rotation matrix rotating around the other two x, y axes as:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{1.4}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{1.5}$$

If the rotation axis sequence is (x, y, z), the rotation matrix is

$$R = R_x(\alpha)R_y(\beta)R_z(\gamma) \tag{1.6}$$

For a vector $\mathbf{a}$ in the world coordination, we can get $\mathbf{a}^{'}$ after one rotation described by R and one translation t. Then combining rotation and translation together are:

$$\mathbf{a}^{'} = R \cdot \mathbf{a} + t \tag{1.7}$$

Through the above formula, we use a rotation matrix R and a translation vector t to completely describe the coordinate transformation relationship of Euclidean space. VO or ego-motion is to estimate R, t under the same world coordinate.

Because VO conducts geometric computations with authentic images from 3D environments, ego-motion estimation faces low accuracy and robustness problems. As mentioned in the last section, a monocular camera estimates disparity through the structure from ego-motion in the rigid scene. Thus, eliminating dynamic objects in the real world and adapting to dynamic scene changes has become one of the significant challenges to removing the gap between the ideal rigid assumption and the real world. Ego-motion estimation couples with disparity estimation under the framework of visual odometry, which makes the simultaneous solution more difficult to be solved.

3. **Optical Flow Estimation**

Optical flow is the instantaneous velocity of pixel movement of 3D moving objects on the

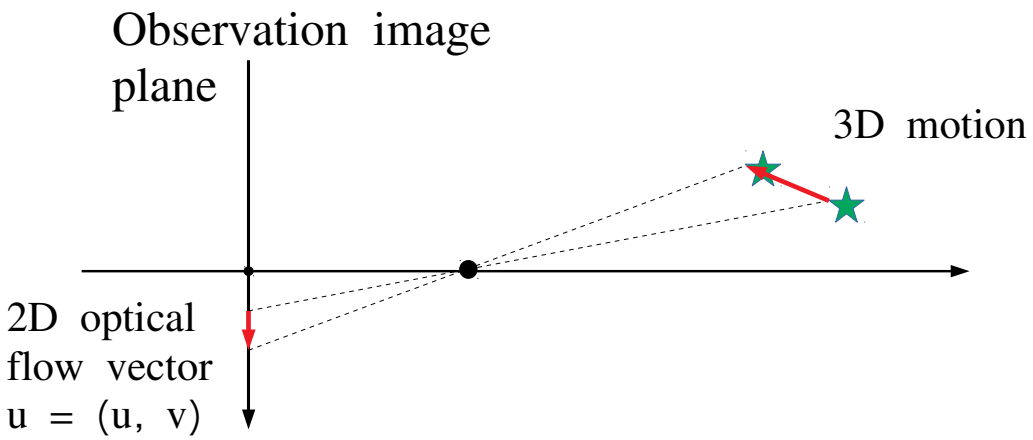

Figure 1.2: 3D motion is projected into optical flow in 2D image plane.

observation image plane as shown in Figure 1.2. Generally, the instantaneous change rate of a specific coordinate point between two images is defined as an optical flow vector. Optical flow comes into being by the movement of the foreground object itself, the movement of the camera, or the joint movement of them in the scene. In 3D space, the motion can be described by a motion field. However, the motion of an object is often reflected by the different grey-scale distributions of different images in the image sequence on the 2D image plane. Therefore, the motion field in space transferred to the image is expressed as optical flow vector $\mathbf{u}$. The optical flow vector is a two-dimensional vector: $\mathbf{u} = (u, v)$, which reflects the changing trend of the grey level of each point on the image, which can be regarded as the instantaneous velocity field generated by the motion of pixels with grey levels on the image plane. The information it contains is the instantaneous velocity vector information of each pixel. The purpose of studying the optical flow field is to approximate the motion field that cannot be directly obtained from the sequence of images. Therefore, the optical flow field corresponds to the motion field in an ideal situation.

If the optical flow method is used for visual odometry, there are two basic assumptions: brightness is constant and time is continuous-time, or exercise is small. When the same target moves between different frames, its brightness will not change significantly. This assumption of the primary optical flow method and all variants of optical flow methods must be satisfied. This assumption is used to obtain the fundamental equation of optical flow methods. Time changes should not cause drastic changes in the target position. The displacement between adjacent frames should be relatively small, an indispensable

assumption of the optical flow method. It is these basic assumptions that become the challenging issues to optical flow methods. In the next chapter, some details of optical flow methods will be discussed in part related to geometry-based visual odometry methods (literature reviews).

4. **Semantic Segmentation**

   Most SLAM solutions are working at a low level, i.e., feature point or pixel level. These feature points and pixels are chosen from computer vision algorithms or mathematical representations of image information. It is hard for individuals to identify feature points in some cases and judge their movement directions based on detected feature points. In contrast, individuals' vision is more efficient in identifying scenes at the object level and estimating the distance through left and right eyes to infer their ego-motion based on object movement in the view. This thesis envisions the higher-level representations including objects and solid shapes and believes semantic segmentation tasks will play a key role in visual odometry for the next generation of SLAM.

   The wide application of deep learning technology has improved the accuracy of semantic segmentation and replaced traditional methods such as support vector machines and conditional random fields. However, the accuracy improvement of segmentation needs developing deep neural networks to be trained on many labeled data. Designing a more efficient network to produce more accurate semantic segmentation is another challenge for visual odometry. In addition, utilizing semantic segmentation results to improve other estimation tasks, such as depth or ego-motion estimation tasks, is another problem worthy of investigation.

Hence, this thesis will set the above four tasks: disparity estimation, ego-motion estimation, optical flow estimation, and semantic segmentation as the challenging tasks to investigate using robust deep learning networks.

1.3 Methodology and Contributions

In recent years, deep learning methods have been proven to be more effective in processing robust and efficient prediction tasks. Researchers strive to transfer these deep networks to learn to estimate traditional physical quantities of visual odometry as a concise end-to-end

system. Analogously, deep learning networks can do ego-motion regression or disparity estimation by training on a large quantity of data set with ground truth according to recent years of work. This supervised deep learning method performs well on the same training data set. However, it does not perform well when migrating to a new scenario or other data sets with different parameter settings. This problem is the generalization ability of deep neural networks, and this is directly related to the robustness of the network's learning ability.

In order to solve this poor generalization problem of learning systems, researchers try to use an unsupervised learning methodology. The training process of unsupervised deep learning does not require the ground truth labels to enable the network to learn the ability to predict tasks. This unsupervised method dramatically increases the adaptive learning ability of networks and the prediction ability to unfamiliar scenes. Thus, the methodology of this thesis is proposing novel network solutions for the unresolved problems. By designing different network structures and new training loss functions, the thesis implements various estimation tasks based on the public training dataset. Afterward, evaluation experiments are processed on the public testing dataset to verify our novel contributions. The specific method uses unsupervised deep learning to estimate depth, ego-motion, and optical flow. Traditional geometric methods can constrain the learning convergence of these three fundamental quantities to design the loss functions for unsupervised training. In addition, semantic segmentation is a higher level of human-defined perception tasks. It does not have actual physical meaning, so supervised training is inevitable for semantic segmentation tasks in the thesis.

Learning-based VO networks have been proposed to jointly estimate the depth (task 1) and ego-motion (task 2). Because these works depend on the ideal rigid transformation assumption, one of the challenging problems that affect the robustness and accuracy of these deep networks is dynamic objects and occlusion areas in the scenes. The main work of this thesis is how the networks can learn the dynamic features and occlusion areas to improve the accuracy of learning-based visual odometry. This optimized estimation task of dynamic objects is achieved through unsupervised optical flow (task 3) estimation for dynamic objects detection. In addition, semantic segmentation (task 4) is integrated into the unsupervised visual odometry training process in order to create a novel deep learning pathway for high-level object-based representations of visual odometry.

The main contributions of this thesis are digging into the novel research of unsupervised

deep learning networks and dynamic noise processing of training approaches for the above four tasks in visual odometry which are summarised as below:

1. **Stacked Generative Adversarial Networks based Visual Odometry (Chapter 3)**
   Recently end-to-end unsupervised deep learning methods have achieved an effect beyond geometric methods for visual depth and ego-motion estimation tasks. These data-based learning methods perform more robustly and accurately in some of the challenging scenes. The encoder-decoder network has been widely used in depth estimation, and the RCNN has brought significant improvements in ego-motion estimation. Furthermore, the latest use of Generative Adversarial Network (GAN) in depth and ego-motion estimation has demonstrated that the estimation could be further improved by generating pictures in the game learning process. This section proposes a novel unsupervised network system for visual depth and ego-motion estimation: Stacked Generative Adversarial Network(SGANVO), consisting of a stack of GAN layers. The lowest layer estimates the depth and ego-motion, while the higher layers estimate the spatial features. It can also capture the temporal dynamic due to the use of a recurrent representation across the layers. This part is based on the following journal publication:

   - Tuo Feng, and Dongbing Gu. "SGANVO: Unsupervised deep visual odometry and depth estimation with stacked generative adversarial networks." IEEE Robotics and Automation Letters 4.4 (RA-L) (2019): 4431-4437.

2. **Recurrent Spatial-Temporal Networks for Estimating Depth, Ego-Motion, Optical Flow and Dynamic Objects (Chapter 4)**
   Depth and camera ego-motion estimations from raw monocular unlabeled consecutive RGB frames challenge learning-based visual odometry (VO). Most current monocular visual odometry learning works are based on warping monocular frames to neighbor views to process the geometric reconstruction. Considering the limitation of the monocular visual odometry learning task, the geometric reconstruction method fundamentally has some problems: occlusions and dynamic objects can bring about the artifacts error during the photometric loss's computation and break down the static environment assumption. In addition, construction cannot be inferred under a static scene with nil camera ego-motion. This thesis proposes a novel monocular visual odometry system with a recurrent

spatial-temporal feature learning network (RSTNet) to overcome these challenges. This contribution includes the design of a dynamic auto-masking loss to get rid of occlusions, nil ego-motion, and dynamic objects problems through warping source disparity to the target disparity based on the rigid flow field. This part of the work initially employs the error between full optical flow and rigid optical flow as the relatively camera static optical flow field for a dynamic object detection network to predict dynamic object masks. The evaluation experiment results outperform some state-of-the-art unsupervised monocular approaches. This part is based on the following publications:

- Tuo Feng, Dongbing Gu. "RSTNet: Recurrent Spatial-Temporal Networks for Estimating Depth, Ego-Motion, and Dynamic Objects." IEEE Transactions on Cybernetics. (T-Cyb), 2021. (Under Review)

3. **Coalesce semantic segmentation network and geometry-based method into our self-supervised learning of visual odometry (Chapter 5)**
   Semantic segmentation is a task of high-level processing objects in the field of computer version field. It can provide additional information of objects for depth estimation to assist in improving the estimation accuracy. Applying semantic segmentation to the training of depth estimation has become a popular research topic. Most of the previous work has focused on using pre-trained segmentation networks to predict semantic maps or ready-made segmented images as additional input to the depth estimation network. This method directly applies supervised learning with semantic labels in the two-stage training process. Namely, semantic segmentation and depth estimation are independently trained. In order to explore whether these two training stages can be combined into one training process to achieve the training convergence of two tasks simultaneously, this thesis proposes a new single-stage depth and ego-motion estimation coupled with the semantic segmentation task. It relies on a semantic edge-ware loss function.

   In addition, as geometric methods perform better than deep learning methods for ego-motion estimation, especially in the case of small rotations, this part also explores the development of a visual odometry system that can have both the accuracy of geometric methods and the robustness of deep learning methods. The system is a combination of our designed RSTNet and the commonly used geometric method. This part is based on

the following project:

- Tuo Feng, Dongbing Gu. "DeepSLAM Demonstration System Project." (University Project)

1.4 Outline of Thesis

This first chapter serves as an introduction to the full thesis. The second chapter will state the background literature review of my research. All of my work's contributions are distributed in Chapter 3, Chapter 4, and Chapter 5. The last one, Chapter 6 presents the conclusions and potential future work of this thesis. The details of each chapter are summarized as follows:

- Chapter 2 plays a role as the theoretical basis and source of this thesis through reviewing related work to the research's topic. The basic geometric visual odometry is initially introduced as a technical premise. There are three mainstream methods: feature-based methods, semi-direct methods, and direct methods. After that, learning-based visual odometry is reviewed as the core implementation reference. From previous supervised learning methods to recent unsupervised methods, this section presents the timeline of the related research progress. It introduces the basic models of networks for later implementation and the training data sets for the most common horizontal comparison standards. Afterward, the related work is reviewed in the following four parts according to the different tasks. Firstly, the literature review provides a comparative analysis of two disparity estimation methods: monocular and stereo. Then the published estimation networks of the camera's ego-motion and object's motion are discussed with sensor fusion learning networks. Related work of optical flow is divided into supervised methods and unsupervised methods. At last, high-level object segmentation work is listed, from semantic segmentation and instance segmentation to dynamic object detection.

- Chapter 3 proposes a stacked generative adversarial network (SGANVO) based unsupervised system for visual depth and ego-motion estimation. The related unsupervised learning works of depth and ego-motion estimation are introduced in the beginning. Then the system's overview is presented as a generator and discriminator architecture according to the structure of generative adversarial networks. The core novel contribution of

this visual odometry is the adversarial training procedure through the generator loss functions and discriminator loss functions. To evaluate the performance of our SGANVO, the depth estimation and ego-motion estimation experiments are demonstrated as evaluation methods. This chapter concludes that this system generates more accurate depth estimation results and comparable visual odometry results compared with other published unsupervised learning networks.

- Chapter 4 drives deep into the research of dynamic feature representation of deep learning networks used for visual odometry and proposes the recurrent spatial-temporal network (RSTNet) for estimating depth, ego-motion, and dynamic objects. The first part introduces dynamic objects and occlusion as two commonly faced problems of depth estimation and ego-motion estimation networks. The previous related work implemented the unsupervised learning-based visual odometry networks without considering the noise caused by dynamic objects to rigid transformation scenes. Furthermore, some improvement learning methods make a step forward in the network estimation performance by crudely eliminating dynamic masks. To achieve a refined dynamic object mask estimation, this chapter of the thesis proposes a novel network model consisting of recurrent spatial-temporal encoder-decoder modules. RST module can learn the perfect spatial-temporal features extraction through its internal recurrent features representation and refined sub-pixel layers. Because of proposing such a progressive network model, this thesis chapter presents the self-supervised monocular recurrent spatial-temporal visual odometry named RSTNet. RSTNet jointly manages four complex tasks: depth, ego-motion, optical flow and dynamic objects estimations in one stage training process of networks. The evaluation experiments provide sufficient verification for the performances of all involved tasks. The conclusion summarizes that our novel visual odometry can estimate more accurate depth, ego-motion, and optical flow with the advanced treatment of dynamic objects.

- Chapter 5 integrates the semantic segmentation supervised training into our novel unsupervised visual odometry training procedure. The high-level object segmentation allows the networks to learn to understand objects according to massive artificially defined labels, which costs an amount of hardware memory space. We use a semantic segmentation

network instead of an optical flow estimation network to implement the one-stage training target in our visual odometry system. The first part of this chapter is the introduction of related semantic estimation works. Afterward, depth estimation, ego-motion estimation, and semantic segmentation collaborative networks are proposed with a semantic-guided unsupervised training procedure. The evaluation experiments show that our novel networks estimate the accurate depth and ego-motion and produce competitive semantic segmentation results. This chapter also leverages geometry-based visual odometry and learning-based networks for engineering applications. The geometry-based visual odometry is integrated with parts of our trained network model to develop a visual odometry system by Python 3.6. Our synthesized visual odometry system's performance has less drift compared with some pure learning-based methods and more robustness than some pure geometry-based methods.

- Chapter 6 gives a conclusion for all the thesis and summarizes the contributions as academic papers for publications. Our novel ideas provide some new directions for future work in the direction of the SLAM research area.

This first chapter introduces my thesis, consisting of motivation, research tasks and challenges, methodology and contributions, and an outline of the whole thesis. The following chapter will present the thesis literature review.

Chapter 2

Literature Review

Before this thesis begins to present its main contribution work, some imperative background reviews need to be laid out. Section 2.1 introduces the related works of traditional geometry-based visual odometry as theoretical preparation for implementing the network training essence of learning-based methods. There are three kinds of basic geometry-based visual odometry: feature point methods, direct methods, and semi-direct methods. From the consistent timeline of research progress, the reviews of learning-based visual odometry are presented from supervised deep learning to unsupervised deep learning in Section 2.2. This section outlines the improvement of unsupervised training methods compared with previous supervised works and lists the most popular training data sets and basic deep learning network models transferred from the computer vision field. More details of related deep learning-based works to estimate disparity, ego-motion, optical flow, and object segmentation are divided into the following Sections 2.3, 2.4, 2.5, and 2.6.

2.1 Geometry-based Visual Odometry

This section provides fundamental principles and an overview of three classical geometric SLAM solutions, including feature points and direct and semi-direct methods. Feature point methods calculate the camera's ego-motion through the feature points extracted and matched from two consecutive frames and optimize the ego-motion through bundle adjustment to minimize the reprojection error like representative work in [15]. Because the feature point methods need to rely on a repetitive feature extractor and correct feature matching process to calculate the camera's motion correctly, it has apparent shortcomings: time-consuming

extractor limits the speed of SLAM, the sparse feature points waste valid information and downtime in case of featureless scenes such as a white wall or a vast sky. Therefore, a more holistic and more elegant way to deal with data association issues is proposed as the direct method. Just like the work of [16], the direct method estimates the motion of the camera based on minimizing photometric errors from brightness information of the pixels without knowing the correspondence between point pairs, which avoids the disadvantages of feature point methods. The optical flow method is one of the direct methods which is discussed in the task 2. Likewise, direct methods also strongly rely on constant grey values and small movements, limiting its application scenarios. Taking into account the characteristics of both feature point methods and direct methods, representative work in [17] puts forward the semi-direct methods, which contain selecting the feature points and computing the photometric errors of pixel block region surrounding the feature points. The related literature reviews of each method are detailed in the following three sub-sections.

2.1.1 Feature Point Methods

A large amount of work on the feature point method has been published each year. This section only outlines a basic timeline of research development. The timeline is not exhaustive, but it covers most typical works as shown in Table 2.1. It started with the initial real-time SLAM [18] in 2003 and its optimized version [19] in 2004. Meltzer *et al.* [20] developed a vision-based SLAM algorithm incorporating feature descriptors derived from multiple views of a scene, incorporating illumination and viewpoint variations. CV-SLAM in [21] proposed a fast and robust ceiling SLAM technique using a single ceiling vision sensor, and Smith extended the real-time monocular SLAM with straight lines in [22].

After summarizing these works, a representative MonoSLAM was emerged by Davison *et al.* [23] in 2007. MonoSLAM employed a probability framework to create sparse and consistent 3D feature points for 30Hz real-time mapping on a standard computer. However, its tracking and mapping components are synchronized in one linked thread. Because estimating the 6-DoF ego-motion and visualizing the 3D point cloud map on every image, the number of sparse futures of this single thread method is restricted. In the same year, Klein *et al.* [24] proposed a parallel tracking and mapping (PTAM) system solving the inefficiency problem by tracking each frame to compute the 6-DoF ego-motion but updating the map by

Table 2.1: Summary of Feature point methods

Methods	Year	Reference
Feature Points Method	2003	Real-time SLAM [18]
	2004	Real-time 3D SLAM [19]
	2004	Meltzer *et al.* [20]
	2005	CV-SLAM [21]
	2006	Smith *et al.* [22]
	2007	MonoSLAM [23]
	2007	PTAM [24]
	2009	Migliore *et al.* [25]
	2013	RGB-D SLAM *et al.* [26]
	2013	Li *et al.* [27]
	2014	Weikersdorfer *et al.* [28]
	2015	ORB-SLAM [15]
	2015	Leutenegger *et al.* [29]
	2016	Forster *et al.* [30]
	2017	ORB-SLAM2 [31]
	2017	Bundlefusion [32]
	2017	Mur *et al.* [33]
	2018	ProSLAM [34]
	2018	Sun *et al.* [35]
	2018	ICE-BA [36]
	2018	VINS-mono [37]
	2019	Geneva *et al.* [38]
	2020	Structure-SLAM [39]
	2020	DM-SLAM [40]

demanding bundle adjustment calculation on keyframes in the other thread. Nevertheless, PTAM system only performed very well in a small scene. In 2009, Migliore *et al.* [25] used a single camera to solve the problem of SLAM in dynamic environments obtaining.

Enhanced sensors can provide more helpful information for SLAM. Works in [26], [27], [28], [29], [30], and [37] take the advantage of enhanced sensors, such as RGB-D and inertial measurement unit (IMU). With the development of hardware, a new kind of sensor appears called event camera or Dynamic and Active-pixel Vision(DAVIS) sensor. Some SLAM algorithms like in [41] had a decent performance, especially in some challenging scenarios. Because it is not relevant to the methodology of this research, this section will not go deep into the methods with enhanced sensors.

ORB-SLAM [15] is considered as one of the most popular features points of monocular SLAM algorithms up to now, and the optimized version stereo ORB-SLAM2 [31] was proposed in 2017. The innovation point of this work is focused on the use of ORB features to recognize location through Bag-of-Words (BoW). ORB was proposed in [42] as a rotational invariant and scale-aware descriptor that can be extracted at high frequencies. The most outstanding advantage of ORB-SLAM is the efficient visual odometry with re-localization and loop closure detection as a complete real-time system. Following the PTAM's parallel threads in [24], but this ORB feature-based system can work well in large-scale environments. The most solid foundation work has been initiated from ORB-SLAM. The later works make different kinds of optimizations in [32], [33], [34], and [38] or transfer to indoor environments as [40] or add enhanced sensors like inertial information as in [39].

2.1.2 Direct Methods

Direct methods put data association and pose estimation in a unified nonlinear optimization framework. In contrast, feature point methods are solved step by step: the association among data is obtained by matching feature points, then estimates the ego-motion based on the association. These two steps are usually independent in feature point methods. However, direct methods directly estimate the movement of the camera based on the pixel brightness information, without calculating key points and using descriptors of feature points at all.

Silveira *et al.* [43] firstly proposed an efficient direct approach for visual SLAM directly using image intensities as observations which formulated the visual SLAM problem as a

Table 2.2: Summary of Direct methods and semi-direct methods

Methods	Year	Reference
Direct Method	2008	Silveira *et al.* [43]
	2011	DTAM [44]
	2011	Kinectfusion [45]
	2012	Kintinuous [46]
	2013	Whelan *et al.* [47]
	2013	Weikersdorfer *et al.* [48]
	2013	Kerl *et al.* [49]
	2013	Engel *et al.* [50]
	2014	LSD-SLAM [16]
	2015	Stereo LSD-SLAM [51]
	2015	Bloesch *et al.* [52]
	2016	ElasticFusion [53]
	2016	EVO [54]
	2017	DSO [55]
	2017	Stereo DSO [56]
	2018	DSVIO [57]
	2018	Zhou *et al.* [58]
	2019	BAD SLAM [59]
	2020	DVL-SLAM [60]
	2020	DSM [61]
Semi-Direct Method	2010	Newcombe *et al.* [62]
	2014	SVO [17]
	2016	SVO2 [63]
	2018	Lee *et al.* [64]
	2019	RESLAM [65]
	2019	Fmd Stereo SLAM [66]
	2020	SD-VIS [67]
	2020	Zhao *et al.* [68]
	2021	Liang *et al.* [69]

nonlinear image alignment task. Then Newcombe *et al.* [44] offered a dense tracking and mapping (DTAM) system. DTAM system estimates the depth of each pixel in an image to generate its dense 3D map. This approach inspires depth and ego-motion joint estimation in deep learning. Profit from using RGB-D cameras, KinectFusion was proposed in [45] as another dense registration and mapping method which is relied on a truncated signed distance function (TSDF) for pixel grid representation and utilizes Iterative Closest Point (ICP) for aligning depth images. Both DTAM and KinectFusion run indoors in real-time with commercial GPU. Whelan developed Kintinuous in [46] for optimizing the KinectFusion in the next year. The same author proposed a robust real-time visual odometry for dense RGB-D mapping. The similar attempts were made in [48] and [49] until semi-dense visual odometry [50] was proposed in 2014. This work increased the efficiency of dense-based methods without using all pixels of one image but using the pixels with a non-negligible image gradient to run in real-time on CPUs. SVO estimated the semi-dense disparity and tracked the 6-DoF ego-motion with the alignment of estimated disparity maps. Engel *et al.* [16] extended SVO to Large-Scale Direct Monocular SLAM (LSD-SLAM) system which could run in large-scale environments with CPUs. The optimization of LSD-SLAM is employing the $\mathfrak{sim}(3)$ to detect scale drifts and providing a probabilistic solution to improve the depth prediction during tracking. The stereo version of LSD-SLAM in [51] was developed in the next year. After the works in [52], [53] and [54] tried different explorations, Engel proposed another very representative work: direct sparse odometry (DSO) in [55]. This work combined photometric errors with geometric disparity errors. It optimized all the model parameters jointly to obtain good performances such as high accuracy in tracking and mapping and robustness under featureless challenges. The stereo version and inertial version of DSO were proposed following that in [56] and [57]. Zhou *et al.* [58] presented the semi-dense 3D reconstruction based on stereo event camera. T.Schops in [59] applied the bundle adjustment to process direct SLAM based on RGB-D cameras. DVL-SLAM [60] and DSM [61] utilized the sparse information to implement their optimizations for direct methods.

2.1.3 Semi-Direct Methods

Semi-direct methods establish feature correspondences based on direct methods. This method constructs the tracking map by minimizing the photometric errors and reprojection errors. In

the second part of Table 2.2, there are not so many works like the other two methods. Newcombe *et al.* [62] made the first attempt to develop a monocular semi-direct method. Forster proposed a fast semi-direct monocular visual odometry (SVO) in [17] and the second version of SVO was proposed by the work [63]. Lee *et al.* [64] improved the semi-direct based monocular SLAM in a loosely-coupled way. Focus on the fast performance of semi-direct method, Tang *et al.* [66] and Liu *et al.* [67] explored novel SLAM systems. For robustness of SLAM in complex environments, Zhao *et al.* [68] proposed a feature-aided semi-direct stereo SLAM and Liang *et al.* [69] proposed a SLAM algorithm in challenge environments. Although semi-direct methods have some advantages of both feature point methods and direct methods, they have a high requirement on image quality and are sensitive to photometric changes. This is similar to pure direct methods.

To sum up, these three approaches have achieved good performance in some targeted scenes. However, their robustness and accuracy in complex scenes such as high dynamic and large-scale environments still need further improvement. Learning-based visual odometry is proposed as the new emerging pathway to solving the above problems. Nevertheless, the basic ideas of these three geometry-based methods provide the primary theoretical foundation for learning-based visual odometry.

2.2 Learning-based Visual Odometry

Learning-based visual odometry estimates the required tasks in an end-to-end mode without matching manually designed features between images. Deep neural networks extract implicit features inside the multi-layer of network structure. The expression layer of networks can export the unexplained feature points, disparity, optical flow, or ego-motion estimation for the back-end of SLAM.

At the beginning of this chapter, the thesis presents a wraparound summary of related works for supervised, unsupervised, and semi-supervised methods. The basic deep learning network models, popular loss functions, and training data sets are introduced in the following sub-sections of the first part. The next four sections introduce the main four tasks: disparity, ego-motion, optical flow, and high-level object segmentation.

Table 2.3: Learning-based visual odometry literature summary. SP represents supervised learning methods, UnSP represents unsupervised learning methods, and SemiSP represents semi-supervised learning methods.

Reference	Year	SP	UnSP	SemiSP	Task
Eigen *et al.* [70]	2014	√			Depth
Li *et al.* [71]	2015	√			Depth, Surface Normal
Liu *et al.* [72]	2015	√			Depth
Konda *et al.* [73]	2015	√			Pose
Kendall *et al.* [74]	2015	√			Pose
Costante *et al.* [75]	2015	√			Pose
Mayer *et al.* [76]	2016	√			Depth, Flow
Kendall *et al.* [77]	2017	√			Depth
Clark *et al.* [78]	2017	√			Depth, Pose
Wang *et al.* [79]	2017	√			Pose
CNN-SLAM [80]	2017	√			Depth, Pose
Fu *et al.* [81]	2018	√			Depth
Xue *et al.* [82]	2018	√			Pose
Xue *et al.* [83]	2019	√			Pose
Chen *et al.* [84]	2019	√			Pose
Facil *et al.* [85]	2019	√			Pose
Garg *et al.* [86]	2016			√	Depth
Godard *et al.* [1]	2017			√	Depth
Kuznietsov *et al.* [87]	2017			√	Depth
Poggi *et al.* [88]	2018			√	Depth
Ramirez *et al.* [8]	2018			√	Depth
Aleotti *et al.* [89]	2018			√	Depth
Pilzer *et al.* [90]	2018			√	Depth
Wang *et al.* [91]	2018			√	Depth, Pose
Zhan *et al.* [92]	2018			√	Depth, Pose
Li *et al.* [93]	2018			√	Depth, Pose
Wang *et al.* [94]	2019			√	Depth, Pose
Pilzer *et al.* [95]	2019			√	Depth
Tosi *et al.* [96]	2019			√	Depth
Chen *et al.* [97]	2019			√	Depth
Fei *et al.* [98]	2019			√	Depth
Zhou *et al.* [4]	2017		√		Depth, Pose
Vijayanarasimhan *et al.* [99]	2017		√		Depth, Pose
Yang *et al.* [100]	2017		√		Depth, Pose
Mahjourian *et al.* [101]	2018		√		Depth, Pose
Zou *et al.* [102]	2018		√		Depth, Pose, Flow
Yin *et al.* [103]	2018		√		Depth, Pose, Flow
Ranjan *et al.* [104]	2019		√		Depth, Pose, Flow
Wang *et al.* [105]	2019		√		Depth, Pose
Li *et al.* [106]	2019		√		Depth, Pose

2.2.1 Deep Learning Methods Summary

Learning-based visual odometry handles the required tasks through training deep neural networks. The networks are trained periodically through their designed objective function until it converges. According to the resources required by different objective functions, deep learning network training methods are divided into supervised, unsupervised, and semi-supervised. This training procedure is supervised if the objective function needs the ground truth labels of required task data to build the loss functions. Conversely, the needless ground truth data method is unsupervised. A semi-supervised approach between them does not need the task's ground truth data but utilizes ground truth labels for other variables. In this subsection, the thesis provides a summary of deep learning-based visual odometry works that are divided into three categories: supervised, unsupervised and semi-supervised in Table 2.3. This table also summarizes the main tasks of each work. Because the main contributions of this thesis are based on unsupervised learning methods, and related works of each task are discussed in the following sections, the unsupervised learning methods of this part are not exhaustive. In the Table 2.3 SP, UnSP, and SemiSP represent supervised unsupervised and semi-supervised three learning methods. The disparity is the inverse of depth tasks, and the pose represents the camera's ego-motion for simplicity.

Eigen *et al.* [70] made the initial attempt to predict the depth map from monocular data through training a multi-scale deep network. Li *et al.* [71] added the surface normal into the network estimation tasks. Liu *et al.* [72] presented another depth estimation method with single images using the deep convolutional neural field (DCNF) which integrates continuous Conditional Random Field (CRF) components into an unified deep convolution neural network (CNN). To improve real-time performance, Li *et al.* [71] proposed the super pixel pooling method combined with fully convolutional networks (FCN). The most intuitive idea of these supervised learning works from [73] to [79] and [82] to [85] is directly applying the ego-motion ground truth as the supervised training signal to train the networks to regress the 6-DoF ego-motions according to input images sequence. Works in [77] and [81] changed the supervised task to depth estimation. Jointly estimating the multi-tasks in supervised learning methods was introduced in [71], [76], [78], and [80].

Semi-supervised learning networks are almost trained for a single task such as depth

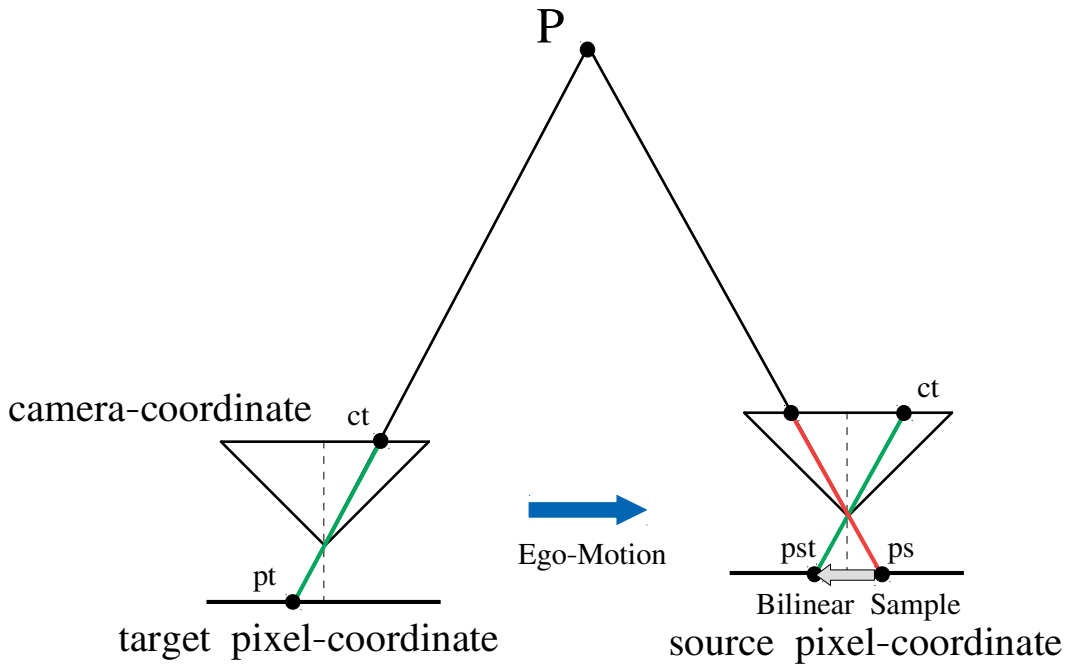

Figure 2.1: Target view synthesize through warping source view.

or ego-motion. Suppose it needs to estimate disparity by using a monocular camera. In that case, the camera's ego-motion should be known in advance, as discussed in the third section methodology of Chapter 1. Geometry stereo depth estimation applies the knowledge of similar triangle geometry as shown in Figure 1.1 which can be regarded as two sequential monocular images on a time series with the known motion. Therefore, the semi-supervised learning for monocular depth estimation is based on the ego-motion supervised estimation in [86] to [90] and [95] to [98]. For stereo depth estimation in [91] to [94], the deep networks can be trained to jointly estimate depth and ego-motion using unsupervised learning with stereo training input images.

2.2.2 Unsupervised Learning Implementation

Supervised learning is very straightforward. Nevertheless, the most critical problem arises: estimating the ego-motion or depth through unsupervised training only by feeding monocular image sequences. To illustrate this problem, it is necessary to start with the view synthesis through image warping based on rigid transformation. Let *pt* denote the homogeneous coordinates in the target view, and *K* denote the camera intrinsic matrix. We can obtain the *pt*'s

projected coordinates *pst* onto the source view *ps* as bellow:

$$pst \sim K\hat{T}_{t\to s}\hat{D}(pt)K^{-1}pt \tag{2.1}$$

The projected coordinates *pst* are continuous values. In Figure 2.1, P is a point in the 3D space, *pt* is the homogeneous coordinates on the target image frame. The camera coordinates *ct* can be converted by pixel coordinates *pt* by using the depth $\hat{D}(pt)$ of pixel *pt* and camera intrinsic matrix K. The target camera coordinate *ct* can be projected to the source pixel coordinate *pst* through the camera intrinsic matrix multiplying ego-motion $\hat{T}_{t\to s}$ in equation 2.1. By using the bilinear sample technique, we can obtain the value of the warped image $\hat{I}_s$ at location *pt* from $I_s(ps)$: $\hat{I}_s(pt) = I_s(ps)$. The composite target image $\hat{I}_s(pt)$ warping from source view $I_s(ps)$ constructs the supervised signal with the input image $I_t(pt)$ for training although the synthesized image $\hat{I}_s(pt)$ could lose some edge pixels sometimes. In this way, the network can jointly estimate the depth and ego-motion without ground truth labels to implement unsupervised learning.

2.2.3 Deep Neural Network Models

Supervised and unsupervised learning methods need to build deep neural networks to represent the quantities required by various tasks. The most basic neural network model is the convolutional neural network (CNN). Albawi *et al.* [107] presented this CNN name coming from mathematical linear convolution operation between matrixes. A CNN has a structure with multiple layers: including convolutional layer, non-linearity layer, pooling layer, and fully-connected layer. The convolutional and fully connected layers have parameters, but pooling and non-linearity layers do not have parameters.

There are three crucial network models to be used. They are recurrent neural network (RNN), Encode-to-Decoder, and generative adversarial network (GAN).RNN is a class of artificial neural networks where the connections between nodes form a directed graph along a temporal sequence as introduced in [108]. Its structure allows it to exhibit temporal dynamic behavior through maintaining the memory of hidden states in terms of time via feedback loops as shown in Figure 2.2. RNN model runs depending on the current input x_t and previous states h_t. A simple RNN cannot handle the exponential exploding of weights or disappearance of gradients with recursion, and it is difficult to capture long-term correlation.

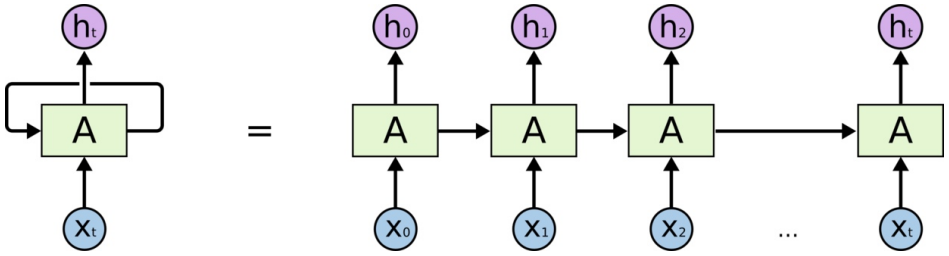

Figure 2.2: Basic RNN unit expands in time series.

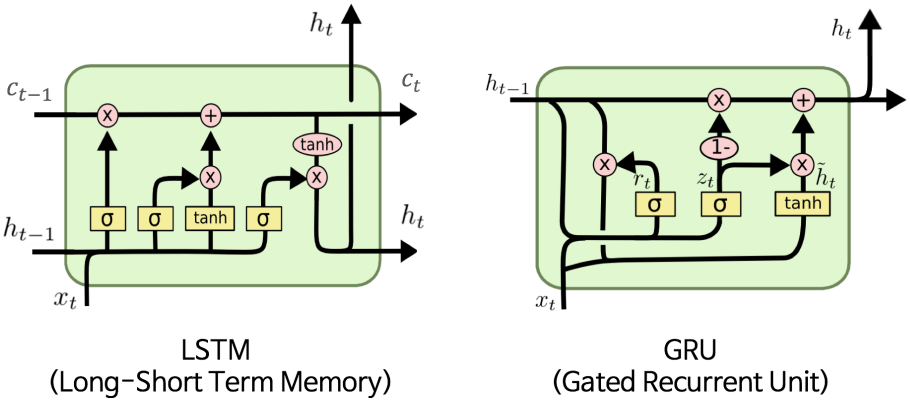

Figure 2.3: Internal structures of LSTM unit and GRU unit

Therefore, a unique recurrent structure is long short-term memory (LSTM) which can solve the problem well. LSTM is proposed in [109] which is suitable for processing and predicting important events with very long intervals and delays in time series. However, the evident disadvantages of LSTM are the complex network structure and huge storage memory of network parameters. Because LSTM training is relatively slow, gated recurrent unit (GRU) is introduced by [110] slightly modifying LSTM to make speed much faster, but the accuracy is unchanged. The internal structures of the LSTM unit and GRU unit are shown in Figure 2.3. To construct a spatial-temporal sequence prediction model and grasp time-space information at the same time, the fully connected weight in LSTM is changed to convolution, which is called ConvLSTM in [111]. The same spatial-temporal improved GRU network is proposed as ConvGRU in [112]. More technical details such as the mathematical relationship of the internal unit structure have been clearly stated in the related papers. The recurrent neural network unit used by Chapter 4 is the ConvGRU model.

Auto-encoder is another essential model of deep learning networks, as shown in Figure 2.4. The purpose of this model is the learning presentation for a group of data, especially for dimensionality step-down like compressed sensing [113]. It is different from compressed

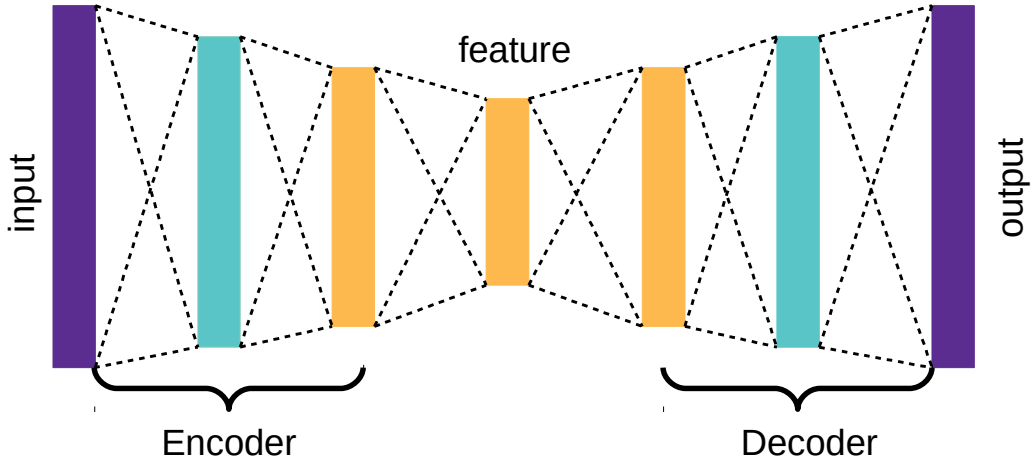

Figure 2.4: Auto-encoder model

sensing, where the signal is sampled internally and recovers the signal. It works by compressing the input signal through CNN to learn the feature representation and recovering it through mirror-symmetric CNN from features with reduced distortion. Compressed sensing has the advantage of a fast recovering signal by little sampling information, but encoder-decoder has the advantage of less distortion encoding like work in [114]. There are two primary components of the auto-encoder model: encoder and decoder. The encoder and decoder are connected to form a feed forwarding mesh structure with the coding features generated in the middle layers. This encoder-to-decoder mathematical logic can be expressed in the following formula:

$$\mathbf{s}_o = D(\mathbf{f}); \quad \mathbf{f} = E(\mathbf{s}_i) \tag{2.2}$$

which $\mathbf{s}_i$ is the input signal feeding into the encoder E that is a non-linear function representation. The encoder compresses the input signal through CNN to export coding features $\mathbf{f}$. The decoder D is also a non-linear function processing $\mathbf{f}$ through deconvolution layers, dilated convolution layers, upsampling layers, or convolution layers to output $\mathbf{s}_o$. Auto-encoder has been widely used for image classification and image recognition in the computer vision field. The most popular network models used in depth estimation, optical flow estimation, and segmentation are the encoder-decoder models. Some details will be introduced in the following sub-sections, along with related tasks.

Generative adversarial network (GAN) is a class of machine learning frameworks de-

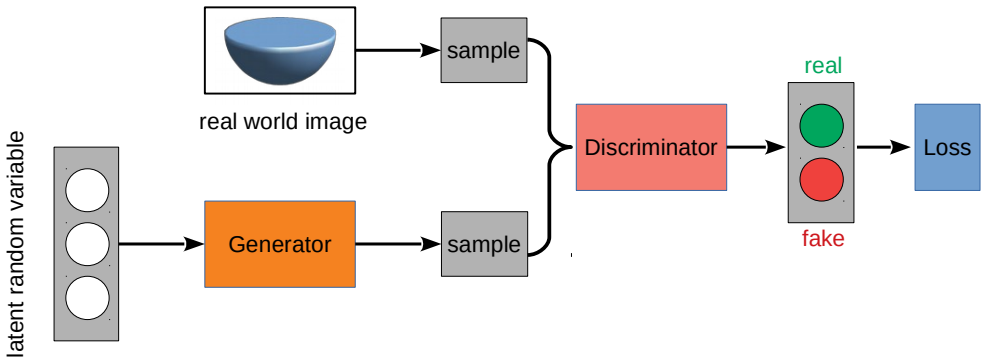

Figure 2.5: Generative adversarial network model

signed by Ian Goodfellow *et al.* [115] in 2014. The basic model structure is two neural networks contesting in a zero-sum game where one agent's gain is another agent's loss. The model of GAN is shown in Figure 2.5. Generator and Discriminator are two neural networks acting as the players of the game. The generator network generates learning samples from latent random variables. The input to the Discriminator network is an actual image sample with the generated sample. Discriminator learns to detect if the generated sample is real or fake according to the actual image sample. The generator loss and Discriminator loss are formed from the zero-sum game rule. The loss functions generally consist of multiple loss functions such as the photometric errors and reprojection errors of geometry-based methods for GAN-based visual odometry. More details of implementation are presented in Chapter 3.

Speaking of the typical network model, it is inevitable to introduce the Spatial Transformer Network [116]. For computer vision tasks, we hope that the model can have a certain degree of invariance to the change of object pose or position to analyze objects in different scenes. Traditional CNN uses convolution and Pooling operations to achieve translation invariance to a certain extent. However, this artificially set transformation rule makes the network excessively dependent on prior knowledge, which can neither truly achieve translation invariance. Invariance's requirements are very high for translation which makes CNN lack the proper feature invariance for geometric transformations such as rotation and distortion that have not been artificially set. Spatial Transformer Network with derivable properties does not require extra annotations and can adaptively learn the spatial transformation methods for different data. It can perform the spatial transformation on the input and be inserted

into any layer of the existing network as a network module to achieve spatial transformation of different Feature maps. Finally, let the network model learn the invariance to translation, scale transformation, rotation, and more common distortions, and also make the model also shows better results on many benchmark data sets.

2.2.4 Tackle the Long-Range Dependencies of CNN

From the effect point of view, the spatial invariance can realize the degeneration, such as parameter sharing and translation. The channel specificity allows the convolution kernel to collect diversified information encoded in different channels. However, it should be noted that convolution is not perfect, and there are some inherent defects. On the one hand, although the combination of space invariance and space compactness plays a role in improving efficiency and translation and other degeneration, it causes the convolution kernel to lose the ability to adapt to the diversified visual modes of different spatial positions. The locality also limits the receptive field of convolution, and it is challenging to capture long-distance spatial interactions at one time. On the other hand, even for many successful deep neural networks, there is apparent inter-channel redundancy in the convolution filter, making it possible for us to reduce the specificity of the convolution kernel in different channels without significantly affecting its expression ability.

The design of the convolutional layer needs to ensure locality through a limited receptive field and ensure translation equivariance through weight sharing. Research [116] has shown that these two attributes are the critical inductive biases when designing image processing models. However, the inherent locality of the convolution kernel makes it impossible to obtain the global context in the image, and to recognize the objects in the image better, the global context is essential.

The self-attention mechanism [117] is a recent development in obtaining long-range interactivity, but it is mainly used in sequence modeling and generative modeling tasks. The key idea behind the self-attention mechanism is to take the weighted average of the values calculated by the remote unit. Unlike pooling or convolution operators, the weight used in the weighted average operation is dynamically obtained through the similarity function between the hidden units. Therefore, the interaction between input signals depends on the signals themselves rather than predetermined by their relative positions. In particular, it is

worth mentioning that this enables the self-attention mechanism to obtain the Great Wall interactivity without increasing the parameters. In the future, our RSTNet in chapter 4 can utilize this idea to optimize our network to obtain longer-range learning capabilities.

2.2.5 Training Dataset

The learning mechanism of deep neural networks is very much like the human learning process, which needs a large amount of data for training. Individuals understand this world by seeing a lot, hearing a lot, and feeling a lot to establish reactions. This data-driven learning mode makes deep neural networks simulate the individual perception of the environment. This section will briefly overview popular datasets used for depth estimation, ego-motion estimation, optical flow estimation, and object segmentation.

The first dataset KITTI is one of the most leading comparison benchmarks, which is collected in outdoor environments in a driving vehicle in [118]. The KITTI provides monocular, and stereo images with ground truth of 6-DoF ego-motions derived from the fusion of multiple sensors data and the depth maps derived from the calibrated laser. The KITTI also provides a small number of artificial labels of semantic segmentation and instance segmentation. The second popular dataset is Cityscapes [119] which also provides monocular and stereo image sequences with ground truth of depth map, ego-motions, and semantic segmentation. Almost all the work of depth estimation and ego-motion estimation are evaluated on these two benchmarks. For object segmentation tasks, COCO [120] is the most frequently used data set. To remedy the lack of abundant segmentation training labels and 3D labels, the KITTI 360 [121] is proposed for the joint estimation of depth, ego-motion, and semantic segmentation.

Considering the fairness of comparison with other works, KITTI [118] is the primary data set used in this thesis for depth estimation, ego-motion estimation, optical flow estimation, and semantic segmentation.

2.3 Disparity Estimation with Deep Learning

Disparity estimation is a crucial task for visual odometry. Some geometry-based methods utilize the disparity to implement the ego-motion computation. Following the summary of deep learning methods in Chapter 2, this section introduces related unsupervised learning

work of disparity estimation. By focusing on only using cameras as input without any other additional sensors, unsupervised deep learning methods of disparity estimation are divided into monocular depth estimation and stereo depth estimation.

2.3.1 Monocular Depth Estimation

How to implement unsupervised learning for depth estimation has already been described in earlier sections. The primary method is to design two networks separately, estimating the depth map of each frame and the ego-motion between two consecutive frames. The learning cost function consists of reprojection errors and photometric errors through warping to synthesize target frame from a source frame in the SfM principle which jointly constrains the learning convergence of the two networks.

The groundbreaking works are represented by SfMlearner [4], and SfM-net [99]. Both of them proposed the auto-encoder model-based disparity network for predicting depth and pose network for predicting ego-motion. SfMlearner proposed an additional decoder network to predict the uncertainty mask for eliminating non-rigid motion noise. SfM-net [99] proposed using two auto-encoder networks to predict depth and object masks. In addition, SfM-net applied the features from the middle layer of a motion network connecting with the object's motion and camera's motion representation layers to predict ego-motion. Similarly, Geonet [103] improved the estimation performance of depth network and pose network and constructed the residual optical flow network to refine optical flow estimation. This work divided the network's training into two stages: first, train the SfM networks to obtain good initial estimations and then train the residual optical flow learning network to estimate full flow. Yang *et al.* [100] added the edge-aware depth normal consistency loss in the cost function. Furthermore, Mahjourian *et al.* [101] was inspired by the optimization method: Iterative Closest Point (ICP) to build the 3D geometric constraint loss function.

2.3.2 Stereo Depth Estimation

Unsupervised stereo depth estimation is an improvement pathway of pure monocular depth estimation without other expensive sensors. Inspired by the Deep3D [122], the deep neural network can learn the depth representation like an individual's stereo eyes through training a large number of stereo images. The change from the left to the right is viewed as a camera motion and used as additional depth training information. The main idea runs as the

representation capability of auto-encoders, where the encoder represents the left image with a predicted monocular depth map. At the same time, the decoder is a warp function that synthesizes a reconstructed image through the predicted depth map from the right image.

This left-right continuity work was proposed in [86] which reconstructed the error of left image from right warped image to build the warping loss functions to train the networks. Zhong *et al.* [123] developed the unsupervised deep depth estimation network by feeding pure stereo images in this way. Godard *et al.* [1] improved the stereo depth estimation by wrapping left and right images I_l and I_r across each other to synthesize the corresponding images $I_l^{'}$, $I_r^{'}$ for left and right reconstruction loss function E_i:

$$E_i = \sum \left\| I_l - I_l^{'} \right\|_2 + \sum \left\| I_r - I_r^{'} \right\|_2 \tag{2.3}$$

The accuracy of depth prediction could be enhanced by penalizing both left and right photometric losses. UnDeepVO [93] enriched the reconstruction loss functions by warping temporal sequence images forward and backward each other and warping left and right each other. Thus, this work achieved better performance on depth and ego-motion estimations with the vast loss parameters. In addition, the use of 3D geometric registration loss only contributed a little or even adverse effects through this thesis's research experiments in some previous works. Why this kind of loss function performs poorly other than making a significant difference like work in [101]. Because the depth estimation is not accurate in the early stage of the training, the 3D point cloud in the camera coordination, which is transformed from using the depth and image, is not reliable. Accordingly, the joint training of 2D reconstruction error and 3D geometric registration loss cannot guarantee the network to converge to the target tasks. The 3D point cloud warping loss E_p can be listed as below:

$$E_p = \sum \left\| \widehat{P}_2 - P_{1\to 2}^{'} \right\|_L \tag{2.4}$$

The point cloud $\widehat{P}_2$ is transformed from the depth d_2, which is poorly predicted by the depth network at the beginning of the training process. The point cloud $P_{1\to 2}^{'}$ is transformed from the depth d_1. The poor depth estimation cannot provide a valued supervised signal for training. In other words, the quantity to be predicted cannot self-guide training's convergence. Differently, [101] constructed the points cloud through the depth and image, which

are strongly dependent on the image projected onto 3D space according to the depth information. The image pixel loss provides more training weight in this way instead of completely depending on the roughly estimated depth. To sum up, stereo depth estimation networks can be trained fully unsupervised in an end-to-end approach. Some of the related work can be treated as improved monocular depth approaches during the inference phase with the pose estimation networks. Benefit from geometrical constraints. Recent unsupervised networks even outperform some supervised methods in terms of accuracy of depth estimation.

2.3.3 Depth Estimation Discussion

Disparity or depth estimation plays an essential role in building up dense maps for SLAM systems. The related depth estimation works have made good estimation performance based on monocular camera or stereo camera. Depth estimation methods still face some problems from their theoretical underpinning assumptions: rigid transformation and static scene. More and more works are being proposed to overcome these problems such as modifying loss functions and predicting more subsidiary tasks. Monocular depth estimation networks can learn disparity representation from monocular image sequences, which could save more resources than stereo networks.

No matter how these works optimize the depth estimation networks, they cannot produce a good result without ego-motion estimation under an unsupervised learning framework. In the next section, deep learning-based motion estimation works will be reviewed.

2.4 Ego-Motion Estimation with Deep Learning

Traditional geometry-based visual odometry utilizes rigid transformation technical knowledge to estimate the camera's ego-motion. Deep neural networks can be trained to learn sensing distance and position and use this feedback to adjust the pose in an end-to-end data-driven method. Thus, the depth and the ego-motion are jointly estimated by networks in the unsupervised learning mode. In addition, object motion estimation opens up a pathway to deal with dynamic object noise.

2.4.1 Improved Camera's Ego-Motion Estimation

The prime task of visual odometry is estimating the camera's ego-motion to build a trajectory map. Deep learning networks learn to predict 6-DoF ego-motion as a regression problem in

supervised learning methods such as Kendall *et al.* [74] and [124]. In order to implement unsupervised learning of ego-motion, depth and pose are jointly estimated as introduced in the initial work [4] and [99]. The basic theoretical explanation of how ego-motion unsupervised learning is the same as the depth's unsupervised deep learning in the second sub-section of Chapter 2.2. The required quantities: depth and ego-motion, are estimated by unsupervised deep learning networks under the rigid transformation and static scenario assumption. How to deal with the limitations exposed by the assumption becomes a major challenging task.

Li *et al.* [125] extends his UnDeepVO [93] with uncertainty mask output and loop closure detection to refine the ego-motion estimation where the dynamic mask work is of an identical nature with [4]. This estimated mask by the network during the unsupervised training is not very precise for filtering out the noise caused by dynamic objects in the input image. Godard *et al.* [6] proposed a representative work for dynamic auto-masking method from computing photometric and reprojection loss functions. This optimization treatment masks the large areas of low feature points like the sky and large objects that are relatively stationary. More technical details will be discussed in the related work in Chapter 4. Ranjan *et al.* [104] and Wang *et al.* [105] proposed the additional networks for estimating the optical flow to compute the dynamic mask from the difference between full flow and rigid flow. The flow error mask can mask more distinct dynamic and uncertainty areas during ego-motion unsupervised learning. To express dynamic objects more explicitly, Casser *et al.* [126] utilized additional semantic segmentation input for predicting dynamic segmentation to refine the depth and ego-motion joint unsupervised learning. Gordon *et al.* [127] reduced the harsh requirements of semantic segmentation to object masks to estimate every pixel's motion of dynamic object region. The camera's ego-motion and object motion jointly participate in calculating loss functions based on the rigid transformation to improve the performance of the network's ego-motion estimation. Li *et al.* [128] improved this method by predicting each pixel's 3D motion without any semantic segmentation or object masks. Bian *et al.* [5] proposed a novel mask method by using the view synthesis warping depth map to reconstruct the error with the original estimated depth map.

Unlike the explicit or implicit dynamic masks, amending the network's structure to improve the network's learning ability is another direction for improving ego-motion estimation. Li *et al.* [93] proposed the RNN based pose expression layers in the network to learn the

temporal feature representation for ego-motion estimation. GANVO [129] initially proposed combining the GAN to predict ego-motion in the game setting. SGANVO [130] proposed the stacked generative adversarial networks with the RNN ego-motion expression layers in the stacked structure networks. Profit from more temporal and spatial features to be learned. These methods push the ego-motion towards a higher level of performance.

2.4.2 Object Motion Estimation with Deep Learning

Object motion estimation is essential in the computer vision field, such as human body motion estimation. The actual object motions are almost random, and non-rigid are so complex that a mathematical description cannot calculate them. Nevertheless, accurate object motion estimation can improve visual odometry by eliminating dynamic influence, as discussed in the above section.

Zhou *et al.* [4] tried to estimate the object motion mask in the networks, but it was very rough and inaccurate. Zhou *et al.* [131] introduced a semantic segmentation network for dynamic objects' masks. Yang *et al.* [132] proposed the unsupervised moving object detection via contextual information separation. This work trained a deep neural network to predict the moving context by using the optical flow predicted by the network, and another network attempts to make such context as uninformative as possible. One generator generates dynamic masks. The other is an in painter that tries to inpaint back the optical flow masked out by the corresponding mask to implement unsupervised dynamic object detection. RNN is also introduced into the object motion estimation, such as work in [133]. To sum up, these object motion estimation methods in computer vision provide many inspirations for dealing with dynamic objects in visual odometry.

2.5 Optical Flow Estimation with Deep Learning

Optical flow estimation is a classic research problem in the field of computer vision. As discussed above, this task defines the movement of an object in an image of a video image sequence. This movement can be caused by camera movement or object movement. To facilitate the solution, the traditional optical flow estimation algorithm is generally based on the following assumptions: 1) The assumption of constant brightness: the brightness of the same point changes with time, and its brightness will not change. 2) For small movements,

changes in time will not cause drastic changes in position. 3) The assumption of consistent space: the adjacent points in a scene projected to the image are adjacent points, and the speeds of the adjacent points are the same. The classic traditional optical flow algorithms include LK optical flow [134], PCA-Flow [135], EpicFlow [136], FlowFields [137] and other algorithms. However, the balance of accuracy and speed of these algorithms often restricts their wide applications in practical engineering.

The development of deep learning has brought a breakthrough development in this optical flow estimation: On the one hand, in terms of supervised learning, the data set is constructed through virtual environments, and the deep learning network based optical flow estimation surpasses the traditional algorithms in terms of speed and accuracy. The optical flow networks have extensively promoted the development of this field. On the other hand, by developing unsupervised learning algorithms, the optical flow estimation network based on deep learning has an accuracy level that has reached the traditional optical flow algorithm. Its speed is far beyond the traditional algorithms. The following sub-sections will discuss the related work from the supervised networks to the unsupervised networks.

2.5.1 Supervised Optical Flow Learning

In terms of supervised learning networks, Wulff *et al.* [138] firstly proposed to introduce CNN into the field of optical flow estimation in 2015. They proposed the FlowNet network structure, which feeds two consecutive images as input, and the CNN is directly used for end-to-end training. The output is an optical flow image in the original input image size. To build the data set, the author synthesized the Flying Chairs data set for training. The accuracy of this algorithm is slightly lower than the traditional algorithms, but its speed is much higher than that of the traditional algorithms. Furthermore, the same team of FlowNet proposed FlowNet 2.0 [139] based on the original network that has significantly improved the performance of optical flow estimation through three strategies: including novel data collection and improved training method, multiple network stacking, and small displacement network design. FlowNet 2.0 is close to the best algorithm of traditional optical flow, and its speed is much faster than traditional algorithms. In 2017, SpyNet [140] utilized the pyramid concept in the traditional algorithms to estimate the optical flow from coarse to fine-scale, which better handles the large motion and the small motion problems. When the time came into

2018, the most popular PWCNet [2] continued to introduce the cost volume into the classic networks, which improved the performance of the networks and realized end-to-end training at the same time. This work's contributions make the PWCNet become the new baseline for subsequent optical flow algorithms. Ren *et al.* [141] further proposed PWCNet-fusion to extend PWCNet into a multi-frame information fusion structure with further improved performance in 2019. In addition, LiteFlowNet [142] was proposed in the same year, which used a similar cost volume pathway but introduced the cascading optical flow prediction and feature regularization further to improve the performance of optical flow estimation. In 2020, this team extended LiteFlownet to version 3.0 in [143] which further improved the performance by modifying the network structure and training method. There are a large number of works exploring the optical flow estimation in the computer vision field, such as Hur *et al.* [144] introduced the occlusion map and significantly reduced the number of network parameters through weight sharing for the occlusion problem.

2.5.2 Unsupervised Optical Flow Learning

In terms of unsupervised learning networks, Meister *et al.* [145] proposed UnFlow in 2018, which achieved good results by treating optical flow estimation as an image reconstruction problem. Due to the lack of ground truth labels as training supervision, the occlusion problem in the unsupervised optical flow estimation algorithm is more serious. Janai *et al.* [146] firstly estimated the occlusion map through forward and backward optical flows. The occlusion information by this method is ignored in the image reconstruction loss, which effectively improves the performance of the optical flow estimation. Liu *et al.* [147] proposed DDFlow, which learns to predict the optical flow through the teacher network and artificially constructs occlusion information in the student network. The student network uses the labels in the teacher network as the truth value for training, thereby solving the truth value problem of occlusion.

Due to the similarity of unsupervised implementation principles, optical flow estimation networks treat their training processes like joint unsupervised depth and ego-motion estimation by image reconstruction. Some works completed the optical flow estimation while improving the depth and attitude estimation, such as in [94], [104] and [103]. DF-Net [102] implemented the joint estimation of depth and optical flow by using cross-task consistency.

Because the depth and ego-motion estimation is based on the static scenario assumption, the rigid optical flow can be synthesized. From the dynamic masking of these works, the depth and ego-motion can build the rigid optical flow to warp the image compared with the image warped from full optical flow. However, the rigid optical flow from depth and ego-motion is rougher than the full optical flow, such as dynamic objects. Suppose the depth estimation makes up the dynamic objections and ego-motion estimation processes non-rigid motion for each pixel. In that case, the synthetic optical flow is infinitely close to the actual optical flow. Whether the optical flow is estimated as a direct or indirect target, this task still plays an essential role in deep learning-based visual odometry.

2.6 High-Level Object Segmentation with Deep Learning

Object segmentation is a high-level task for robot's SLAM systems. Traditional robots rely on sensors to measure various physical quantities to perceive the surrounding environment. How to perceive and understand the surrounding environment more like humans has become the new goal of robots in the intelligent era. Individuals who walk on the road rely on high-level information such as the name of a building, road sign direction, and object's shape. Traditional geometric visual odometry cannot complete this kind of work effectively. Nevertheless, deep learning networks are naturally suitable for such high-level object segmentation tasks. Based on the recent research, three kinds of main high-level object segmentation methods could be applied in visual odometry tasks: semantic segmentation, instance segmentation, and dynamic object detection.

2.6.1 Semantic Segmentation

Image segmentation is one of the most popular research topics in image processing and computer vision. There has been a substantial amount of works aimed at developing image segmentation approaches using deep learning models as introduced in [148]. This section presents an essential summary as the base for own contributions.

It is known that the task of image classification is to classify a picture into a specific category, while semantic segmentation is a further step from classification: those belonging to the same category must be classified into one category with a given semantic meaning. The rise of deep learning has dramatically improved the accuracy of semantic segmentation al-

gorithms, and the researcher's enthusiasm for traditional semantic segmentation algorithms has gradually begun to decline. In 2012, Ciresan [149] used CNN to challenge semantic segmentation tasks. This work adopts a sliding window method to take a small image patch centered on each pixel and input it into the CNN to predict the semantic label of the pixel. This is a significant attempt to break the precedent that CNN is only used for target classification. However, the disadvantages of this method are also obvious: firstly, it is necessary to traverse each pixel to extract the patch for training and prediction, which is slow and time-consuming; In addition, choosing an appropriate size of the window is a problem that too minor lacks context information and too large will increase much calculation. Undoubtedly, there are a lot of redundant calculations between many windows.

In 2015, Girshick *et al.* [150] proposed the first deep learning model applied in the direction of target detection: Region-based Convolutional Neural Network (R-CNN). The main process begins with using the selective search algorithm to extract 2000 candidate boxes, then uses the convolutional network to perform serial feature extraction on the candidate boxes. Afterward, use Support Vector Machine (SVM) to classify and predict the candidate boxes according to the extracted features. Finally, use the regression method to correct the area frame. Due to the low efficiency of R-CNN, two improved versions were proposed by Girshick *et al.* [151] and Ren *et al.* [152] in 2015.

A greatly improved semantic segmentation accuracy is achieved after the proposed Full Convolutional Neural Network (FCN). This model completely changes the previous concept that a window is needed to transform a semantic segmentation task into a picture classification task. FCN completely discards the fully connected layer in the picture classification task and only uses the convolutional layer from start to finish. This model defines the process of extracting features as an encoder, that is, the stage where the feature map of FCN becomes smaller in front. The following process of upsampling and deconvolution is called a decoder. The picture is restored to the original size in the decoder. This model is discussed as the encoder-decoder in the above deep neural network model section. After the FCN, the classic network structure based on the encoder and decoder structure has sprung up like mushrooms after rain. The following are representative papers that are essential in the direction of semantic segmentation: U-Net [153], DeepLab serious [154] [155] [156] [157], and PSPNet [158].

Due to the limitation of the convolutional layer structure, the context information provided by FCN is insufficient and needs to be improved. Therefore, various methods have been proposed to explore context dependence to obtain more accurate segmentation results in recent years. There are two main methods for aggregating contextual information: the pyramid-based approach and the attention-based approach. PSPNet [158] is a pyramid-based module or global pooling to regularly aggregate regional or global context information. However, this method captures the context of the same kind but ignores the context of different categories. Attention-based approaches such as channel attention and spatial attention selectively aggregate contextual information between different categories like in DANet [159], CCNet [160], Yu *et al.* [161] and Hou *et al.* [162].

2.6.2 Instance Segmentation

Instance Segmentation is a relatively tricky task of visual perception. It has the characteristics of semantic segmentation requiring classification at the pixel level and the characteristics of object detection. In other words, different instances need to be located, even if they are of the same type. Therefore, the study of instance segmentation has been divided into two pathways: the two-stage method and the single-stage method.

The two-stage method can be divided into top-down and bottom-up. The idea of the top-down instance segmentation method is to first find out the bounding box of the instance through target detection and then perform semantic segmentation in the detection box. Each segmentation result is outputted as a different instance at last. The masterpiece of this type of method is the famous Mask-RCNN [163] and PANet [164]. Mask-RCNN can complete tasks such as target classification, target detection, semantic segmentation, instance segmentation, and human pose estimation by adding different branches. For instance, a branch is added for semantic segmentation based on Faster-RCNN with classification and regression branch. Huang *et al.* [165] improved the accuracy of Mask-RCNN through mask quality scoring, which is named MS-RCNN. The idea of the bottom-up instance segmentation method is first to perform semantic segmentation at the pixel level and then distinguish different instances employing clustering and metric learning. There are not so many such works like introduced in [166].

The single-stage method is affected by the research of single-stage target detection.

Therefore, there are two ways of thinking: One is inspired by one-stage, anchor-based detection models such as YOLO [167], and RetinaNet [168]. Representative works include YOLACT [169] and SOLO [170].

2.6.3 Dynamic Object Detection

Semantic segmentation and instance segmentation can provide object-level information for SLAM. However, dynamic objects are the main noise that could cause interference for depth and ego-motion joint estimation in visual odometry. Dynamic object detection becomes the further task of artificially defined higher-level information. Bak *et al.* [171] introduced the dynamic object detection through visual odometry. Xiao *et al.* [172] utilized a deep learning network to detect dynamic objects during semantic segmentation of the SLAM. Specifically, Zhou *et al.* [4] and Wang *et al.* [94] tried to detect the dynamic objects during the unsupervised depth and ego-motion joint training. The visual odometry performs much better when the learning network achieves more accurate dynamic object detection results.

To sum up, high-level object segmentation plays a vital role with the applications such as scene understanding, image analysis, robotic perception, video surveillance, and augmented reality. It can be seen as a dominant task such as semantic segmentation for semantic SLAM. It can be used to improve the performance of visual odometry networks as an implicit dynamic detection task.

2.7 Discussion

This chapter lists the related work of the traditional geometry method for visual odometry, which contains feature point methods, direct methods, and semi-direct methods. The following learning-based visual odometry firstly provides a summary of the deep learning methods. The following sub-sections introduce the unsupervised learning implementation, deep neural network models, and popular training dataset for different learning-based tasks. The following four sections present the related representative work for disparity estimation, ego-motion estimation, optical flow estimation, and high-level object segmentation. This literature review part lays the research foundation for the whole thesis's work. From the next chapter, the thesis begins to introduce my work: SGANVO, RSTNet, and coalescing semantic segmentation into RSTNet and leverage geometry method for ego-motion estimation.

Chapter 3

Stacked Generative Adversarial Networks based Visual Odometry

Recently end-to-end unsupervised deep learning methods have demonstrated an impressive performance for visual depth and ego-motion estimation tasks. These data-based learning methods don't rely on the same limiting assumptions that geometry-based methods do. The encoder-decoder network has been widely used in the depth estimation and the RCNN has brought significant improvements in the ego-motion estimation. Furthermore, the latest use of Generative Adversarial Nets (GANs) in depth and ego-motion estimation has demonstrated that the estimation could be further improved by generating pictures in the game learning process. This chapter proposes a novel unsupervised network system for visual depth and ego-motion estimation: Stacked Generative Adversarial Network (SGANVO). It consists of a stack of GAN layers, of which the lowest layer estimates the depth and ego-motion while the higher layers estimate the spatial features. It can also capture the temporal dynamic due to the use of a recurrent representation across the layers. See Figure 3.1 for details. We select the most commonly used KITTI [118] data set for evaluation. The evaluation results show that our proposed method can produce better or comparable results in depth and ego-motion estimation.

3.1 Introduction

Object's depth estimation and camera's ego-motion estimation are two essential tasks in autonomous robotic applications. Impressive progress has been achieved by using vari-

ous geometric based methods. Recently, unsupervised deep learning based methods have demonstrated a certain level of robustness and accuracy in some challenging scenes without the need of labeled training data set.

Inspired by the image wrapping technique - spatial transformer [116], Garg et al. [86] proposed an unsupervised Convolutional Neural Network (CNN) to estimate the depth by using the left-right photo-metric constraint of stereo image pairs. Godard [1] further extended this method by employing a loss function with the left and right images wrapping across each other. Because both left and right photo-metric losses are penalized, this method improved the accuracy of depth estimation. Zhou et al. [4] proposed two separate networks to infer the depth and ego-motion estimation over three temporal consecutive monocular image frames. The middle frame performs as the target frame and the previous and the following frames as the source frames. Yin [103] proposed the GeoNet which not only estimates the depth and ego-motion but also the optical flow for dynamic objects. Similarly, Godard et al. [6] proceeded to merge the pose network with the depth network through sharing the network weights. Furthermore, benefit from the recent advance of deep learning methods for single-image super-resolution, Pillai et al. [173] proposed the sub-pixel convolutional layer extension for obtaining depth super-resolution. Their superior pose network is bootstrapped with much more accurate depth estimation.

Most of recent work are all based on the encoder-decoder neural networks, but with different loss functions based on view reconstruction approach. Li *et al.* [93] designed their loss functions not only combining the loss functions defined in [1] and [4] but also adding the 3D points transformation loss. However, their results perform not very well in the scenes where dynamic and occluded objects occupy a large part of the field of view due to artificially designed rigid transformation used. To deal with dynamic and occluded objects in the scenes, Ranjan [104] and Wang [94] introduced additional networks to estimate the optical flow jointly with the depth and pose networks from videos. Ranjan used two networks to learn the optical flow and dynamic object masks, and jointly compute the flow and mask loss functions to refine the depth and pose networks. Wang added the optical flow estimation network to deal with the dynamic objects and refine the depth network. Due to the use of a PWC network [2] to handle the stereo depth estimation, Wang's Undepthflow improved the unsupervised depth estimation by an order of magnitude and its pose estimation also

reached the optimal rank. Almalioglu [129] introduced a visual odometry (GANVO) to estimate the depth map using Generative Adversarial Network [115]. Its Generator network generates the depth map and the pose regressor infers the ego-motion. Together they build up the view reconstruction. Then its Discriminator network games the original target image and the reconstructed image to refine the depth and pose networks. This work is similar with [174] [89] [90] [175], which use the Generator to generate the depth map instead of using estimation networks. From these work, it can be seen that the generative nature in GAN networks is beneficial to the scenes with dynamic objects. However, these visual odometry related GANs focus on the depth estimation, but not on the ego-motion estimation.

In this chapter, we propose a novel unsupervised deep visual odometry system with Stacked Generative Adversarial Networks (SGANVO) (see Figure 3.1). Our main contributions are as follows:

- To the best of our knowledge, this is the first time to use the stacked generative and adversarial learning approach for joint ego-motion and depth map estimation.
- Our learning system is based on a novel unsupervised GAN scheme without the need for ground truth.
- Our system possesses a recurrent representation which can capture the temporal dynamic features.

The outline of this chapter is organized as follows: Section 3.2 gives an overview of our proposed SGANVO system. Section 3.3 describes various loss functions used for the Generator and Discriminator. Section 3.4 presents our experimental results of depth and ego-motion estimation. Finally, conclusion and future work are drawn in Section 3.5.

3.2 System Overview

We address the depth and the ego-motion estimation as a whole visual odometry system in Figure 3.1. The system consists of an ego-motion representation layer for ego-motion estimation, and multiple feature extraction layers for feature estimation. The error image A^l from error unit E^{l-1} in current layer is propagated up to higher layer as the input. The hidden state in recurrent representation layer R^l is propagated down to lower layer for capturing the

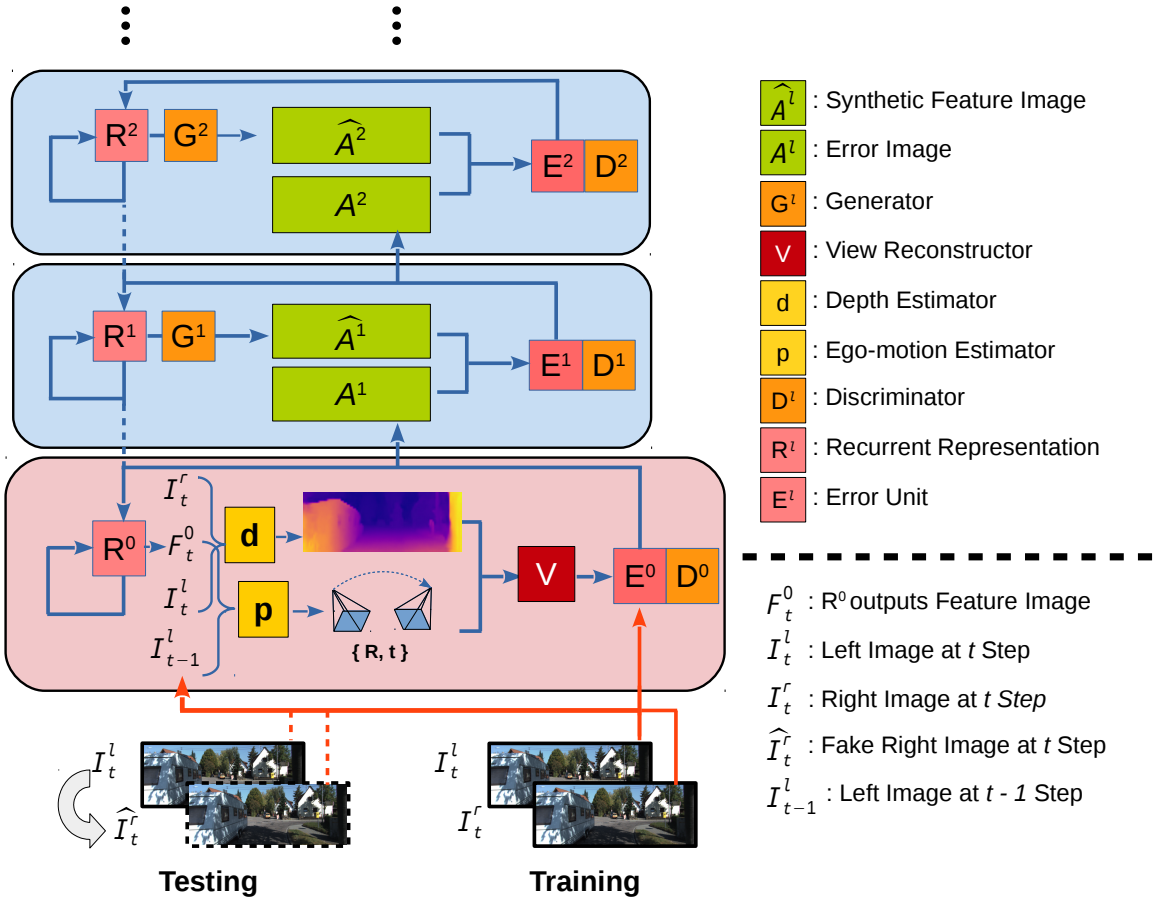

Figure 3.1: Our proposed SGANVO architecture for the depth and ego-motion estimation. It is a series of stacked GANs. The bottom layer estimates the depth and ego-motion. The other layers estimate the spatial features.

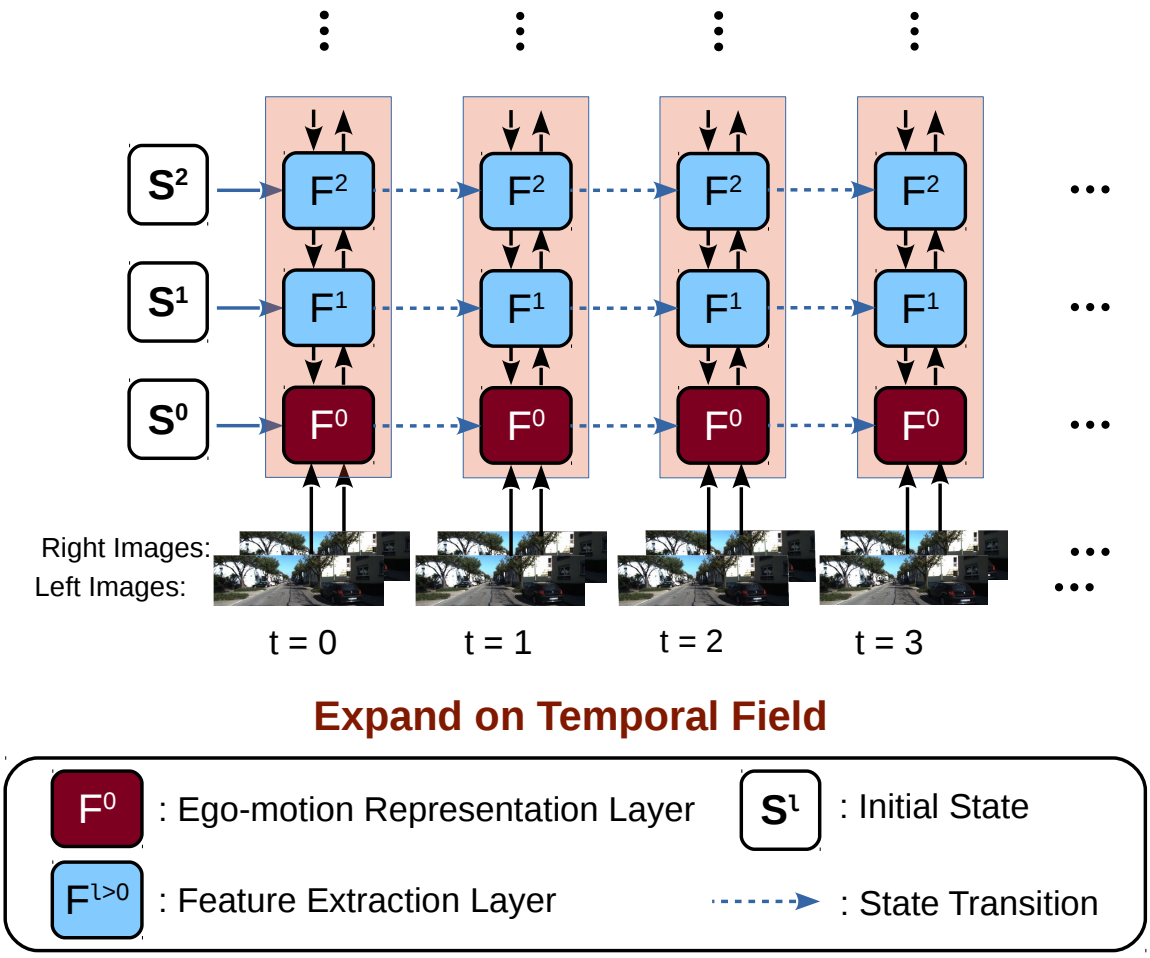

Figure 3.2: The network is unfolded in time. The temporal dynamic is captured in the recurrent representation.

temporal dynamic. There are mainly two units, Generator (G^l) and Discriminator (D^l), in each layer.

Figure 3.2 shows the network unfolded in time step. After the initial states S^l are set to the network, the current left and right frames are fed to the network's input. The network runs for a sequence of N consecutive frames. After N steps, the initial states are set to the network again.

3.2.1 Generator

In the bottom layer, the Generator consists of depth estimator d, ego-motion estimator p, and view reconstructor V. It takes the hidden state from recurrent representation R^0 as its input and generates an estimated image from view reconstructor V for current input image. In the higher layer ($l > 0$), the Generator is a convolutional layer which takes the hidden

state from recurrent representation R^l as its input and generates an estimated error image $\hat{A}^l$ for the input A^l from lower layer. The recurrent representation R^l is a convolutional Long Short Term Memory network (ConvLSTM) [176] and connected top down between layers. The hidden state R^l_t is updated according to $R^l_{t-1}, R^{l+1}_t, E^l_{t-1}$. It is used to store the temporal dynamic from consecutive video which is defined as:

$$R^l_t = ConvLSTM(E^l_{t-1}, R^l_{t-1}, UpSample(R^{l+1}_t)) \tag{3.1}$$

The hidden state R^l_t is updated through two passes: a top down pass from the *UpSample* of R^{l+1}, and then a level pass calculated by previous hidden state R^l_{t-1} and previous error E^l_{t-1}.

The depth estimator is an encoder-decoder network to generate the multi-scale dense depth map, like the one used in [1]. Inspired by [173], we replace the *UpSample* branches in the depth decoder with the sub-pixel convolutional branches used in [177] to generate the up-scaled features. Sub-pixel convolutional branches consist of a sequence of three consecutive 2D convolutional layers with 64, 32, 4 output channels with 1 pixel stride. The depth estimator directly generates the dense depth map by using a pair of stereo images separately concatenating the features from unit R^0_t along with the image channel to train the network.

The ego-motion estimator is a VGG-based CNN architecture. It takes current image and the hidden state representing the information in previous frame as the input, and generates the 6-DoF ego-motion estimation between current frame and previous frame. We decouple the translation and the rotation with two separate groups of fully-connected layers after the last convolutional layer for better performance. We feed the input data concatenating two continuous images and the features from unit R^0_t along with image channel to train the ego-motion estimator.

If we feed one image into the stacked networks each time, the pose estimator will generate a 6-DoF ego-motion for two consecutive frames. However, the initial frame of each temporal window will introduce the failure initialization of pose estimation. To resolve that, we reuse the first frame as the previous frame of beginning and start to compute pose translation from the second temporal frame.

The view reconstructor V takes the depth estimate, ego-motion estimate, and previous frame I_{t-1} as the input and generates a predicted image $\hat{I}_t$ for the current frame I_t. By using

the spatial transformer network [116] and the homogeneous coordinate of the pixel $I_{t-1}(i,j)$ in the $(t-1)$th frame, we can derive its corresponding pixel $\hat{I}_t(i,j)$ in the tth frame through

$$\hat{I}_t(i,j) = K\hat{T}_{t-1,t}d_{t-1}^{-1}(i,j)K^{-1}I_{t-1}(i,j) \tag{3.2}$$

where K is the camera intrinsic matrix, $d_t(i,j)$ is the estimated disparity, $\hat{T}_{t-1,t}$ is the camera coordinate transformation matrix from the $(t-1)$th frame to the tth frame generated by the ego-motion estimator.

Based on this, $\hat{I}_{t-1}$ and $\hat{I}_t$ can be constructed from I_t and I_{t-1}, respectively. The error units in the bottom layer or higher layers are computed as below:

$$E_t^0 = [ReLU(I_t - \hat{I}_t); ReLU(\hat{I}_t - I_t)] \tag{3.3}$$

$$E_t^l = [ReLU(A_t^l - \hat{A}_t^l); ReLU(\hat{A}_t^l - A_t^l)] \tag{3.4}$$

where E_t^0 is the error image between the generated from the view reconstructor and the current frame at t time, E_t^l is the error image at the higher layer ($l > 0$), $ReLU$ is an activation function operation. The input error image A_t^l is defined as:

$$A_t^l = Maxpool(ReLU(Conv(E_t^{l-1}))) \tag{3.5}$$

where $Conv$ denotes the convolutional operation. The Generator in the higher layer ($l > 0$) consists of one convolutional unit with 3 dimension filters to generate error feature image $\hat{A}_t^l$:

$$\hat{A}_t^l = ReLU(Conv(R_t^l)) \tag{3.6}$$

3.2.2 Discriminator

The Discriminator is a convolutional network that can categorize the images fed to it. In the bottom layer, we feed the generated image $\hat{I}_t$ and original image I_t into it. In the higher layer ($l > 0$), the inputs to the Discriminator are the generated error image $\hat{A}_t^l$ and the error image A_t^l from the lower layer. We used WGAN [178] in our architecture, which employs the Wasserstein distance instead of Kullback-Leibler (KL) divergence or Jensen-Shannon (JS) divergence for stabilizing the training process using gradient descents. The Discrimi-

Table 3.1: Discriminator Network Architecture.

Block Type	Kernel Size	Strides	Filters	Input
Conv1	5×5	2	16	Image
SELU1	none	none	none	Conv1
Conv2	5×5	2	32	SELU1
SELU2	none	none	none	Conv2
Conv3	5×5	2	64	SELU2
SELU3	none	none	none	Conv3
Conv4	5×5	2	128	SELU3
SELU4	none	none	none	Conv4
Conv5	4×4	1	1	SELU4

nator architecture is shown in Table 3.1. Considering the GPU's memory space, we set all the Discriminators five simple convolutional layers connected by four SELU block units as provided in [179].

The bottom base layer of the stack networks takes stereo images as input at each time step within a consecutive temporal window. Then bottom base computes the error between reconstruction and original image and feed to the high-level's Generator and the bottom feed yielding component. The feed yielding component produces the dynamic features for ego-motion estimator and depth estimator to respectively generate the scaled 6-DoF ego-motion and depth maps. From the second layer of stack networks, Generator only uses the difference of the original feed image and reconstructed image from the lower layer as its feed. Analogously, the higher Generator generates such higher dimensional difference as supervised object for gaming through Discriminator. The higher Generators learn the temporal dynamic features and the bottom Generator learns the spatial appearance feature.

3.2.3 Stereo Generating

For monocular SGANVO's training and testing, we apply WGAN to learn stereo generating technology to feed into SGANVO system as shown in Figure 3.3. Different from [122], we implement Generator process generating right images from left images. In Generator, left images sequence are feed into VGG encoder network and achieve five scaled outputs. We use five deconvolution layers to predict five scaled feature masks. Original left images multiple the sum of scaled feature masks to generate right images. Afterwards, both the original right image and projected right image separately feed into Discriminator. The fake image output and real image output of Discriminator form the WGAN loss functions for training. This component can build the stereo images input for SGANVO's training and testing experiments with monocular data set.

3.3 Training Procedure

The SGANVO training follows the improved WGAN training procedure [180] like above stereo generating component. We jointly compute the losses of Generators and Discriminators in all the layers and take the weighted sum as the final G loss and D loss. We feed the stereo data sequence of KITTI dataset into the depth estimator, and the monocular data sequences of KITTI dataset into the ego-motion estimator no matter monocular or stereo data set. The loss functions are defined as below:

3.3.1 Generator Loss

Our G losses mainly include three parts. The first one is from the Discriminator and defined as:

$$L_g^D = \sum_l \lambda_l E[D(\hat{x}^l)] \tag{3.7}$$

where λ_l is the weight at lth layer, and $D(\hat{x}^l)$ is the output from the Discriminator when its input is from generated image $\hat{x}^l$ in its layer. E is the mean operation.

The second one is the weighted sum of all the error images within a sequence of N consecutive frames.

$$L_g^N = \sum_{t=1}^{N} \lambda_t \sum_l \frac{\lambda_l}{n_l} \sum_{n_l} \|E_t^l\|_1 \tag{3.8}$$

where λ_t is the temporal weight factor, λ_l is the layer weight factor, n_l is the total number of

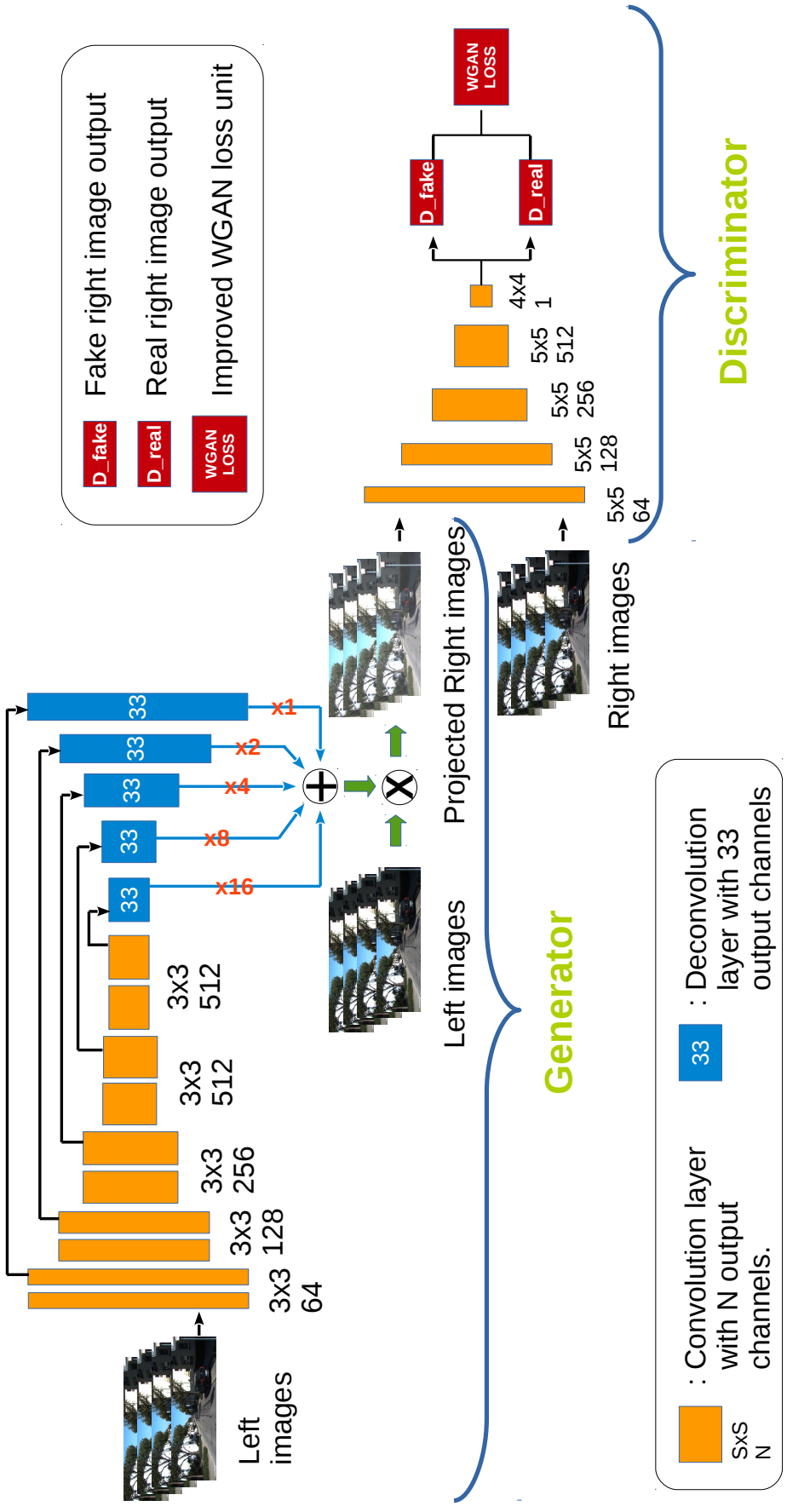

Figure 3.3: SGANVO Stereo Images Generating Networks

ReLU units of the lth layer, and $\|\cdot\|_1$ is the L_1 norm.

The last one is the disparity consistency loss which is used to improve the depth smoothness. Denote d_t^l and d_t^r the left and right disparity maps, respectively. The disparity consistency loss is defined as:

$$L_g^d = \sum_{t=1}^{N} \sum_{i,j} \|d_t^l(i,j) - d_t^r(i,j)\|_1 \tag{3.9}$$

The final G loss is the weighted sum of above three parts.

$$L_g^{final} = \alpha L_g^D + \beta L_g^N + \gamma L_g^d \tag{3.10}$$

where α, β and γ are the weight parameters.

3.3.2 Discriminator Loss

The D loss is defined as:

$$L_d^D = \sum_l \lambda_l \arg\max_D D_d^l \tag{3.11}$$

where D_d^l is the Discriminator loss in the lth layer when its inputs are real image x and generated image $\hat{x}$. Afterwards the D_d^l is defined as :

$$D_d^l = E[D(x^l)] - E[D(\hat{x}^l)] + \lambda_D E[(||\nabla D(x'^l)||_2 - 1)^2] \tag{3.12}$$

where E is the mean operation and ∇ is the solving gradient operation. The innovation of WGAN-GP is on the last item of the above loss function where the x' is the random interpolation samples between real value and generated value defined as:

$$x' = x * \varepsilon + \hat{x} * (1 - \varepsilon) \tag{3.13}$$

In our implementation, we choose $\lambda_D = 10$ and ε is a random number from uniform distribution between 0 and 1.

3.3.3 Adversarial Training Procedure

To sum up, our SGANVO generates the features of temporal field over a time window and of spatial field over stacked layers. It can estimate the depth and the ego-motion at current

time step from previous frame in the bottom layer.

Initially, the first frame in the first temporal window is fed to the network. This frame is also used as the initial previous frame to resolve the initial matrix irreversible problem. Thus our system can generate one depth estimation and one ego-motion estimation at each time step of the window. We set all the initial states S^l to zeroes. They are passed to the recurrent representation R^l. The synthetic error image $\hat{A}^l$ are generated as zero matrices. And the initial estimate of ego-motion is zero as the initial current input and initial previous image are the same. But an initial depth map can be estimated from the depth estimator.

After the initialization, the first frame is fed to the system, and the system begins to generate the results in each layer.

After completing the feeding of current temporal window to the network, the system computes all the G the D losses over all the temporal steps (N). At the same time, it can generate both the depth and ego-motion estimate. Then the final G and D losses are applied to back propagate the Generators and Discriminators. The min-max training of our SGANVO is described as:

$$W(x,\hat{x}) = \arg\min_{D} L_d^D \tag{3.14}$$

The Discriminator training procedure is to make the $W(x,\hat{x})$ convergent to the minimum. The above procedure is repeated for next temporal window until all the frames in the data set are used.

3.4 Experiments

We implemented the proposed SGANVO architecture with the publicly available Tensorflow framework and trained with NVIDIA GTX 1080TI GPUs. The Adam optimizer was employed to speed up the network convergence for up to 30 epochs with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate started from 0.0001 and decreased by half for every 1/5 of total iterations. For the ease of training and data preparation, we used a temporal window size of $N = 3$ and two higher layers $L = 2$ and one bottom layer, but it is possible to use longer sequences and more higher layers for training and testing. The size of input image to the network was 416×128 with the consideration of comparison with other systems. To fine-tune the network, we used the original image size in computing the losses. For data prepossessing, different kinds of data augmentation methods were used to enhance the per-

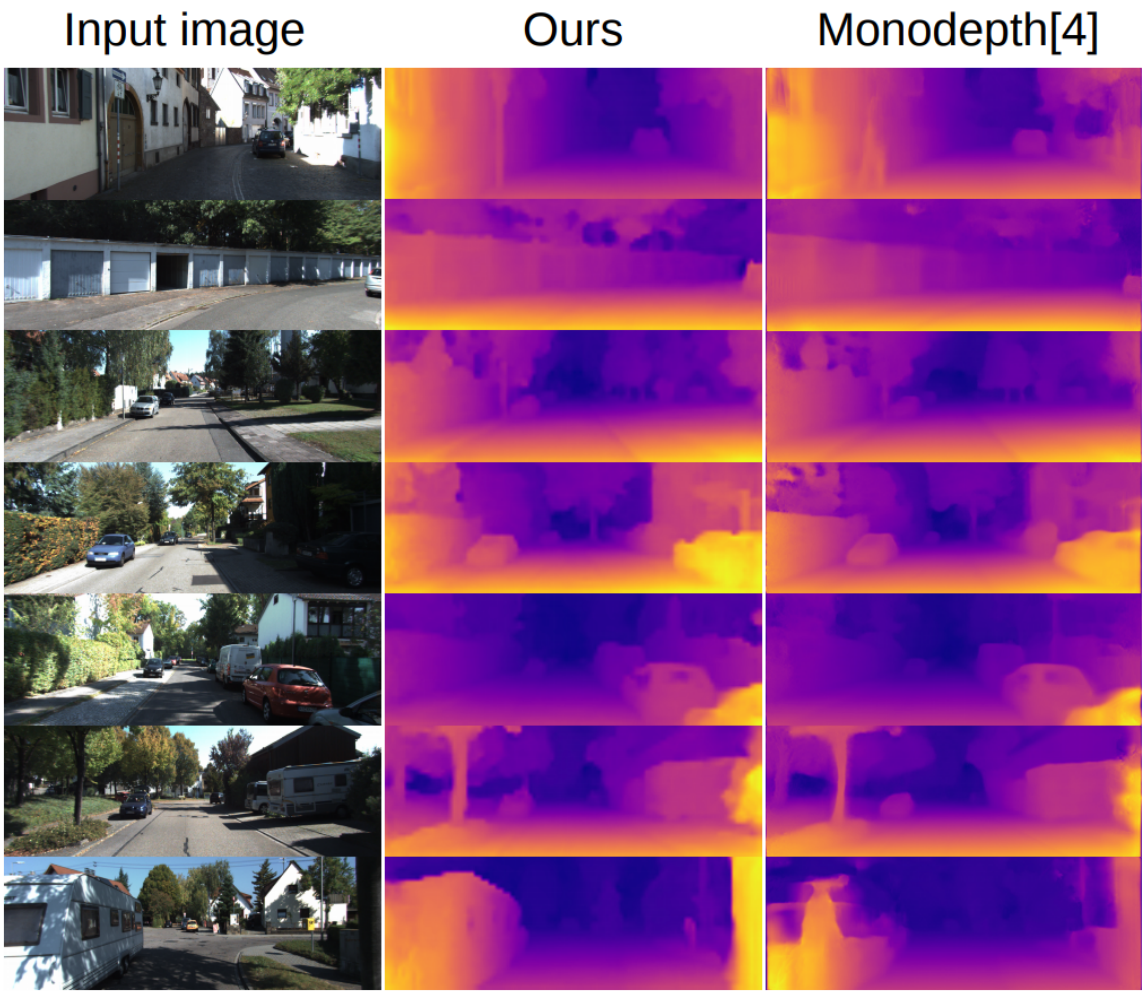

Figure 3.4: Illustrated above are qualitative comparisons of our SGANVO (image size: 416×128) with Monodepth [1] (image size: 512×256). The depth map shows that our approach produces qualitatively better depth estimates with crisp boundaries.

formance and mitigate possible over-fitting, such as image color augmentation [4], rotational data augmentation [181] and left-right pose estimation augmentation [1]. We increased the weight parameter of rotational data to achieve better performance because the magnitude of rotation is very small compared to that of translation. We set $\alpha = 10^{-4}$, $\beta = 1.0$ and $\gamma = 0.1$ in the final G loss Equation (3.10). To test our SGANVO's depth estimation on the monocular dataset, we built a stereo view reconstruction component like [182] to generate the stereo frames from the monocular dataset (see the testing inputs in Figure 3.1). Our SGANVO can test directly on the monocular dataset.

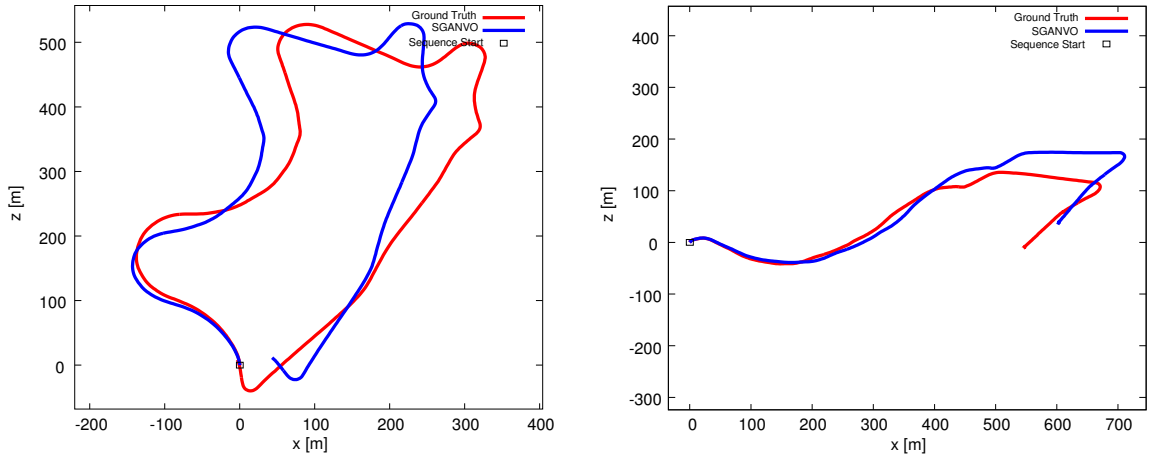

Figure 3.5: Our proposed SGANVO system to estimate the ego-motion on the KITTI odometry benchmark, Sequence 09 (left) and Sequence 10 (right). Our results are rendered in blue while the ground truth is rendered in red.

3.4.1 Depth Estimation Evaluation

We evaluate the performance of our SGANVO depth estimation on the KITTI dataset with a benchmark split from [4]. Figure3.4 shows some raw RGB images from sequence 10 and their corresponding depth estimates from Monodepth [1], and our system. As shown in Figure 3.4, the different depths of cars and trees are explicitly generated, even the depth of trunks and street lights are generated clearly. Compared with the others, our SGANVO can generate more details and clearer textures.

The quantitative depth estimation results are listed in Table 3.2, where *M* stands for using monocular image sequence training and *S* stands for using stereo image sequence training. Compared with the existing unsupervised learning-based methods (see the smallest values in Abs Rel, Sq Rel, RMSE, and RMSE log columns in the table with the highest δ), our SGANVO performs better in terms of the metrics used. And it even outperforms some methods with larger image input.

3.4.2 Ego-motion Estimation Evaluation

We used the KITTI dataset [118] to test the performance of the SGANVO ego-motion estimation. The KITTI dataset only provides the ground-truth of 6-DoF poses for Sequence 00-10. We used Sequence 00-08 for training and Sequence 09-10 for testing. The results are shown in Table 3.3.

Table 3.2: Depth estimation results on KITTI dataset using the split of Eigen et al. [70]. For training data, K = KITTI, CS = Cityscapes [119], M = Monocular and S = Stereo. For testing data, our SGANVO uses the monocular dataset like others' work and our view reconstruction component uses this left image sequence to generate the right view image sequence like [182].

Method	Datasize	Dataset	Train	Abs Rel	Sq Rel	RMSE	RMSE log	δ <1.25	δ <1.25^2	δ <1.25^3
Garg et al. [86]	620 × 188	K	M	0.169	1.080	5.104	0.273	0.740	0.904	0.962
SfMLearner [4]	416 × 128	K	M	0.208	1.768	6.856	0.283	0.678	0.885	0.957
SfMLearner [4]	416 × 128	CS+K	M	0.198	1.836	6.565	0.275	0.718	0.901	0.960
GeoNet [103]	416 × 128	K	M	0.155	1.296	5.857	0.233	0.793	0.931	0.973
GeoNet [103]	416 × 128	CS+K	M	0.153	1.328	5.737	0.232	0.802	0.934	0.972
Vid2Depth [101]	416 × 128	K	M	0.163	1.240	6.220	0.250	0.762	0.916	0.968
Vid2Depth [101]	416 × 128	CS+K	M	0.159	1.231	5.912	0.243	0.784	0.923	0.970
GANVO [129]	416 × 128	K	M	0.150	1.414	5.448	0.216	0.808	0.939	0.975
UnDeepVO [93]	416 × 128	K	S	0.183	1.730	6.570	0.268	-	-	-
Godard et al. [1]	640 × 192	K	S	0.129	1.112	5.180	0.205	0.851	0.952	0.978
SGANVO	416 × 128	K	S	**0.065**	**0.673**	**4.003**	**0.136**	**0.944**	**0.979**	**0.991**

Table 3.3: Ego-motion estimation results on KITTI dataset with our proposed SGANVO. We also compare with the four learning methods (Monocular: ESP-VO [181], SfMlearner [4]; Stereo: UndeepVO [93], Undepthflow [94]), and one geometric based methods (ORB-SLAM). Our SGANVO, SfMLearner, and UndeepVO use images with 416×128. ESP-VO and ORB-SLAM use 1242×376 images. Undepthflow uses 832×256 images. The best results among the learning methods are made in bold.

Seq.	Monocular								Stereo			
	SGANVO (416×128)		ESP-VO [181] (1242×376)		SfMLearner [4] (416×128)		ORB-SLAM [31] (1242×376)		UndeepVO [93] (416×128)		Undepthflow [94] (832×256)	
	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$
03	10.56	6.30	**6.72**	6.46	13.08	**3.79**	0.67	0.18	5.00	6.17	-	-
04	**2.40**	**0.77**	6.33	6.08	10.68	5.13	0.65	0.18	4.49	2.13	-	-
05	**3.25**	**1.31**	3.35	4.93	16.76	4.06	3.28	0.46	3.40	1.50	-	-
06	**3.99**	**1.46**	7.24	7.29	23.53	4.80	6.14	0.17	6.20	1.98	-	-
07	4.67	**1.83**	**3.52**	5.02	17.52	5.38	1.23	0.22	3.15	2.48	-	-
09*	**4.95**	**2.37**	-	-	18.77	3.21	15.30	0.26	7.01	3.61	13.98	5.36
10*	**5.89**	**3.56**	9.77	10.2	14.36	3.98	3.68	0.48	10.63	4.65	19.67	9.13

- t_{rel}: average translational RMSE drift $(\%)$ on length of 100m-800m.
- r_{rel}: average rotational RMSE drift $(^\circ/100m)$ on length of 100m-800m.
- train sequence: 03,04,05,06,07; test sequence: 09*, 10*

Table 3.4: Absolute Trajectory Error (ATE) on KITTI odometry dataset. The results of other baselines are taken from [94].

Method	frames	Sequence 09	Sequence 10
ORB-SLAM(Full)	All	0.014 ± 0.008	0.012 ± 0.011
SfMLearner [4]	5	0.016 ± 0.009	0.013 ± 0.009
GeoNet [103]	5	0.012 ± 0.007	0.012 ± 0.009
Undepthflow [105]	2	0.023 ± 0.010	0.022 ± 0.016
SGANVO	3	0.015 ± 0.006	0.014 ± 0.009

The metrics are the average translational root-mean-square error (RMSE) drift and average rotational RMSE drift $(^\circ/100m)$ on length of $100m - 800m$. The results are shown in Table 3.4. We compare the results with ESP-VO [181], SfMLearner [4], ORB-SLAM (without loop closure), UndeepVO [93], and Undepthflow [94]. It can be seen that our SGANVO shows a better performance in the testing sequences (09, 10) with these state-of-the-art methods in terms of the ATE metric.

The other metrics used are the absolute trajectory error (ATE) averaged over all overlapping 5-frame snippets. We concatenated all of left and right estimations together for the entire sequences without any post-processing. The estimated trajectory of sequences 9 and 10 from our SGANVO and its ground truth are shown in Figure 3.5. Although the estimated result includes some drift, our SGANVO can estimate all the features of the trajectory and performs well in terms of odometry estimation without loop closure detection. The quantitative results are shown in Table 3.4 where we compare our results with ORB-SLAM(Full), SfMLearner [4], GeoNet [103], and Undepthflow [94]. Our SGANVO produced a comparable result.

3.4.3 Ablation Analysis for Spatial Layers and Temporal Frames

To demonstrate the importance of the high level spatial features and temporal recurrent window, we conducted three contrast experiments for an ablation analysis. In Table 3.5, we set the same network and training parameters, but configured the system with different number of high level layers and size of temporal window to explore the influence of high level spatial features and temporal recurrent window on the learning results. The main evaluation is focused on the ego-motion and depth test for sequence 09 and sequence 10. Considering the

Table 3.5: Performance for different number of layers and window frame. NL = number of layers, NF = number of frames. Sq.09 and Sq.10 Absolute Trajectory Error(ATE) are the ego-motion testing results. From Abs Rel to $\delta < 1.25^3$ are the depth testing results.

NL	NF	Sq.09 (ATE)	Sq.10 (ATE)	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
2	2	0.0178 ± 0.0083	0.0180 ± 0.0107	0.0991	0.873	4.483	0.184	**0.902**	**0.959**	**0.979**
2	3	0.0152 ± 0.0067	0.0149 ± 0.0094	0.0973	0.733	4.623	0.167	**0.903**	**0.969**	**0.988**
3	3	0.0153 ± 0.0061	0.0147 ± 0.0085	0.0651	0.673	4.003	0.136	**0.944**	**0.979**	**0.991**

GPU's memory and training time, we just tested the system configuration with 2 layers or 3 layers combining with 2 frames or 3 frames.

It can be seen from the data in the first two rows that the ego-motion performs much better when the window size increases from 2 to 3 when the number of high level layers are kept unchanged. In addition, the depth estimation also has a little bit improvement. Then we fixed the same temporal window size as 3 frames to explore the influence of different number of high level layers. It can be concluded from the data in the last two rows of Table 3.5 that the number of high level layers can make a significant improvement on the depth estimation, although its influence on the ego-motion estimation is limited.

From these results, we can conclude that the size of temporal recurrent window can make a significant improvement on the ego-motion estimation while the number of high level layers can make a significant improvement on the depth estimation.

3.5 Conclusions

This chapter proposes a novel stacked GAN network for depth estimation and ego-motion estimation from videos. Because the higher layers of our network can learn spatial features with different abstraction levels and the recurrent representation of our network can learn the temporal dynamics between consecutive frames, our SGANVO can generate more accurate depth estimation results and comparable ego-motion estimation result compared with other existing unsupervised learning networks.

In the future, we will extend our system to a visual SLAM system to reduce the drift by adding a loop closure detection. Moreover, more data sets will be used for training and testing to improve the performance further. After this chapter explores the external network structure for visual odometry, the next chapter will discuss the design of the internal network structure that has a deeper impact on the network's estimation performance.

Chapter 4

Recurrent Spatial-Temporal Networks for Estimating Depth, Ego-Motion, Optical Flow and Dynamic Objects

Depth map and ego-motion estimations from consecutive monocular images are challenging to learning-based Visual Odometry (VO) approaches. With dynamic objects or occlusions in input images, the estimation performance could have further deteriorated. This chapter proposes a novel VO architecture: Recurrent Spatial-Temporal Network (RSTNet), which can estimate the depth map, ego-motion, and dynamic objects from consecutive monocular images. The main contribution in our RSTNet includes a novel RST-encoder layer and RST-decoder layer, which can preserve and recover the spatial and temporal features from inputs in their intermediate representations. Our RSTNet extracts the appearance features from input images and the structure and dynamic features from internal results: depth and optical flow for ego-motion estimation. Our RSTNet also includes a pre-trained network to detect dynamic objects from the difference between complete and rigid optical flows. A novel auto-mask scheme is designed in our loss function to deal with some challenging scenes. Our evaluation results on the KITTI odometry benchmark show that our RSTNet outperforms some of the existing unsupervised learning approaches.

This work is inspired by sensor-less challenging navigation tasks for mobile robots and driver-less cars with a monocular camera. Robots can robustly estimate the continuous location information and record its track map through deep learning networks. This chapter

designs a novel spatial-temporal feature learning network for estimating a robot's track and surrounding dynamic objects and proposes an auto-mask training strategy to remove the dynamic interference during the learning process. From the evaluation results on monocular videos collected from an actual vehicle on the road, our network estimates the track map of the vehicle and dynamic objects more accurately. Moreover, the efficient learning component of our network can also be flexibly used as a plug-in unit by other networks, which provides a broad practical prospect.

4.1 Introduction

Visual Odometry (VO) pioneers a new path towards solving simultaneous localization and mapping (SLAM) problems. The accurate depth and ego-motion estimations become the primary target of recent research works in this area. It is interesting to see some unsupervised deep learning VO methods outperform classical geometric algorithms and supervised learning methods in some scenes. Monocular VO methods also demonstrate promising results under the framework of learning-based methods like [183].

Most monocular unsupervised learning VO methods reconstruct a frame through warping neighbor frames and use the reconstructed frame as the supervised signal for learning, such as SfMlearner [4] and SfMnet [99]. GANVO [129] proposes a generative adversarial network to produce the depth and ego-motion estimations. Alternatively, the depth can be estimated through training stereo images. For instance, Monodepth [1] uses a stereo learning network with the left-right consistency loss functions to estimate monocular depth. UnDeepVO [93] coalesces the monocular multi-view reconstruction and stereo version training methods to predict the depth maps and camera ego-motion. Furthermore, SGANVO [130] explores the spatial-temporal features with a stacked generative adversarial network to generate a stereo vision loss function.

Although these methods have established some achievements for depth and camera ego-motion estimations, there are still some challenges to be addressed, such as extracting and retaining more spatial and temporal features from input sequences in intermediate layers or detecting and masking dynamic objects in the scenes more robustly. This chapter proposes a novel recurrent spatial-temporal network (see Figure 4.1) for self-supervised monocular depth and camera ego-motion estimations. Our main innovations are summarized as follows:

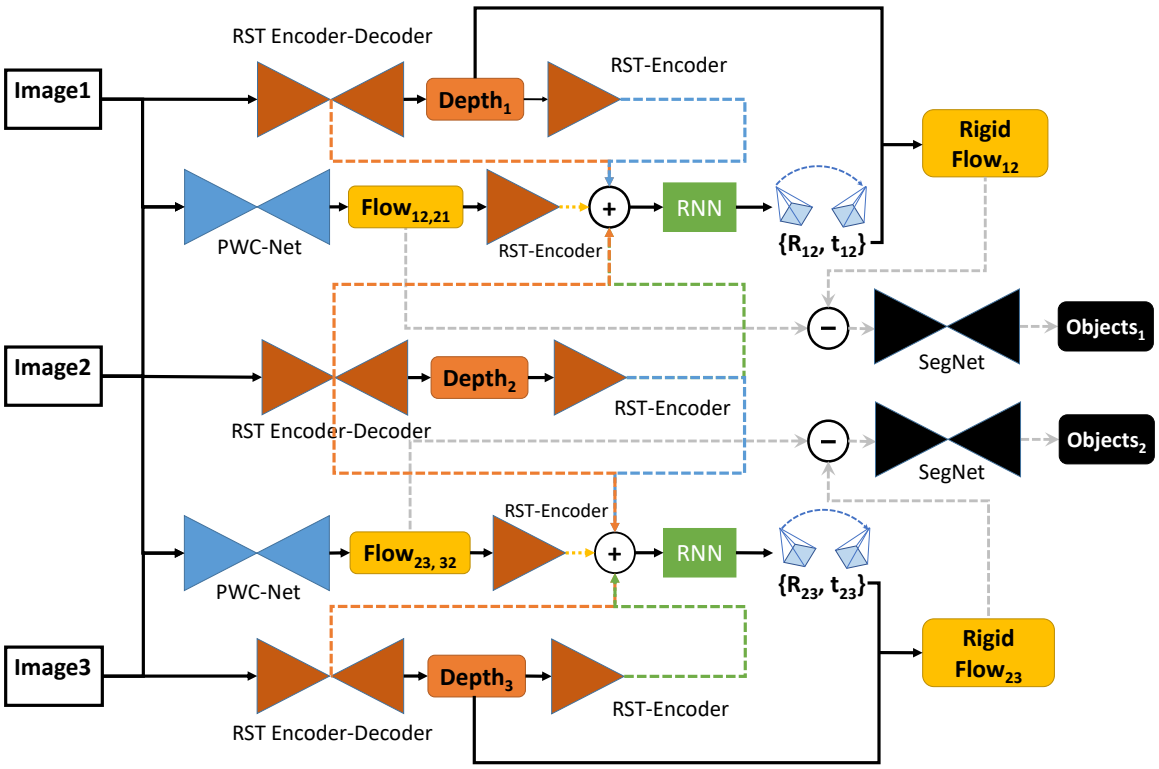

Figure 4.1: Our proposed RSTNet architecture for self-supervised learning. The RST-encoder and RST-decoder components consist of multiple RST-encoder and RST-encoder layers. The depth is estimated from a network including a RST-encoder component and a RST-decoder component. The ego-motion is estimated from a RNN network with inputs from appearance features in input images, structure features from depth, and dynamic features from optical flow. A pre-trained PWC-Net in [2] is used to estimate the full optical flow. The dynamic objects are detected by a pre-trained SegNet [3] with input from the difference flow between full and rigid flows.

- Propose a novel recurrent spatial-temporal network to estimate ego-motion. It uses appearance features from input images, structure features from depth maps, and dynamic features from optical flow.
- Propose novel RST-encoder and RST-decoder layers to learn the detail-preserving representations in a reversible architecture.
- Design a dynamic auto-mask scheme for the loss function, which can robustly mask dynamic objects and some challenging scenes.
- Estimate dynamic objects by using a pre-trained segmentation network from the difference between full and rigid optical flows.

The outline of this chapter is organized as follows: Section 4.2 presents the state-of-the-art works related to depth and ego-motion estimations. Section 4.3 provides the detailed introduction to the RST-encoder and RST-decoder layers. Section 4.4 gives an overview of our proposed RSTNet architecture and its dynamic auto-masking loss function. Section 4.5 presents our experimental results on the KITTI odometry dataset with comparisons of some previous works. Finally, conclusion and future work are drawn in Section 4.6.

4.2 Related Work

We review the research work in estimating the depth and ego-motion from consecutive images using deep learning networks in this section. It starts from self-supervised methods and is followed by methods that could extract more intermediate features or generate more reliable masks for dynamic objects or occlusions.

4.2.1 Self-supervised Depth and Ego-motion Learning Networks

SfMlearner [4] uses the geometric warping technique between neighbor frames to build their unsupervised loss functions for monocular depth and ego-motion estimations. SfMnet [99] extends the work for estimating the object's motion to warp point clouds for generating the optical flow. Both of them predict the dynamic mask to cut down the noise caused by dynamic objects. Nevertheless, the performance of the dynamic mask detection is not reliable. Monodepth [1] trains a single-view depth network by using stereo image pairs. The left-right consistency between the left image and synthesized left image warped from

the right image is used as the supervision signal. Zhan et al. [92] extend the training of reconstruction networks to stereo videos. SGANVO [130] puts forward a stack generative adversarial network-based on stereo pair training. The stereo reconstruction networks do not require the ground truth of depth.

Compared with stereo reconstruction methods, monocular multi-view reconstruction methods have relatively fewer constraints. Godard et al. [6] extends Monodepth [1] to train the depth network and ego-motion network on monocular data sets. They propose an auto-masking loss function to solve the occlusion problem, but the problem caused by dynamic objects, such as cars and pedestrians, still needs to be solved. Yin [103] proposes the GeoNet to train a depth network, an ego-motion network, and a residual optical flow network in two stages. Based on the trained monocular depth and ego-motion networks, residual optical flow can be trained to estimate the full optical flow, including dynamic objects. Prasad [184] makes use of the Epipolar constraints in the learning system so that the system can perform successfully even in the case of failure when minimizing the photometric error. To improve the monocular depth estimation, SC-SfMlearner [5] proposes a geometry consistency loss for scale-consistent predictions and induces a self-discovered mask for handling moving objects and occlusions. However, the performance of self-discovered masks in the unknown marginal region and dynamic object occlusion is limited.

4.2.2 Learning with Intermediate Features

Inspired by the sub-pixel network [177] which performs convolutions with single image super-resolution, Superdepth [173], and Lipu [185] introduce the sub-pixel networks for self-supervised monocular depth estimation to improve the performance of networks. Further digging into the network architecture, Packnet-SFM [7] proposes the 3D packing and unpacking networks to replace the max-pooling layer and bilinear upsample layer of traditional depth networks. The detail-preserving properties of this architecture could reconstruct the near-lossless features of images.

Similar with DDVO [91], Wang [94] introduces additional networks to estimate the optical flow and use the full optical flow to refine the ego-motion estimation. Ranjan [104] proposes four networks to predict the monocular depth, camera ego-motion, optical flow, and dynamic object masks. The 3D geometric constraint is used in [101]. Using semantic

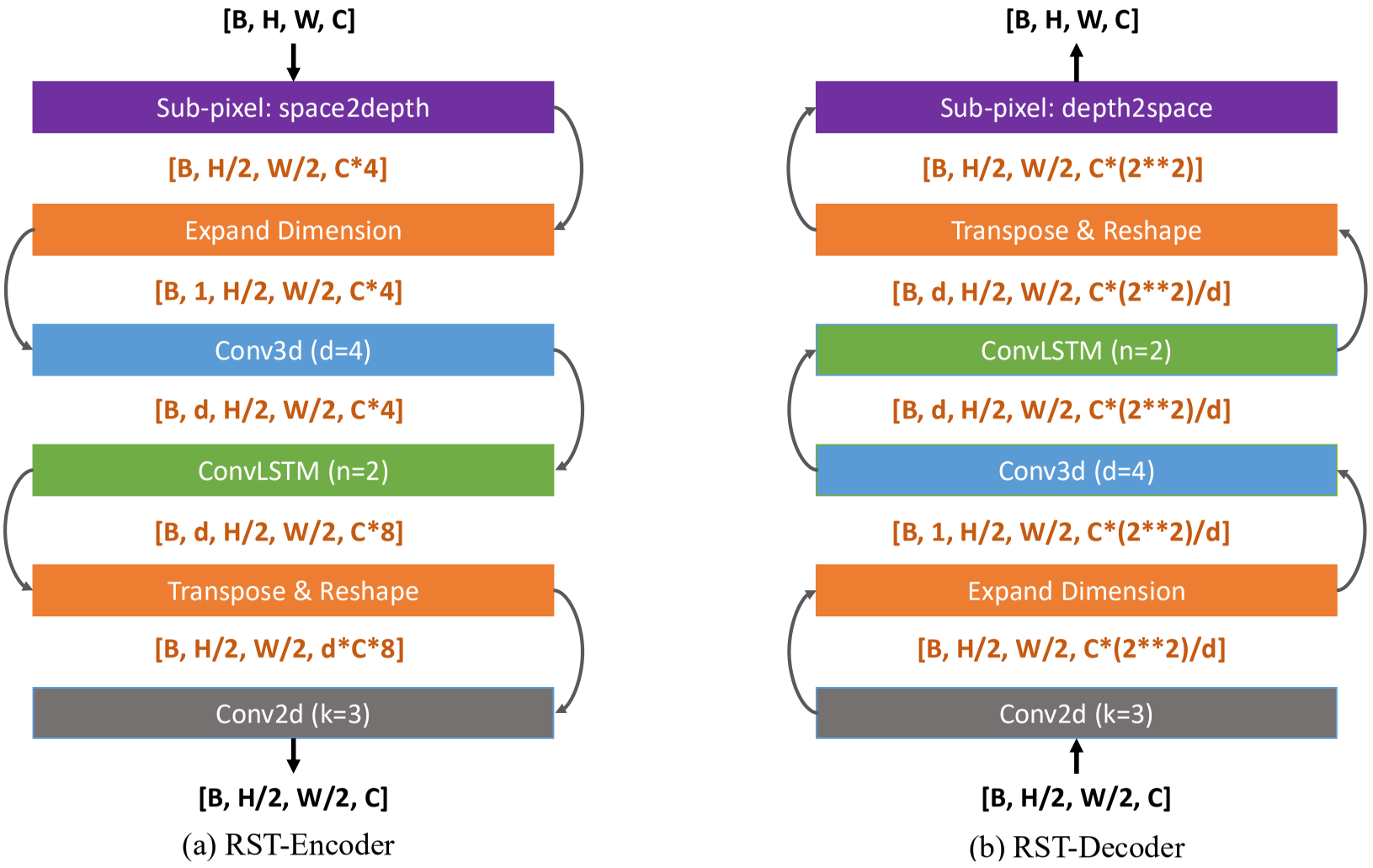

Figure 4.2: Recurrent Spatial-Temporal layers: (a) a single RST-encoder layer and (b) a single RST-decoder layer. The sub-pixel operation, Conv3d, and ConvLSTM in the RST-encoder layer replace the striding and pooling operations normally used in CNNs. They are also symmetrically used in the RST-decoder layer instead of using upsamping operation.

segmentation networks can provide more accurate object information as introduced in [8]. Casser [126] uses segmentation masks as the additional features and estimates the 3D object motion to refine the depth and ego-motion estimations. Gordon [127] goes a further step to implement the unsupervised learning of camera intrinsic parameters and reduce the semantic or instance segmentation input requirements. At the same time, better performance of depth and ego-motion estimations is maintained. Li [128] extends this approach by removing the requirement for any auxiliary semantic information from images. The camera intrinsic can be either specified or learned. Klingner [186] applies the semantic segmentation decoder to predict semantic segmentation in the supervised training stage. After that, semantic masks are used to calculate the loss functions for depth and ego-motion estimations. Wagstaff [187] applies the pose correction to improve the accuracy of ego-motion estimation. Ambrus [188] presents a network that can learn the structure features from additional depth inputs to optimize the ego-motion estimation.

We propose to use the appearance features from input images and intermediate features from the depth and optical flow to estimate the ego-motion in this chapter. The intermediate features from depth are related to the structure or surface of the scenes, and the intermediate features from optical flow are related to dynamic objects.

4.2.3 Learning with Auto-Masking

To further improve the performance of estimation networks, the most intuitive approach is to mask the uncertain or error areas in the training images. For instance, the initial works in [4] propose using the pose network to predict the uncertain mask while estimating the 6-DoF pose. Furthermore, [94] introduces the optical flow to help mask the dynamic objects and occlusion areas. Monodepth2 in [6] proposes the implied auto-masks to improve the training accuracy. To optimize the uncertain masks in [4], [5] presents the self-discovered masks using depth map rigid warping error to mask the uncertain areas. Some research works try to combine the dynamic error to build the training loss functions. Struct2depth in the [126] uses the pre-segmentation semantic masks to mark dynamic objects to estimate their poses and add objects rigid warping into the total loss functions. Gordon in [127] simplifies the segmentation inputs to estimate the unsupervised ego-motion. Jiang [189] tackles the photometric errors by masking out the invisible or manonstationary pixels in the error

map using a statistical technique. Zhao [190] recovers the relative pose by sampling correspondences which are normalized to produce the inlier mask. The residual mask research work shows that both the inlier mask and outlier mask can improve the networks' estimation performance. Chang [191] proposes a neural network block to remove reflection influence for depth estimation.

Based on the auto-mask scheme in [6] which can remove objects moving at similar speeds to the camera and scenes where the camera is temperately static, we deal with the influence of dynamic objects on estimated results by adding mask component via using a depth error. We also use a pre-trained SegNet [3] to detect dynamic objects. In addition, salient object detection from images is an important, challenging vision task such as introduced in [192]. To the best of our knowledge, this chapter is the first to use the difference flow between full and rigid flows for dynamic object detection.

4.3 Recurrent Spatial-Temporal Layers

The traditional convolutional layer pursues enhancing its receptive field size through aggressive striding and pooling operations. However, these operations potentially could lead to the loss of detailed pixel representations, and the resize-convolution operation in upsampling layers could not recover the details. In addition, most convolutional layers do not preserve the temporal features from their inputs. We propose a novel recurrent spatial-temporal encoder layer (RST-encoder layer) that can preserve detailed spatial and temporal features from its inputs to address these problems. We also propose an RST-decoder layer, which is symmetrical to the RST-encoder layer. Both are the building blocks for the encoder and decoder components in our RSTNet presented in the next section.

4.3.1 RST-encoder layer

In Figure 4.2(a), the RST-encoder layer starts with a space2depth operation, which is taken from the sub-pixel network in [177]. This layer can fold the height (H) and width (W) dimensions of convolutional feature maps into extra feature channels ($C * 4$). The output tensor is at a reduced resolution. This operation is a reversible transformation without any loss, which is different from striding or pooling operations. This is followed by a dimension expanding operation, which adds an extra dimension to the tensor for a 3D convolution layer

(Conv3d) with $d = 4$. This extra dimension is used to retain the detailed spatial features. The 3D convolution layer is used to compress the concatenated features and outputs the same size tensor. Then a convolutional LSTM layer (ConvLSTM) consisting of two ConvLSTM cells ($n = 2$) is used to capture temporal features via memory. The ConvLSTM layer outputs the same size tensor with feature channels ($C * 8$). After that, we transpose and reshape the extra dimension (d) of the tensor to the channel dimension ($d * C * 8$), including detailed spatial-temporal features. In the end, a 2D convolution layer (Conv2d) is used to produce the tensor with the desired number of feature channels. This structure of cascading multiple complex convolutional layers allows the RST-encoder layer to preserve detailed spatial and temporal features extracted from the input tensor.

4.3.2 RST-decoder layer

The RST-decoder layer is the symmetrical architecture with cascading multiple deconvolution layers as shown in Figure 4.2(b). The compressed concatenated features from the RST-encoder layer are the input tensor to the first layer (Conv2d). This layer adjusts the number of channel dimensions ($C * 4/d$) to adapt to the following 3D convolutional layer (Conv3d). Then this 3D convolution layer expands back the compressed spatial features to a *ConvLSTM* layer. The *ConvLSTM* layer also consists of two *ConvLSTM* cells to preserve temporal features via memory. The channel dimension in its output is $c * 4/d$. Next, we transpose and reshape the tensor to $d * c * 4/d = c * 4$ channels. This layer ends with a depth2space operation that reduces the channel dimensions to the original C.

4.3.3 RST Layers for Depth Estimation

Our RST-encoder and decoder layers can be used to estimate the depth from consecutive monocular images. To show the capability of our RST layers in recovering lossless spatial-temporal features, we use the RST encoder and decoder layers to replace the pooling, striding, and upsampling layers of the depth estimation network in [4] without changing any other data processing, loss function, and network parameters. The input images are shown in the first column in Figure 4.3. The estimated depth maps from the network in [4] replaced with our RST layers are shown in the second column (RST-SfMlearner). The estimated depth maps from [4] is shown in the third column (SfMlearner). Benefit from the sub-pixel layer, Conv3d, and ConvLSTM layers. These results show that our RST layers can learn more

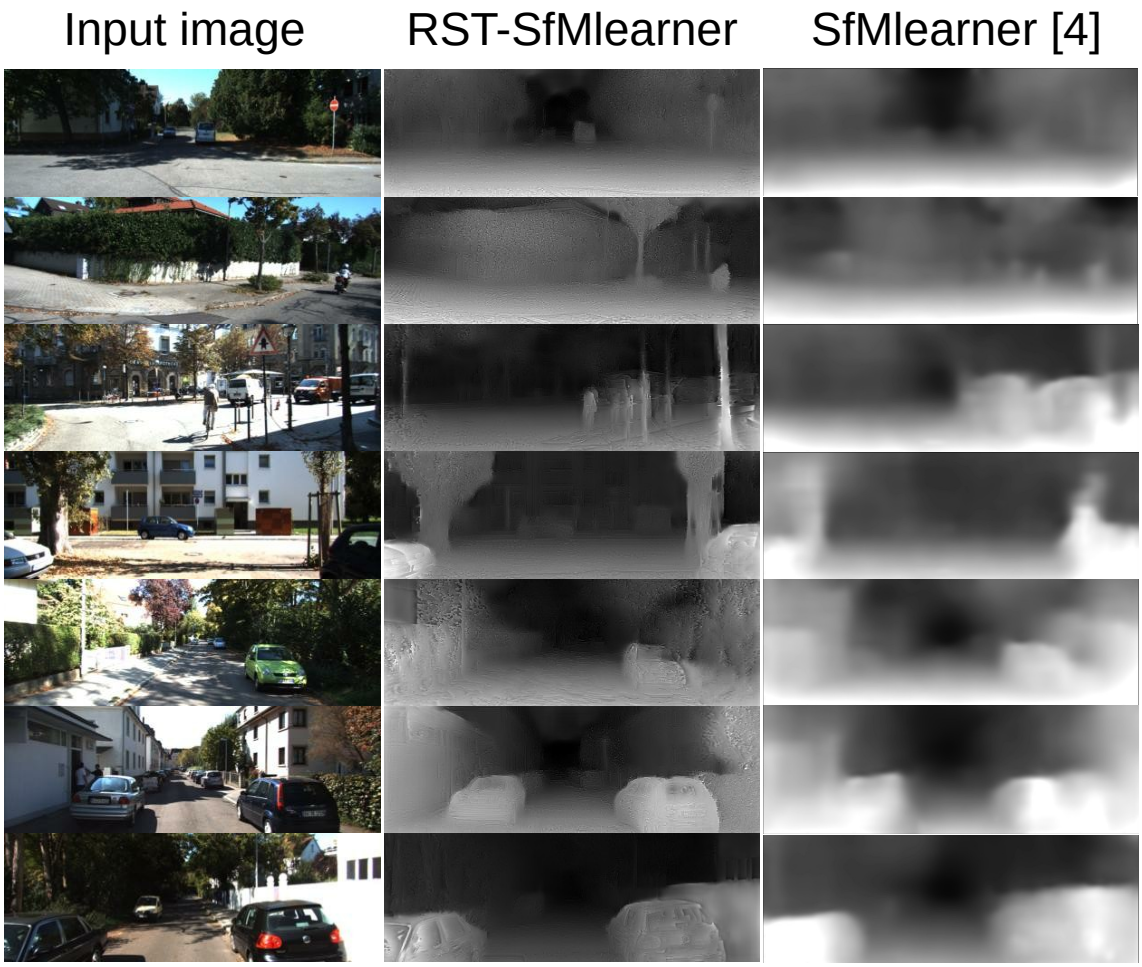

Figure 4.3: Depth estimation performance of our RST layers compared with SfMlearner [4]. The images on the first row show a small and distant view scene. The images on the second, third, and fourth rows show dynamic objects and irregular objects. The images on the fifth, sixth, and seventh rows show high resolution scenes such as dense foliage and car parts.

dynamic structures and detailed spatial features than traditional networks for specific tasks. In Figure 4.3, the depth estimation with our RST layers has more spatial details than the original network, such as road signs, vehicles, and irregular branches. In addition, the results also show that our RST layers perform much better on dynamic objects, such as persons by motorcycle or bike.

4.4 Recurrent Spatial-Temporal Network: RSTNet

4.4.1 RSTNet Overview

Our proposed RSTNet targets four main estimation tasks: depth, ego-motion, optical flow, and dynamic objects. As shown in Figure 4.1, three monocular consecutive images ($Image_{1,2,3}$) are used as its inputs. The RSTNet estimates three corresponding depth maps using three RST encoder and decoder networks. Two PWC-Nets estimate two full optical flow maps ($Flow_{12}$,$Flow_{23}$). It estimates two ego-motions ($R_{12}, t_{12}, R_{23}, t_{23}$) from two RNNs, each of which uses the features extracted from two consecutive monocular images, two estimated depth maps, and one estimated optical flow as the inputs. Two dynamic object maps ($Object_1, Object_2$) are detected from two pre-trained networks (SegNet) [3], each of which uses the difference between a full flow from PWC-Nets and a rigid flow computed from the depth and ego-motion estimations. The RST encoder or decoder layers are used as the building blocks in constructing the networks.

4.4.2 Depth Estimation Networks

Our depth network (Figure 4.4) is constructed with the symmetrical encoder and decoder network structure, including multiple RST-encoder and RST-decoder layers. The monocular image input is a tensor with $[B,H,W,C]$ shape where B is the batch size dimension, H is the height dimension, W is the width dimension, and C is the channel dimension. The first two 2D convolution layers extract the features as 64 output channels. The first RST-encoder layer folds the feature tensor, expands the spatial-temporal features for the next encoding level, and skips symmetric decoding. Starting from the second level of encoding, each of the following encoding layers will shrink with a scale of 2 times to produce the output with more feature channels. Each of the following encoding layers begins with two residual blocks and ends with an RST-encoder layer. The residual block is composed of two stacked 2D Convolution

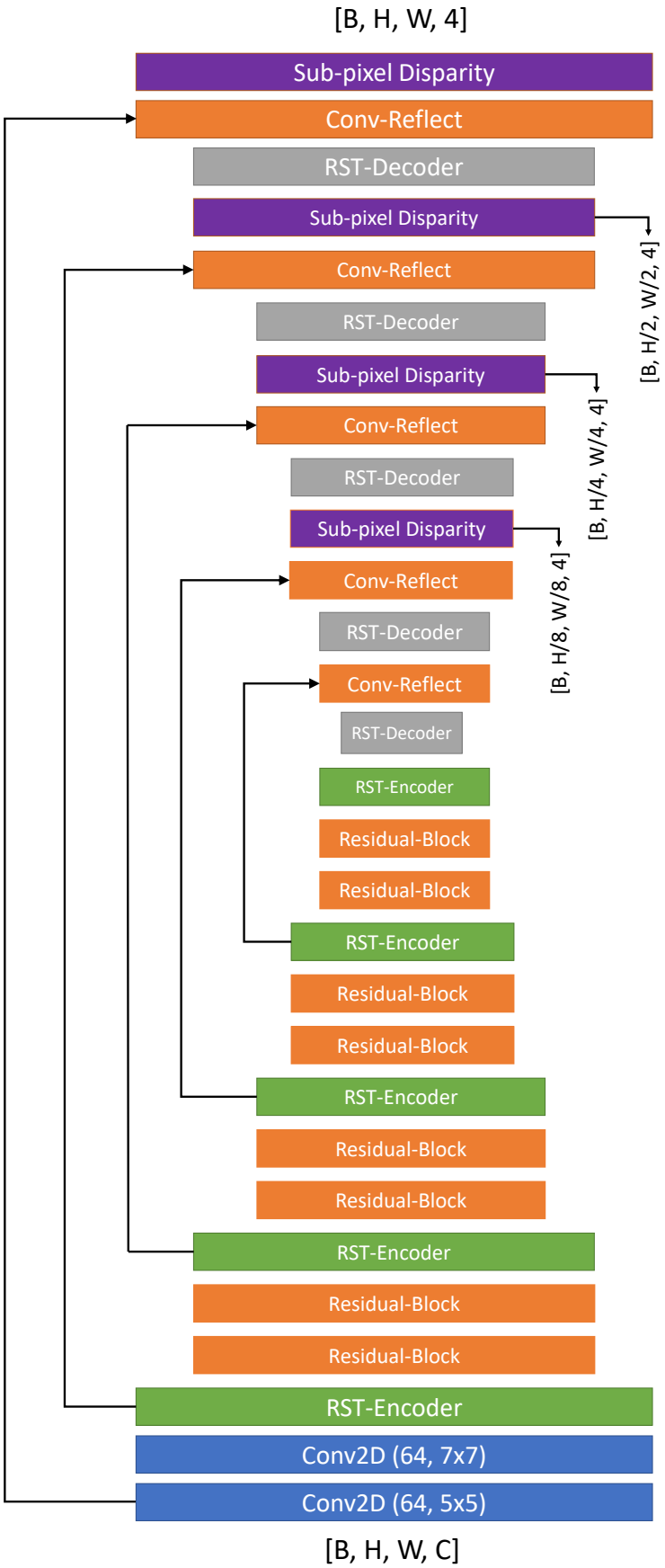

Figure 4.4: Recurrent spatial-temporal networks for depth estimation. This depth network applies the encoder-decoder architecture to extract spatial-temporal features by the RST-Encoder units and express the depth estimation through the RST-Decoder units with Sub-pixel layers.

layers, Batch Normalization [193], and RELU Activation Function [194]. After five RST encoder layers, the height and width of the input tensor are compressed by 16 times.

The decoder part begins with an RST-decoder layer to extend the size of its input tensor to 4 times. Afterward, the Convolution Reflect block combines its output and skip features from the fourth level of the encoder part to decode the tensor. Then each of the following four decoder layers consists of an RST-decoder layer, a Convolution Reflect block, and a sub-pixel disparity layer. Each of them outputs a tensor two times larger in height and width of its input so that the decoder part can obtain the depth maps with four scales. The Convolution Reflect block is a 2D Convolution layer with the half size of filter pad. The sub-pixel disparity layer is the same architecture as the Convolution Reflect block with four output channels. The decoder part of this depth estimation network can produce the disparity maps with four scales $[H/8, W/8]$, $[H/4, W/4]$, $[H/2, W/2]$, and $[H, W]$. Its output resolution on each scale is four times larger than most published depth estimation networks so that it can estimate more details.

4.4.3 Ego-Motion Estimation Network

Depth map carries the structural information of objects, optical flow represents the dynamic information, and original images provide many appearance features in the view. To estimate the ego-motion with reasonable accuracy, we use two adjacent images, their depth maps, and the corresponding optical flow as the source of features extraction.

In the ego-motion network of RSTNet (see Figure 4.1), we reuse the appearance features of two consecutive images extracted in the encoder part of the depth estimation network as inputs. We also use the structural features extracted from two consecutive depth maps as inputs. Furthermore, we use the dynamic features extracted from the estimated optical flow as inputs. The optical flow is estimated from the pre-trained PWC-Net. These features are concatenated on the channel dimension of inputs and fed into an RNN network to estimate the ego-motion. The RNN network consists of two convolution LSTM cells with 256 output channels for decoding the rotation R and the translation t.

Figure 4.5: The first row includes input image (left) and our estimated depth (right). Our auto-mask (μ_d) (left) are compared with the self-discovered mask [5] (right) in the second row. The projected image errors from our RSTNet and [6] are shown in the third row. Our auto-mask $\mu_t + \mu_s$ is shown in the bottom two rows. The vehicle with similar velocity as the camera is shown in the red circle (left) and eliminated as shown in the black (right) in the fourth row. The static distance sky (left) is shown in the fifth row and eliminated as shown in the black (right).

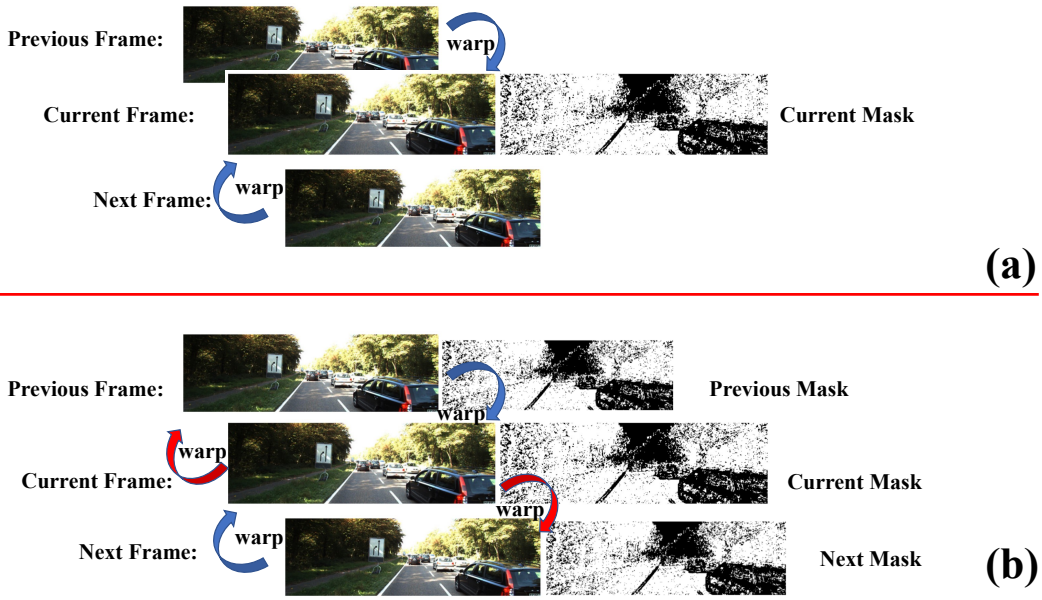

Figure 4.6: Comparison between the Auto-Masking [6] and our Dynamic Auto-Mask.

4.4.4 Loss Functions and Auto-Masking

The synthetic target view $I^i_{s\to t}$ is generated from the ith source view I^i_s by using the spatial transformer [116]:

$$I^i_{s\to t}(x,y) = K\hat{T}_{s\to t}d_t^{-1}(x,y)K^{-1}I^i_s(x,y) \tag{4.1}$$

where K is the camera intrinsic matrix, $d_t(x,y)$ is the estimated disparity, $\hat{T}_{s\to t}$ is the camera coordinate transformation matrix from the ith source frame to the target frame and estimated by the ego-motion network. Our photometric reprojection loss function L_p is defined as below:

$$L_p = \sum_{i=1}^{N} \min_t pe\left(I_t, I^i_{s\to t}\right) + \sum_{i=1}^{N} \min_s pe\left(I^i_s, I^i_{t\to s}\right) \tag{4.2}$$

where N is the number of source views and $N = 2$ for three consecutive input frames. The pe is a photometric reconstruction error computed from $SSIM$ [195] and L_1 [126] [127]:

$$pe(I_a, I_b) = \frac{\alpha \times (1 - SSIM(I_a, I_b))}{2} + (1-\alpha) \times \|I_a - I_b\|_1 \tag{4.3}$$

We also use a depth edge-aware smooth loss function to regularize the depth in texture-

less and low-image gradient regions.

$$L_s = |\delta_x d_t^*| e^{-|\delta_x I_t|} + |\delta_y d_t^*| e^{-|\delta_y I_t|} \tag{4.4}$$

where $d_t^* = d_t / \bar{d}_t$ is the mean-normalized disparity, δ_x and δ_x are the gradient.

Finally, our total loss function includes the photometric reprojection loss with a dynamic auto-masking weight μ and the edge-aware depth smooth loss:

$$L_{total} = \mu L_p + \lambda L_s \tag{4.5}$$

This loss function performs very well for the training under the assumption of static scenes with a moving camera. However, its performance could have deteriorated rapidly when the camera is stationary, or there are dynamic objects in the scene [6]. The auto-masking loss function in [6] alleviates some of these problematic scenes. It can mask the target image to reduce the influence of objects when moving at the same velocity as the camera or ignore the whole image when the camera stops moving. However, it does not exploit dynamic information provided by moving objects.

We propose a novel dynamic auto-masking scheme. The full mask includes three parts $\mu = \mu_t + \mu_s + \mu_d$. The first part μ_t is the same as the one purposed in [6] as shown in Figure 4.6(a).

$$\mu_t = \left[\sum_{i=1}^{N} (\min_t pe\left(I_t, I_{s\to t}^i\right) < \min_t pe\left(I_t, I_s^i\right)) \right] \tag{4.6}$$

This mask can remove the pixels representing other objects moving at a similar velocity as the camera or all pixels when the camera is static.

The second part μ_s plays a similar role as μ_s but with reverse projection $I_{t\to s}^i$ as shown in Figure 4.6(b):

$$\mu_s = \left[\sum_{i=1}^{N} (\min_s pe\left(I_s^i, I_{t\to s}^i\right) < \min_s pe\left(I_s^i, I_t\right)) \right] \tag{4.7}$$

The third part μ_d is defined as a depth error between the estimated depth map d_t and its warped map $d_{s\to t}^i$ from the ith source depth based on optical flow as shown in Figure 4.7:

$$\mu_d = \sum_{i=1}^{N} \left(1 - \frac{\|d_t - d_{s\to t}^i\|}{d_t + d_{s\to t}^i} \right) \tag{4.8}$$

Figure 4.7: Previous Dynamic Auto-Mask and Next Dynamic Auto-Mask with Current Frame Image.

Different from the auto-masks [6] and the self-discovered masks [5], we use a full optical flow to warp a depth map to generate the full dynamic auto-mask. A full optical flow could record all the dynamic information within a scene. The depth map estimated from the depth network relies more on rigid motion in a scene as it is linked to the ego-motion estimation and the rigid warping operation used in the loss functions. The flow warped depth map contains more information on dynamic objects. Their difference reflects more accurately on dynamic objects.

In Figure 4.5, there is a clear dynamic object (motorcycling) in the input image (left on the first row). Our depth estimation also clearly shows the scene (right on the first row). Our auto-mask (μ_d) and the self-discovered mask [5] are compared on the left and right of the second row, respectively. Our mask can provide a much more distinct figure and a thinner outline of the trees than the self-discovered mask. Given a more precise object detection, the mask used in the loss function could avoid the information loss from the input image. The projected image errors from our RSTNet and [6] is shown in the left and right of the third row, respectively. It is demonstrated that our mask detects dynamic objects much better. In the last two rows, the performance of our auto-mask ($\mu_t + \mu_s$) is shown. The vehicle with a similar velocity as the camera is shown in the red circle (left) and eliminated as shown in the black (right) in the fourth row. The static distant sky (left) is shown in the fifth row and eliminated in black (right).

4.4.5 Dynamic Object Estimation

We use a pre-trained network (SegNet) to detect dynamic objects in our RSTNet. The SegNet is an encoder-decoder architecture and used here to classify each pixel as two classes: static and dynamic. The input is the difference between the full flow estimated from PWC-Net and the rigid flow reconstructed by using estimated depth and ego-motion (see Figure 4.1). This

is a different use against the work in [103] [196] where they estimate the residual flow using a network, then reconstruct the full flow with a rigid flow.

Some results from this network are shown as in Figure 4.8. We use an image with a size of 384×184. The forward flow ($Flow_{12}$) and backward flow ($Flow_{21}$) are predicted by the PWC-Net. The rigid flow is computed by using estimated depth and ego-motion. The difference flow is the full flow minus the rigid flow. The dynamic object is separated from the static background by the SegNet (Mask Generator). The result shows the profile of the dynamic object is clearly detected with intermediate optical flows in Figure 4.9.

4.5 Experiments

We implemented the proposed RSTNet using the Tensorflow framework and trained it with one NVIDIA GTX 1080TI GPU. Adam optimizer was employed to speed up the network convergence for up to 30 epochs with parameter $\beta_1 = 0.9$. The learning rate started from 0.0001 and decreased to 0.00001 after the $3/4$ of total iterations. Because of our GPU's memory size limitation, the size of input images was 416×128 under the consideration of comparison with other networks. For data prepossessing, we used different kinds of data augmentation methods introduced in the previous work [4] [6] that can enhance the performance and mitigate possible over-fitting.

4.5.1 Depth Estimation Evaluation

To evaluate the performance of the proposed RSTNet depth estimation, we choose the KITTI dataset with a benchmark split from [4] as the benchmark. As shown in Figure 4.10, we compare our method with the state-of-the-art unsupervised depth estimation methods, such as Monodepth2 [6], SC-SfmLearner [5], and Packnet-SfM [7]. The dynamic, challenging scenes, including pedestrians, are shown in the first and second rows. Our RSTNet performs best compared with the other three methods, such as the shape details for pedestrians. It can also be seen in the third to the sixth rows, where motorcycles and vehicles are dynamic objects. In addition, the RSTNet depth estimation of static objects, such as the trees, street lights, and signposts, has a higher resolution and a clearer view of structures. These results are from a smaller size image (128×416) than others Packnet-SfM (192×640), SC-SfMLearner (256×832), and Monodepth2 (192×640).

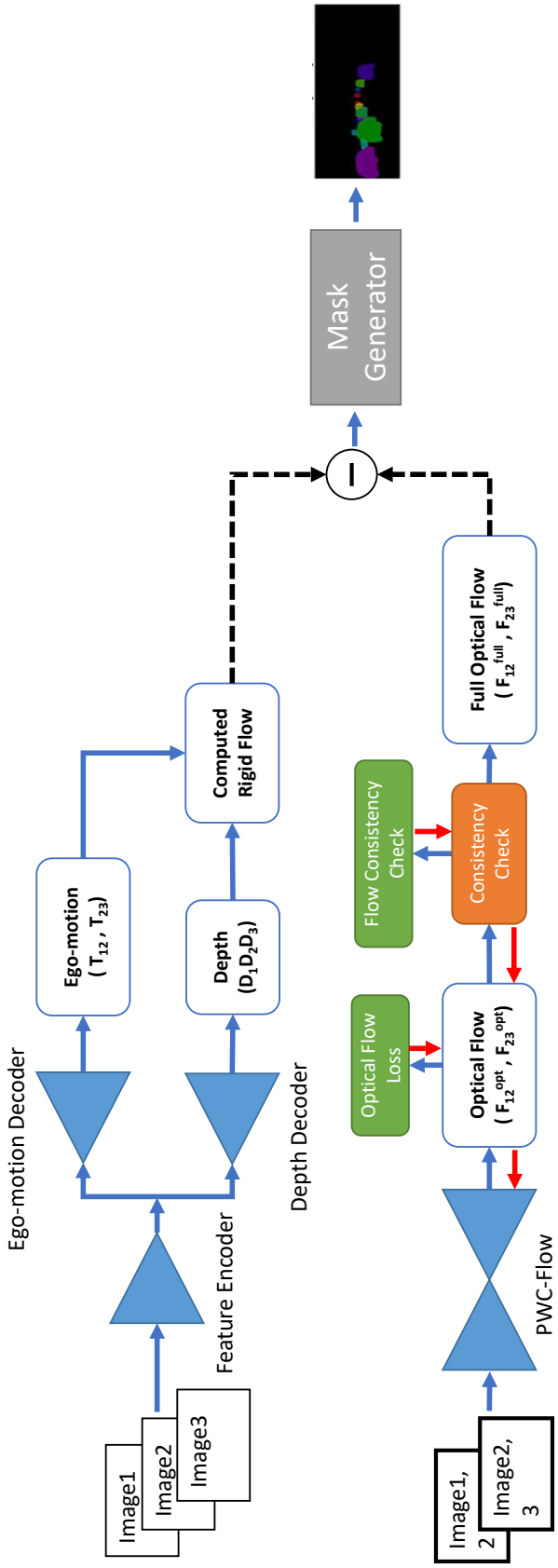

Figure 4.8: Our dynamic objects detection networks.

Figure 4.9: Dynamic object detection in RSTNet. The forward flow $Flow_{12}$ and backward flow $Flow_{21}$ are predicted by the PWC-Net. The rigid flow is computed through the estimated depth and ego-motion. The difference flow is the difference between full and rigid flows. The pre-trained SegNet estimates the dynamic objects.

Figure 4.10: Illustrated above are qualitative comparisons of our RSTNet depth estimation (image size: 416×128) with Monodepth2 [6] (image size: 640×192), SC-SfMLearner [5] (image size: 832×256), and Packnet-SfM [7] (image size: 640×192). The depth map comparisons show that our approach produces qualitatively better dynamic estimates with crisp boundaries and high resolution of details.

We list the quantitative depth estimation results in Table 4.1, where *M* stands for using monocular image sequence for training and *S* stands for using segmentation inputs for training. Compared with the existing state-of-the-art unsupervised learning-based methods, our RSTNet has a better performance in terms of the metrics used (see the smallest values in Abs Rel, Sq Rel, RMSE, RMSE log columns and the highest values δ of the last three columns in the table). Our RSTNet uses the smallest image size but achieves the better depth estimation results without the additional inputs like work in the [126] and [127]. Profit from our lossless RST-encoder and RST-decoder layers, our depth network estimates more accurate results than other methods. Some comparative data of table are obtained from [130] and [7].

4.5.2 Ego-Motion Estimation Evaluation

We used the KITTI dataset to test the performance of the proposed RSTNet ego-motion estimation. The KITTI dataset only provides the ground truth of 6-DoF poses for Sequence 00-10. We used Sequence 00-08 for training and Sequence 09-10 for testing. The metrics are the average translational root-mean-square error (RMSE) drift and average rotational RMSE drift $(^\circ/100m)$ on length of 100m-800m. The results are shown in Table II. We compare the results with SGANVO [130], Zhao [190], ORB-SLAM (without loop closure) [31], UndeepVO [93], and Undepthflow [94]. SGANVO is based on monocular image training. UndeepVO and Undepthflow are based on stereo image training. ORB-SLAM is a geometry feature method. Zhao leverages geometry-based methods and deep learning to estimate the ego-motion. It can be seen that our RSTNet shows a better performance in the testing sequences (09, 10) with these state-of-the-art methods in terms of the translational and rotational RMSE drift metrics. It outperforms other end-to-end learning-based methods (SGANVO, UndeepVO, and Undepthflow). However, the average rotational drifts of learning-based methods (RSTNet, SGANVO, UndeepVO, and Undepthflow) are rifely larger than the geometric methods such as ORB-SLAM and Zhao's [190]. It shows that the deep learning method has a limited learning ability on rotation estimation compared with the geometry methods.

Another metric used is the absolute trajectory error (ATE) averaged over all overlapping 5-frame snippets. We concatenated all of the left and right estimations together for the entire sequences without any post-processing. The estimated trajectories of sequences 02, 08, 09

Table 4.1: Depth estimation results on KITTI dataset using the split of Eigen et al. [70]. For training data, $K = KITTI, CS = Cityscapes$ [119], $M = Monocular$ and $S = Segmentation\ Inputs$. For testing data, our RSTNet uses monocular images.

Method	Datasize	Dataset	Train	Abs Rel	Sq Rel	RMSE	RMSE log	δ <1.25	δ <1.25^2	δ <1.25^3
Garg et al. [86]	620 × 188	K	M	0.169	1.080	5.104	0.273	0.740	0.904	0.962
SfMLearner [4]	416 × 128	K	M	0.208	1.768	6.856	0.283	0.678	0.885	0.957
SfMLearner [4]	416 × 128	CS+K	M	0.198	1.836	6.565	0.275	0.718	0.901	0.960
GeoNet [103]	416 × 128	K	M	0.155	1.296	5.857	0.233	0.793	0.931	0.973
GeoNet [103]	416 × 128	CS+K	M	0.153	1.328	5.737	0.232	0.802	0.934	0.972
Vid2Depth [8]	416 × 128	K	M	0.163	1.240	6.220	0.250	0.762	0.916	0.968
Vid2Depth [8]	416 × 128	CS+K	M	0.159	1.231	5.912	0.243	0.784	0.923	0.970
GANVO [129]	416 × 128	K	M	0.150	1.414	5.448	0.216	0.808	0.939	0.975
Monodepth2 [6]	640 × 192	K	M	0.115	0.882	4.701	0.190	0.879	0.961	0.982
Packnet-sfm [7]	640 × 384	K	M	0.111	0.785	4.601	0.189	0.878	0.960	0.982
Struct2Depth [126]	416 × 128	K	S + M	0.183	1.730	6.570	0.268	-	-	-
Gordon et al. [127]	640 × 192	K	S + M	0.129	1.112	5.180	0.205	0.851	0.952	0.978
RSTNet	416 × 128	K	M	**0.108**	**0.767**	**4.524**	**0.188**	**0.891**	**0.974**	**0.991**

Table 4.2: Ego-motion estimation results on KITTI dataset with our proposed RSTNet. We also compare with four learning methods (Monocular: SGANVO [130], Zhao [190]; Stereo: UndeepVO [93], Undepthflow [94]), and one geometric based methods (ORB-SLAM [31]). RSTNet, SGANVO, and UndeepVO use images with 416×128. Zhao and Undepthflow use 832×256 images. ORB-SLAM use 1242×376 images. The best results among the learning methods are made in bold.

Seq.	Monocular								Stereo			
	RSTNet (416×128)		SGANVO [130] (416×128)		Zhao [190] (832×256)		ORB-SLAM [31] (1242×376)		UndeepVO [93] (416×128)		Undepthflow [94] (832×256)	
	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$
03	**4.67**	**3.08**	10.56	6.30	-	-	0.67	0.18	5.00	6.17	-	-
04	**2.28**	**0.69**	2.40	0.77	-	-	0.65	0.18	4.49	2.13	-	-
05	**2.97**	**1.04**	3.25	1.31	-	-	3.28	0.46	3.40	1.50	-	-
06	**3.81**	**1.13**	3.99	1.46	-	-	6.14	0.17	6.20	1.98	-	-
07	**2.83**	**1.77**	4.67	1.83	-	-	1.23	0.22	3.15	2.48	-	-
09*	**4.35**	**2.22**	4.95	2.37	6.93	**0.44**	15.30	0.26	7.01	3.61	13.98	5.36
10*	**5.54**	**3.27**	5.89	3.56	**4.66**	**0.62**	3.68	0.48	10.63	4.65	19.67	9.13

- t_{rel}: average translational RMSE drift $(\%)$ on length of 100m-800m.
- r_{rel}: average rotational RMSE drift $(^\circ/100m)$ on length of 100m-800m.
- train sequence: 03,04,05,06,07; test sequence: 09*, 10*

Table 4.3: Absolute Trajectory Error (ATE) on KITTI odometry dataset. The results of other baselines are taken from [103]

Method	frames	Sequence 09	Sequence 10
ORB-SLAM(Full) [31]	All	0.014 ± 0.008	0.012 ± 0.011
SfMLearner [4]	5	0.016 ± 0.009	0.013 ± 0.009
GeoNet [103]	5	0.012 ± 0.007	0.012 ± 0.009
Monodepth2 [6]	2	0.017 ± 0.008	0.015 ± 0.010
RSTNet	3	0.011 ± 0.006	0.010 ± 0.005

and 10 from our RSTNet and its ground truth are shown in Figure 4.11 and Figure 4.12. Although the estimated results include some drifts, our RSTNet can estimate all the trajectory features and performs well when no loop closure detection is used. The quantitative results are shown in Table 4.3 where we compare our results with ORB-SLAM(Full) [31], Sfmlearner [4], Geonet [103], and Monodepth2 [6]. In the comparison, our RSTNet produced a good result.

In addition, the rotation error and translation error of sequence 09 and 10 are shown in Figure 4.13 and Figure 4.14 which are based on the 100m to 800m path length and 25km/h to 60km/h speed. From this figure, we can see that it reflects the same drift of translation and rotation as Table 4.2 in four-line charts of the first and third column, and the other four line charts reveal that high-speed movement does not fail the network's estimation even the errors have a nimble reduction in Sequence 09 and 10.

4.5.3 Dynamic Object Estimation Evaluation

The pre-trained SegNet was used for detecting dynamic objects. The performance of using this network for dynamic object detection is shown in Figure 4.15. There are dynamic objects such as vehicles and pedestrians in each input image. Visually, our estimation has good accuracy, and the outline and shape of dynamic objects (car in the first to third rows and pedestrians in the fourth row) are clear. It can be seen that the shadow of the moving car in the second, third, and fourth rows is also detected as the part of dynamic objects, which is a good result given the shadow was also moving with the car.

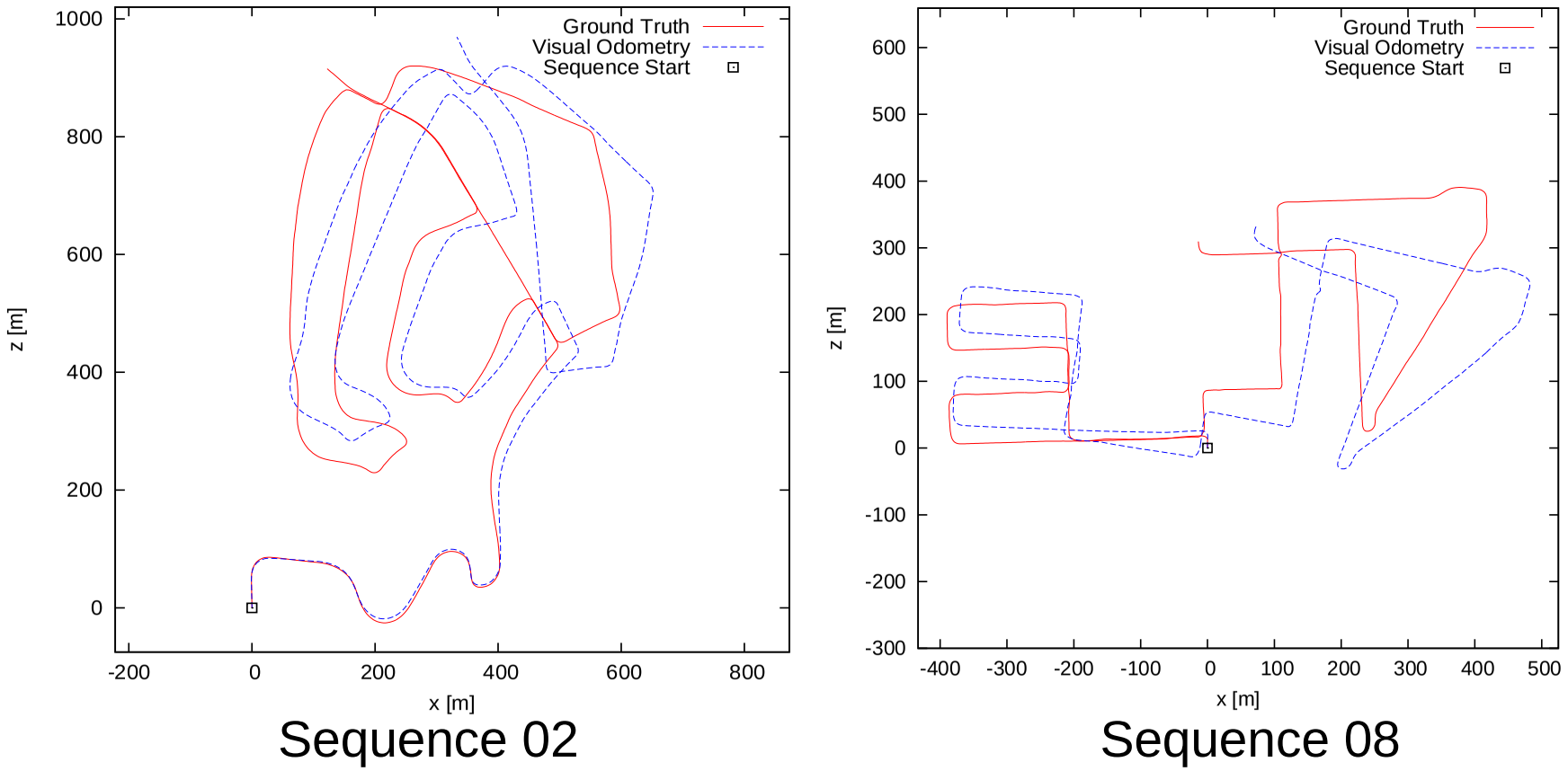

Figure 4.11: Our proposed RSTNet to estimate the ego-motion on the KITTI odometry benchmark, Sequence 02 (left) and Sequence 08 (right). Our results are rendered in blue while the ground truth is rendered in red.

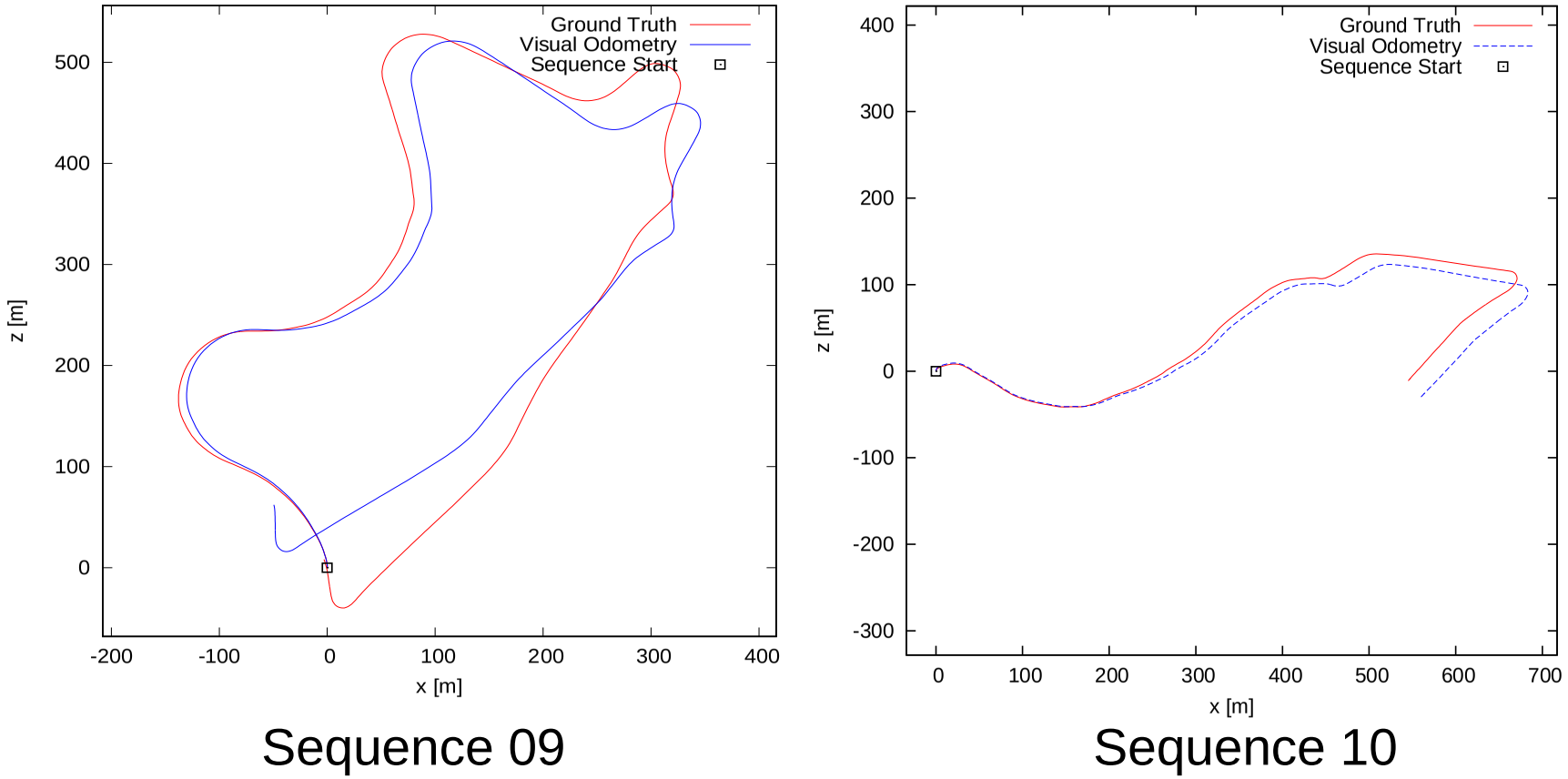

Figure 4.12: Our proposed RSTNet to estimate the ego-motion on the KITTI odometry benchmark, Sequence 09 (left) and Sequence 10 (right). Our results are rendered in blue while the ground truth is rendered in red.

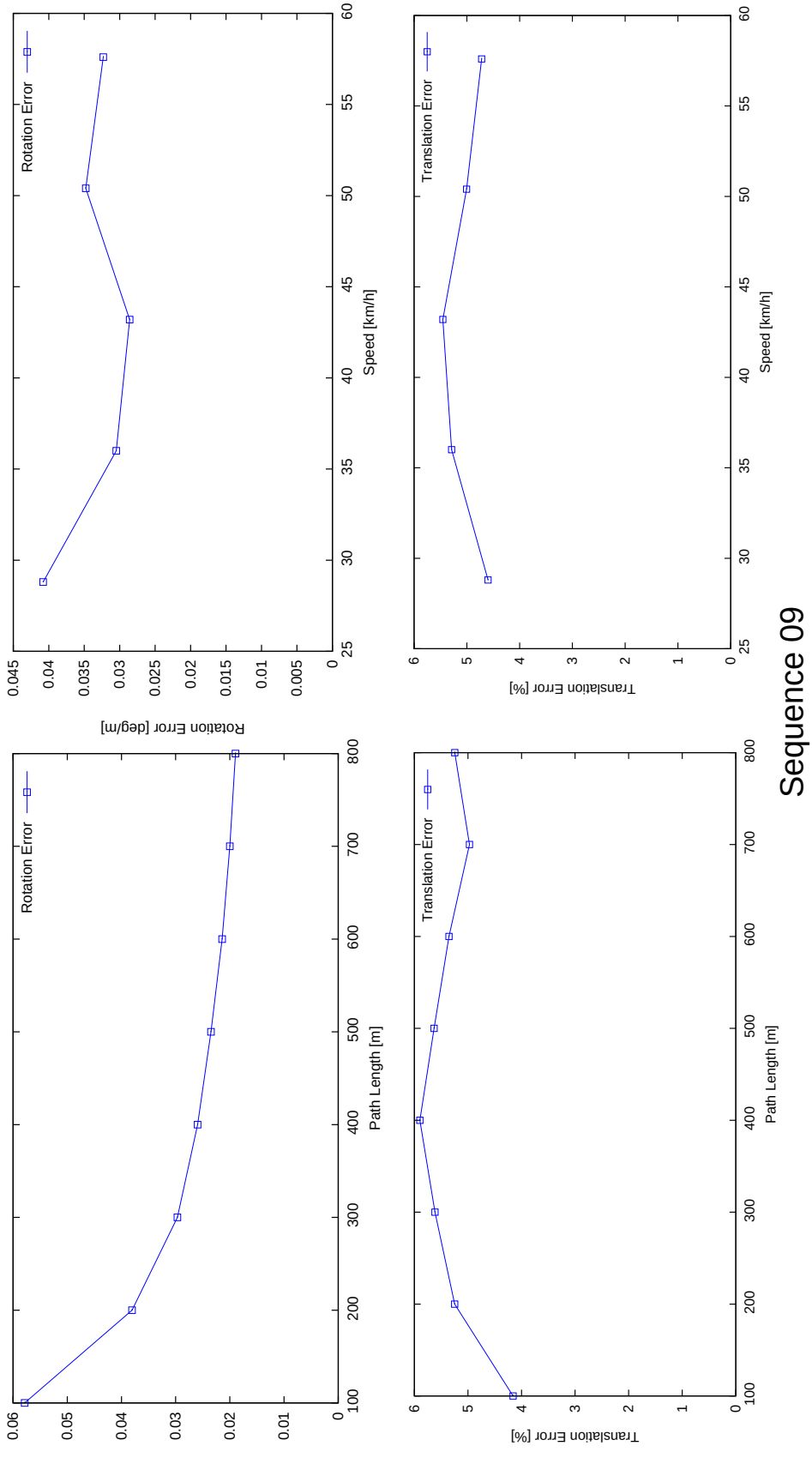

Figure 4.13: RSTNet's Rotation Error and Translation Error based on different Path Length and Speed for Sequence 09. Path Length is distributed 100m to 800m an Speed is 25km/h to 60km/h.

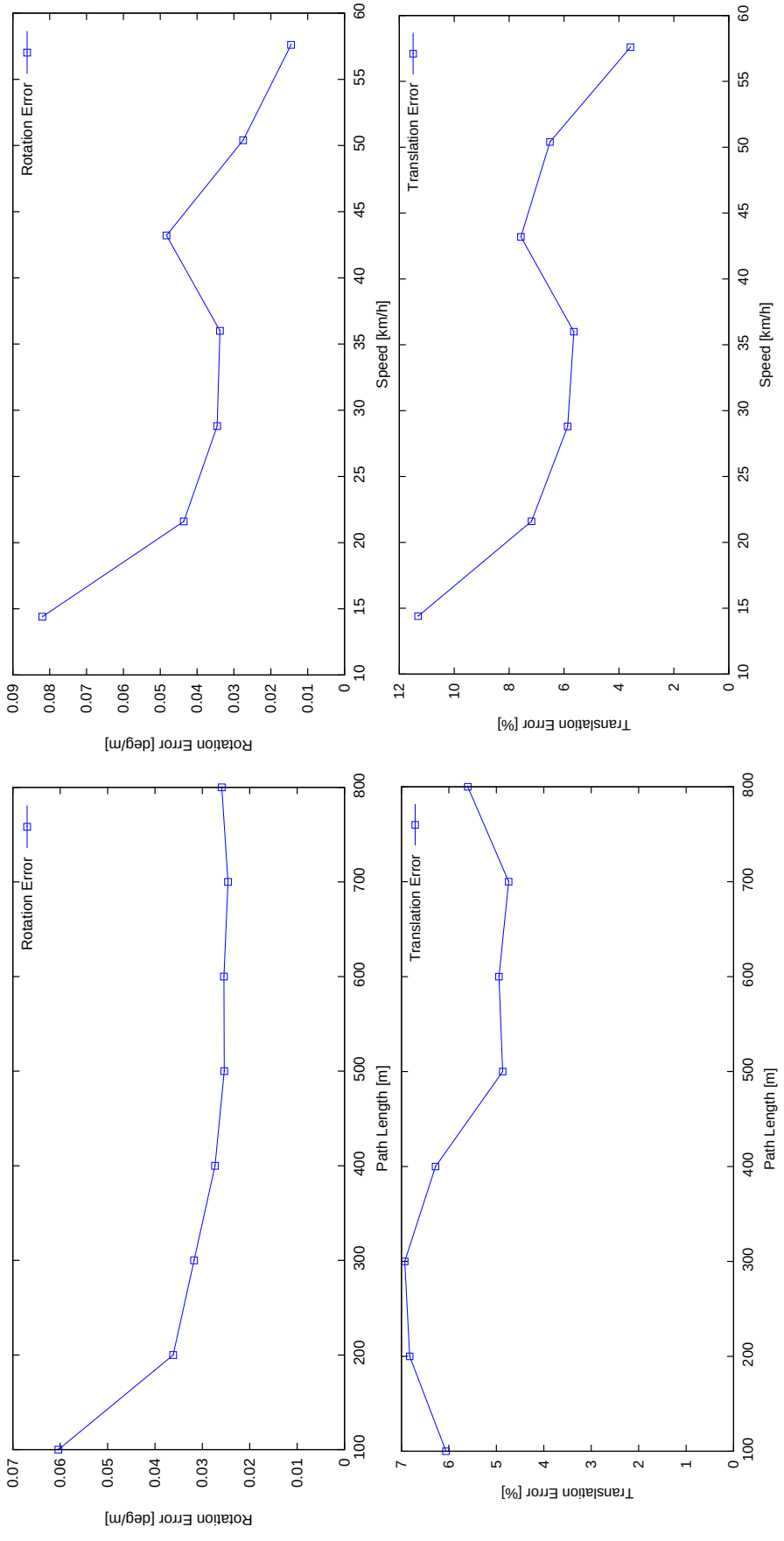

Figure 4.14: RSTNet's Rotation Error and Translation Error based on different Path Length and Speed for Sequence 10. Path Length is distributed 100m to 800m an Speed is 25km/h to 60km/h.

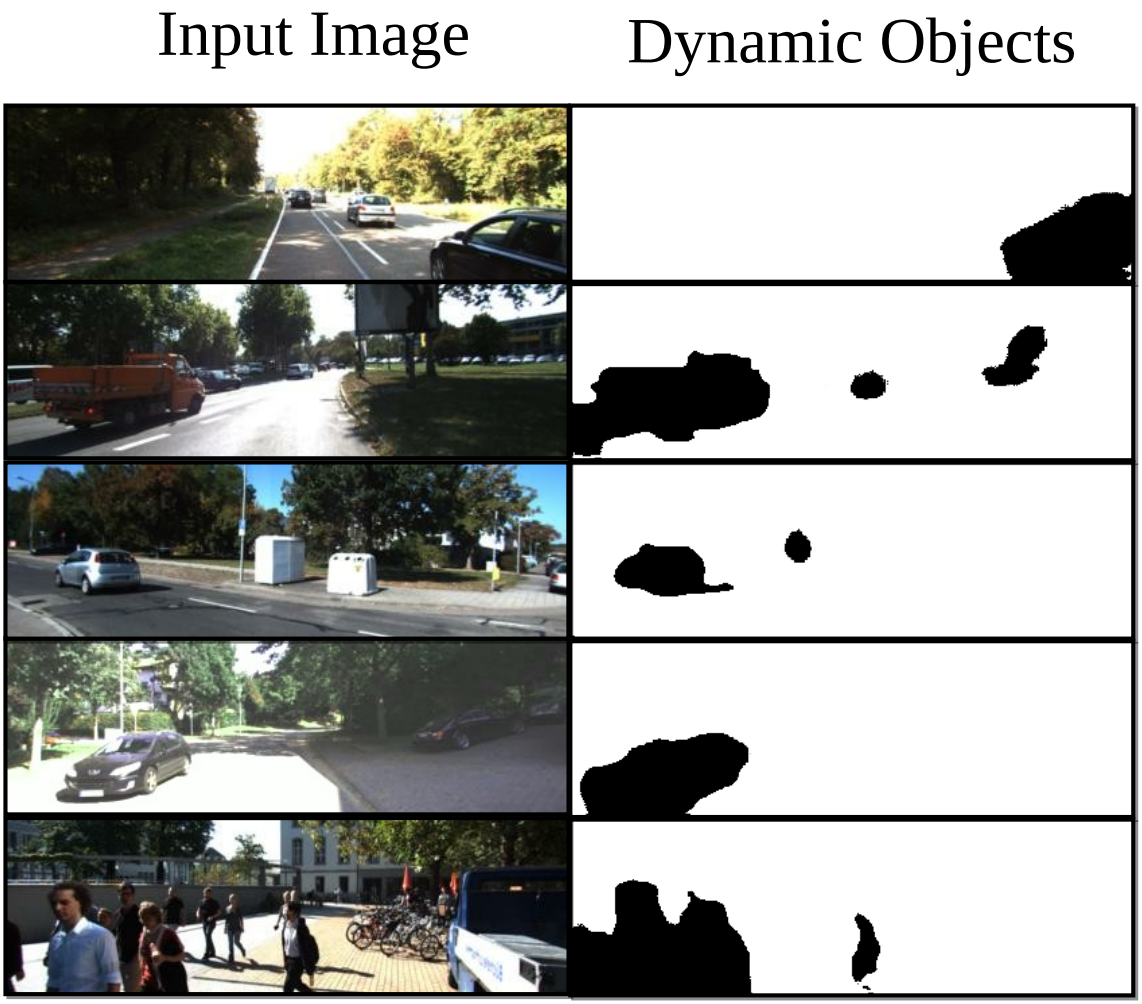

Figure 4.15: Our proposed RSTNet to estimate the dynamic objects on the KITTI odometry benchmark.

Table 4.4: The number of parameters and inference time for different versions of networks.

Networks Type	Parameters	Inference Period(s/it)
3D-Packnet [7]	23170736	1.24357
RST-V1	23125232	1.23886
RST-V2	37062512	1.42846
RST-V3	26444144	1.33599
RST-V4	23789168	1.30512
RST-V5	23125232	1.23902
RST-FULL	40863344	1.89274

4.5.4 Network Complexity

Our RSTNet demonstrated a significant performance in various estimation tasks, such as depth, ego-motion, and dynamic objects. Nevertheless, it comes at the price of computational network complexity. For a fair comparison, our experiments were conducted under Tensorflow 1.14 version with the same size (128×416) for the input image. We implemented our RST-encoder and RST-decoder layers based on the RES18 network. We replaced the striding, pooling, and upsampling operations in each layer of the RES18 network with our RST layers. We constructed five versions of networks (RST-V1 to RST-V5) by replacing each of five filters in the RES18 network with RST layers and a full version (RST-FULL) by replacing all five filters together. We implemented the 3D-Packenet [7] in the same Tensorflow environment. Table 4.4 presents the network parameters and inference time on the NVIDIA GTX 1080TI GPU. The inference time is an average over the last ten iterations in training epoch 2.

Table 4.4 reveals that the number of parameters for version V2 to V4 is larger than the RES18 network, and accordingly longer inference time is required. The full version in the last row shows that both the number of parameters and inference time became significantly larger but produced good estimation results as demonstrated in Figure 4.3.

4.5.5 Ablation Analysis

We conducted three experiments for an ablation analysis with three network versions (RST-V1, RST-V12, to RST-V123). The version RST-V1 replaces the first filter in the RES18

Table 4.5: Performance comparison for different versions of RSTNet. Sq.09 and Sq.10 (ATE) are the ego-motion estimation results. From Abs Rel to δ<1.25^3 are the depth estimation results.

RST Versions	Sq.09 (ATE)	Sq.10 (ATE)	Abs Rel	Sq Rel	RMSE	RMSE log	δ <1.25	δ <1.25^2	δ <1.25^3
RST-V1	0.0123 $\pm$ 0.0078	0.0121 $\pm$ 0.0095	0.1262	1.283	5.214	0.212	**0.860**	**0.957**	**0.981**
RST-V12	0.0118 $\pm$ 0.0070	0.0109 $\pm$ 0.0061	0.1120	1.067	5.024	0.199	**0.863**	**0.961**	**0.984**
RST-V123	0.0116 $\pm$ 0.0066	0.0106 $\pm$ 0.0056	0.0993	0.733	4.623	0.167	**0.903**	**0.969**	**0.988**
RST-FULL	0.0112 $\pm$ 0.0064	0.0101 $\pm$ 0.0053	0.0951	0.673	4.003	0.136	**0.944**	**0.979**	**0.991**

network with an RST layer. The version RST-V12 replaces the first two filters in the RES18 network with two RST layers. The version RST-V123 replaces the first three filters in the RES18 network with three RST layers. In Table 4.5, we set the same network and training parameters to explore the influence of increasing the number of RST layers on the estimation performance. The primary evaluation is focused on the ego-motion and depth estimation for sequence 09 and sequence 10.

It can be seen from Table 4.5 that both ego-motion and depth estimations perform better and better when the number of RST layers increases from 1 to 3. More RST layers lead to more accurate results with a longer computational time, as indicated in the last sub-section.

We also conducted seven experiments for an ablation analysis on features used for ego-motion estimation in our RSTNet. In Figure 4.1, the RNN ego-motion network uses three kinds of input features extracted from two consecutive input images, two estimated depth maps, and one optical flow map. The input image features (F_i) provide appearance information, and they are the most compelling features for ego-motion estimation, as demonstrated in most research works. The depth features (F_d) provide structural information of the scene. The flow features (F_f) provide dynamic information of objects.

The average translational RMSE drift t_{rel} and rotational RMSE drift r_{rel} for Sequence 09 are used for evaluation in Table 4.6. It can be seen the results are not improved when the depth features (F_d) or flow features (F_f) are used alone comparing with image features (F_i). But the results can be improved when the depth features or flow features are used together with image features (see F_{i+d} and F_{i+f} column).

When the combination of depth features and flow features are used (F_{d+f} column), the results get worse, which implies that the image features are the most influential factor. When all three kinds of features are used (F_{i+d+f} column), the results are the best.

4.5.6 Optical Flow Estimation Evaluation

In the above third dynamic object estimation experiments, we utilize the difference between full optical flow and rigid synthetic optical flow. This section evaluates our RSTNet's optical flow estimation, which directly affects the accuracy of dynamic object estimation and auto-mask during the training process. To quickly verify our ideas, the above optical flow network is pre-trained PWC-Net which is supervised training on the FlyingChairs dataset [197]. In-

Table 4.6: Impact of image features (F_i), depth features (F_d), and flow features (F_f) on the ego-motion estimation of Sequence 09.

Features	F_i	F_d	F_f	F_{i+d}	F_{i+f}	F_{d+f}	F_{i+d+f}
$t_{\text{rel}}(\%)$	5.67	8.21	9.12	4.64	4.75	9.06	4.35
$r_{\text{rel}}(^\circ)$	2.94	7.82	8.34	3.25	3.30	7.69	2.22

- t_{rel}: average translational RMSE drift $(\%)$ on length of 100m-800m.
- r_{rel}: average rotational RMSE drift $(^\circ/100m)$ on length of 100m-800m.

spired with Geonet [103] and UnDepthFlow [94], we implement our RSTNet optical flow network unsupervised learning. Because our RST-encoder can be used as a plug-in unit by other networks, we replace the PWC-Net's encoders with our RST-encoders. In addition, our RSTNet can deal well with dynamic information. Our warping optical flow from depth maps and ego-motion perform very close to the real optical flow field. So we can directly use the RSTNet refined PWC-net to estimate optical flow or utilize RSTNet's depth and ego-motion estimations to optical synthesis flow.

In Figure 4.16, our refined PWC-Net can estimate the forward and backward optical flow lying alongside with input image, which has advantages in estimating the dynamic scene field. Our RSTNet synthetic forward and backward optical flow lie alongside a dynamic object image, which estimates dynamic object flow. The experiment results demonstrate that both our refined PWC-Net and RSTNet synthetic optical flow have a good performance in estimating optical flow from images. The application of optical flow from a deep learning network will be introduced in the next chapter with geometry visual odometry.

4.6 Conclusion

This chapter proposes a novel monocular recurrent spatial-temporal network (RSTNet) for estimating depth, ego-motion, and dynamic objects from videos. Since our proposed RST layers can capture detailed spatial and temporal features from consecutive input frames, the RSTNet can generate more accurate depth and ego-motion estimation results compared with other existing unsupervised learning networks. In addition, the dynamic object detection results also demonstrate exemplary performance. The loss function with our auto-mask scheme plays a crucial role in improving the learning quality of depth, ego-motion, and dynamic ob-

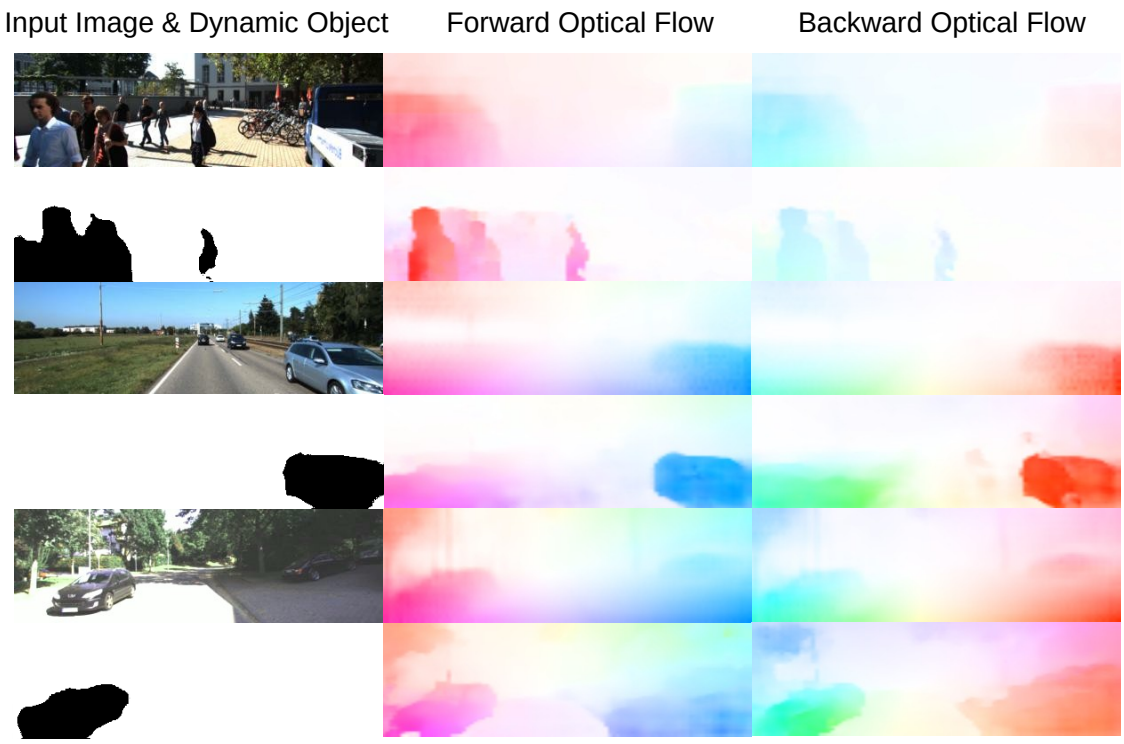

Figure 4.16: Optical flow estimation comparison between the refined PWC-Net estimation and RSTNet synthetic optical flow with dynamic object.

jects.

Many recent works such as [198], [199], and [190] that combines geometric method with deep learning networks can not only improve the accuracy but also reduce computational time. In the future, we will plan to integrate geometric methods with our deep network further to improve accuracy, robustness, and real-time performance.

This chapter explores the novel design of the internal network structure to improve estimation performance. The following chapter will go further to coalesce semantic segmentation into RSTNet for disparity estimation and leverage geometry method for visual odometry.

Chapter 5

Leverage Semantic Segmentation for Depth Estimation and Geometry Method for Ego-Motion Estimation

5.1 Introduction

High-level object segmentation is an essential task for robots to understand their environments. Suppose the learning-based visual odometry can work with object segmentation algorithms. In that case, the system can process higher-level tasks such as object classification and object recognition to build higher-level maps. The segmentation information helps improve the estimation performance of the original tasks in learning-based visual odometry as well. This chapter introduces a pathway for coalescing semantic segmentation networks into the depth and ego-motion estimation networks in section 5.2.

Semantic segmentation is considered to be the most commonly used object detection method in the computer vision field. Instance segmentation can provide more detailed information for different instances in the same category. However, instance segmentation networks need more network parameters to store than semantic segmentation because of more complex network design. Considering the memory limitation of training hardware, this section chooses a relatively nimble semantic segmentation network to coalesce into the visual odometry of our RSTNet.

In addition, to exploit semantic segmentation networks, this chapter also explores in-

tegrating geometry methods into learning-based methods. Although a deep learning-based ego-motion estimator performs well, it still cannot reach the accurate level of some advanced geometry methods. Therefore, more and more research works to leverage deep learning networks and geometry methods to obtain highly accurate results in ego-motion estimation tasks. This chapter implements a geometry-based visual odometry system with deep learning networks in section 5.3.

5.2 Leverage Semantic Segmentation for Depth Estimation

Our RSTNet in Chapter 4 can produce the pretty and legible depth map and relative accurate ego-motion estimation through the complex internal network structure design and utilizing the optical flow network to predict the full flow for building the novel dynamic auto-masking loss functions. It is just this complex and bulky training design that consumes all the current hardware resources in our hands. In this section, the RSTNet is extended to add a high-level object segmentation network rather than using the dynamic objects detection proposed in Chapter 4. This section explores whether semantic segmentation can improve depth estimation or not.

5.2.1 Coalesce Semantic Segmentation into RSTNet

The structure of coalescing semantic segmentation into our RSTNet is shown in Figure 5.1. The middle image of three consecutive frames is fed into the semantic segmentation network to predict the semantic mask. Meanwhile, a simplified version of our RSTNet estimates the depth maps D_1, D_2, D_3 and two ego-motions T_{12}, T_{23} between them. This lightweight RSTNet network only contains one image feature extraction encoder, which is reused by three images for depth and ego-motion decoder as introduced in Chapter 4.

Different from the others' work like [126] [127], we coalesce the semantic segmentation network into RSTNet in one stage training process with one coalescing total loss function. This total loss function contains three components: supervised semantic segmentation loss, unsupervised RSTNet depth estimation loss, and associative semantic edge-ware loss:

$$L_{coal} = \alpha_d L_d + \alpha_s L_s + \alpha_{ase} L_{ase} \tag{5.1}$$

L_d is introduced as the above photometric re-projection loss function. The supervised

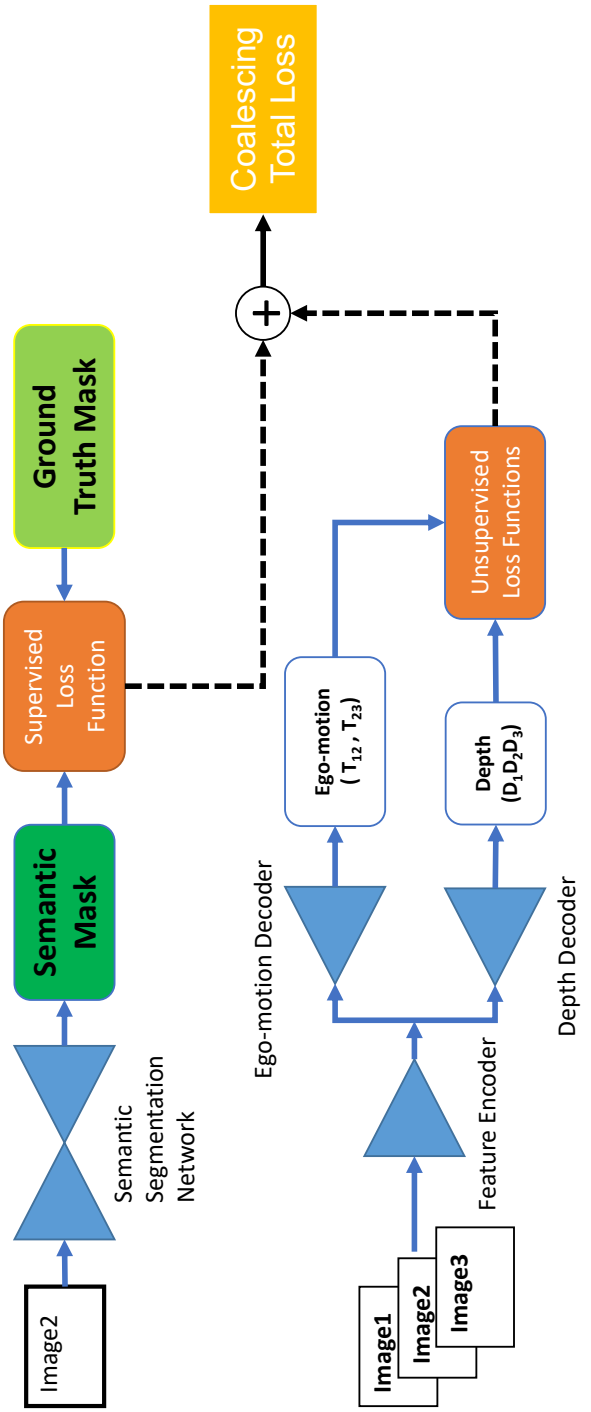

Figure 5.1: Coalesce semantic segmentation into one stage RSTNet training.

segmentation loss function is the standard cross-entropy between the predicted semantic labels and ground truth pixel-wise semantic labels:

$$L_s = H(p_t, \overline{p}_t) + KL(p_t, \overline{p}_t) \tag{5.2}$$

where p_t is the ground truth semantic pixel and $\overline{p}_t$ is the predicted semantic pixel. H donates the entropy, and KL represents KL-divergence. Associative semantic segmentation and depth estimation is L_{ase} as below:

$$L_{ase} = \sum_{i,j}^{N} sign(|\delta_x \mathbf{sem}_{i,j}| e^{-|\delta_x d_{i,j}|}) + sign(|\delta_y \mathbf{sem}_{i,j}| e^{-|\delta_y d_{i,j}|}) \tag{5.3}$$

where $\mathbf{sem}_{i,j}$ is the semantic pixel label, *sign* is the operation that marks the absolute value of the gradients in the semantic map. $d_{i,j}$ is the estimated depth and $\delta_{x,y}$ is the corresponding gradients. This semantic edge-aware smooth loss function can regularize a gradient peak between adjacent pixels belonging to different classes.

5.2.2 Leveraging Semantic Segmentation Depth Evaluation

To keep the better specialties of RSTNet for estimation, the semantic segmentation enhanced version of RSTNet needs to be firstly trained typically as the pre-trained networks. Unlike the unsupervised deep learning of depth, ego-motion, and optical flow, semantic segmentation needs the labeled ground truth for supervised semantic training. The traditional KITTI dataset provides insufficient semantic ground truth labels for training the networks. Most current works use the Cityscapes dataset to train the network to learn semantic segmentation in advance and then transform the pre-trained network to the KITTI dataset for depth and ego-motion estimation. This approach does not incorporate semantic segmentation into the visual odometry networks because the whole networks are not trained in the single-stage at the same time. Benefit from the KITTI segmentation dataset [200], this training dataset can provide the semantic segmentation ground truth labels for supervised training with the unsupervised training for other tasks.

We implement our semantic segmentation-based visual odometry in RSTNet and evaluate the results on the KITTI 2015 dataset. The depth evaluation results of our new system are presented in Figure 5.2. The first column lists the input images. The other two columns are

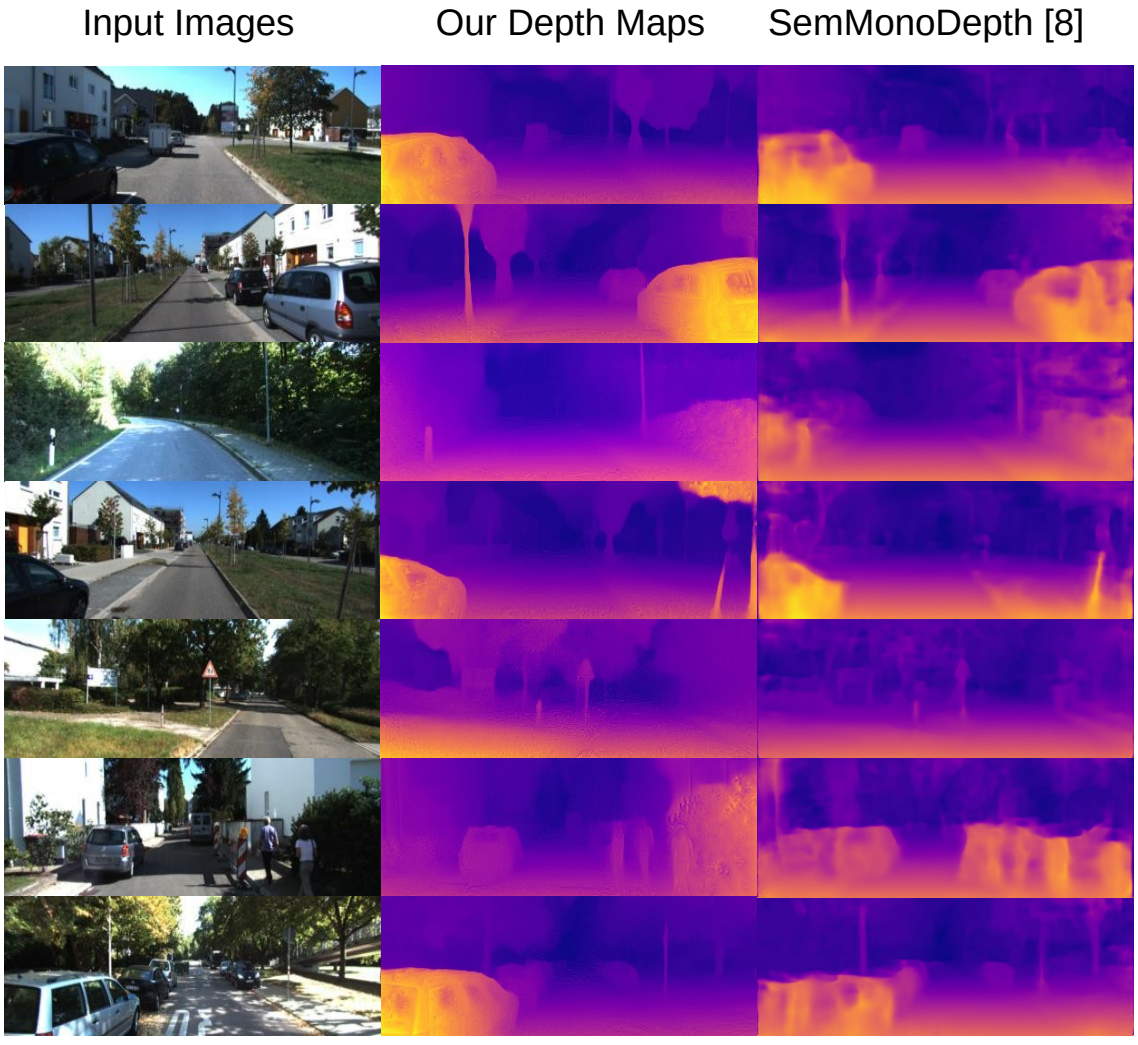

Figure 5.2: Semantic Segmentation based Depth Estimation in RSTNet Compared with SemMonoDepth [8].

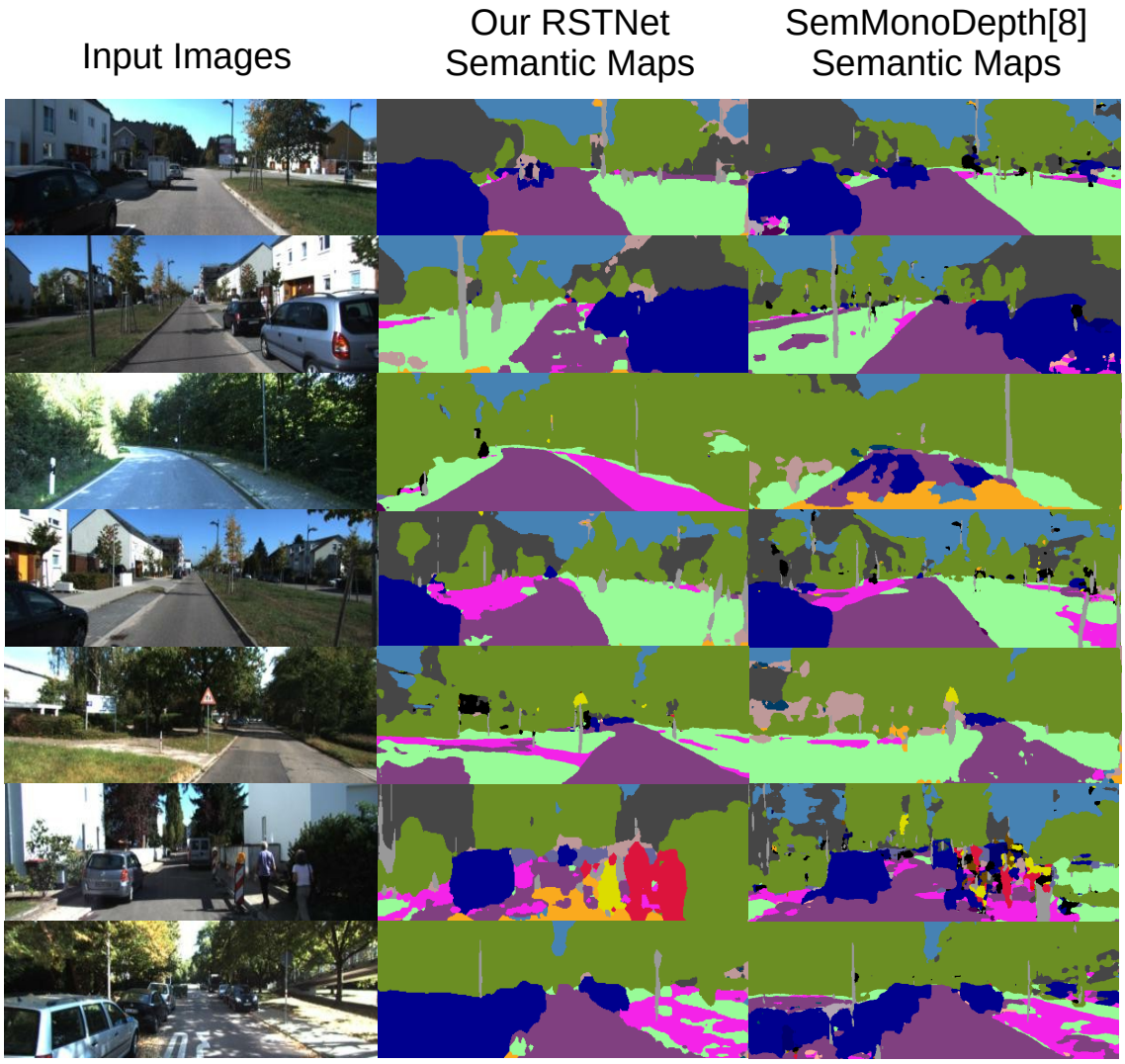

Figure 5.3: Semantic Segmentation Results in RSTNet Compared with SemMonoDepth [8].

Table 5.1: Depth estimation results on KITTI 2015 evaluation dataset. For testing data, our Semantic RSTNet uses monocular images but SemMonoDepth [8] uses stereo images.

Method	Abs Rel	Sq Rel	RMSE	RMSE log	δ <1.25	δ <1.25^2	δ <1.25^3
Semantic RSTNet	**0.127**	**1.112**	**5.176**	**0.202**	**0.858**	**0.953**	**0.979**
SemMonoDepth [8]	0.136	1.872	6.127	0.210	0.854	0.945	0.976

our Semantic RSTNet depth estimations compared with the SemMonoDepth depth estimations. It can be seen that our depth maps are more straightforward and more accurate in the contours of objects such as vehicles, trees, street lamp, and pedestrians. Because of utilizing semantic segmentation ground truth to build the associative semantic edge-ware loss and using our RSTNet unsupervised learning, we can gain better depth estimation results than SemMonoDepth [8] as the quantitative results shown in Table 5.1 even if SemMonoDepth uses stereo images.

From the semantic segmentation evaluation results in Figure 5.3, we can see that our segmentation network can predict the actual object's mask, such as a vehicle, street lamp, tree, and pedestrian. Because of the excellent performance in depth estimation, the edge of the object's semantic segmentation is clear and intact compared with SemMonoDepth [8]. It demonstrates that our semantic segmentation network can infer the intact vehicle in blue, which is closer to the contour shape of the real object. Because our semantic edge-ware loss function couples with depth estimation and semantic segmentation training in one process, good depth estimation results help the semantic segmentation network to have more apparent object edge segmentation and vice versa.

The KITTI semantic segmentation ground truth labels are not enough even that our RSTNet semantic segmentation is also commendable under these hard-limited conditions. We can get more accurate segmentation masks based on more training data with semantic ground truth. In the future, we can extend our RSTNet semantic segmentation on pure high-level object segmentation such as instance segmentation and panoamic segmentation in the computer vision field.

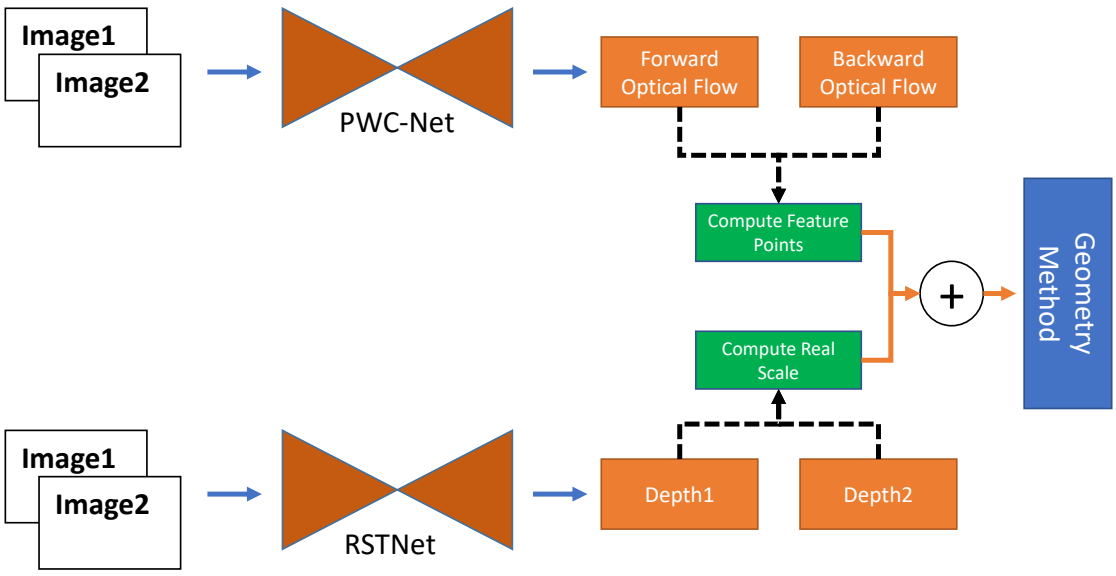

Figure 5.4: Leverage Geometry-based Methods and Deep Learning Networks.

5.3 Leverage Geometry Method for Ego-Motion Estimation

With the rise of deep learning-based visual odometry from 2018, the current learning ability of deep neural networks has stabilized at the bottleneck stage. More and more new research works are emerging to try to make further improvements. One popular approach is proposed to leverage geometry-based method and deep learning networks so that it has the precision and speed of the traditional geometric method and inherits the robustness and stability of the deep learning-based methods. As the future work discussed in the last part of Chapter 4, this section implements leveraging geometry-based methods and our RSTNet based on the framework introduced in [198].

5.3.1 Leveraging Geometry Method and Deep Learning System

- **Deep Learning Component:** As shown in Figure 5.4, this part implements the pre-trained optical flow estimation network based on the PWC-Net between two images. This part only utilizes the refined PWC-Net to detect dynamic objects for optical flow estimation. Meanwhile, the RSTNet depth estimation networks provide two corre-

sponding depth maps. The pre-trained RSTNet's depth estimation can provide accurate depth information for depth scale computing. The purpose of the deep learning component is to replace traditional feature point extraction and represent factual depth information.

- **Geometry Method Component:** Following the deep learning component, the geometry method component utilizes the forward and backward optical flows to predict 2D-2D dense correspondences between images. Give an image pair, (I_i, I_{i-1}), the optical flow describes the pixel movements in I_i which give the correspondences of all the pixels of I_i in I_{i-1}. Inspired by Geonet [103], the optical flow network can predict forward optical flow F_f and backward optical flow F_b. The flow consistency can be computed as:

$$C = -F_f - w(F_f, p_f(F_b)) \tag{5.4}$$

The warping process at a pixel x in backward flow from forward flow F_f is described as:

$$w(F_f[x], p_f(F_b[x])) = F_b[x + F_f[x]] \tag{5.5}$$

After computing the forward-backward flow consistency C, the geometry method component chooses the optical flows with the least inconsistency F_{ic} to form the best N of 2D-2D matches, where N equals 2000 in most experiments. The system forms the feature point matches (p_i, p_{i-1}) from the filtered flows based on flow inconsistency F_{ic} to select the correspondence as introduced in [201]. After selecting good 2D-2D correspondences, the essential matrix can be solved using Epipolar Geometry. The Epipolar constraint is employed for solving the essential matrix: $\boldsymbol{E}$. Then the camera ego-motion $[R,\boldsymbol{t}]$ can be decomposed from the essential matrix $\boldsymbol{E}$ as below:

$$p_{i-1}^T K^{-T} \boldsymbol{E} K^{-1} p_i = 0, \; where \; \boldsymbol{E} = [\boldsymbol{t}]_\times R \tag{5.6}$$

However, the recovered motion $[R,\boldsymbol{t}]$ is up-to-scale. In each frame, the real map scale is computed through the depth estimation based on the geometry. The geometry component uses the predicted depth d_i as a reference for scale recovery. Triangulation is performed for (p_i, p_{i-1}) to recover the up-to scale depth d_i'. A scaling factor s can be

estimated by aligning the triangulated depth map $d_i^{'}$ with the deep learning depth map d_i. Thus, the scaled ego-motion can be calculated as $[R, s \times \boldsymbol{t}]$.

In summary, the deep optical flow learning network outputs forward and backward optical flow estimations to detect the flow inconsistency for feature point extraction. The geometry method for 2D-2D matching computes the camera ego-motion for visual odometry. The deep depth estimation network provides the 3D structure reference for the geometry method's triangulation scale recovery. These two components are complementary.

5.3.2 Leveraging Geometry Method Ego-Motion Evaluation

The performance of the fusing method visual odometry is shown in Figure 5.5. The left big black area is the visual odometry map. The red track is the ground truth and the green track is our result. It can be seen that the leveraging method has less drift on visual odometry estimation than the learning methods in previous chapters. The first image on the right presents the feature points extracted by comparing the forward optical flow and backward optical flow. The two consecutive images matching results are shown in the second row on the right. The depth map is listed in the third row, which is used to compute the actual scale of the map with the forward optical flow. The two bottom images are the backward optical flow and flow difference map.

Because our system utilizes the geometry-based method to estimate the ego-motion, which is based on the feature point extraction from the forward-backward optical flow matching, the track is more accurate without significant drift like pure deep learning-based methods. For quantitative comparison, we compare the leveraging geometry method with the RSTNet ego-motion estimation in the pure deep learning method in Table 5.2. Concerning translation error or rotation error, the leveraging geometry method outperforms the pure learning method. The translation and rotation drifts of the leveraging method are small, so the shape of the trajectory curve is closer to the ground truth curve corresponding to the visualized results in Sequence 09 of Figure 5.5 and Figure 4.12. Nevertheless, the leveraging geometry method performs better than the pure deep learning method in the rotational drift comparison. The leveraging geometric method performs nearly an order of magnitude better than the learning method on average. We can also get a visual comparison in Figure 5.5 which has tiny rotational drift instead of significant rotational drift for Sequence 09 in Figure

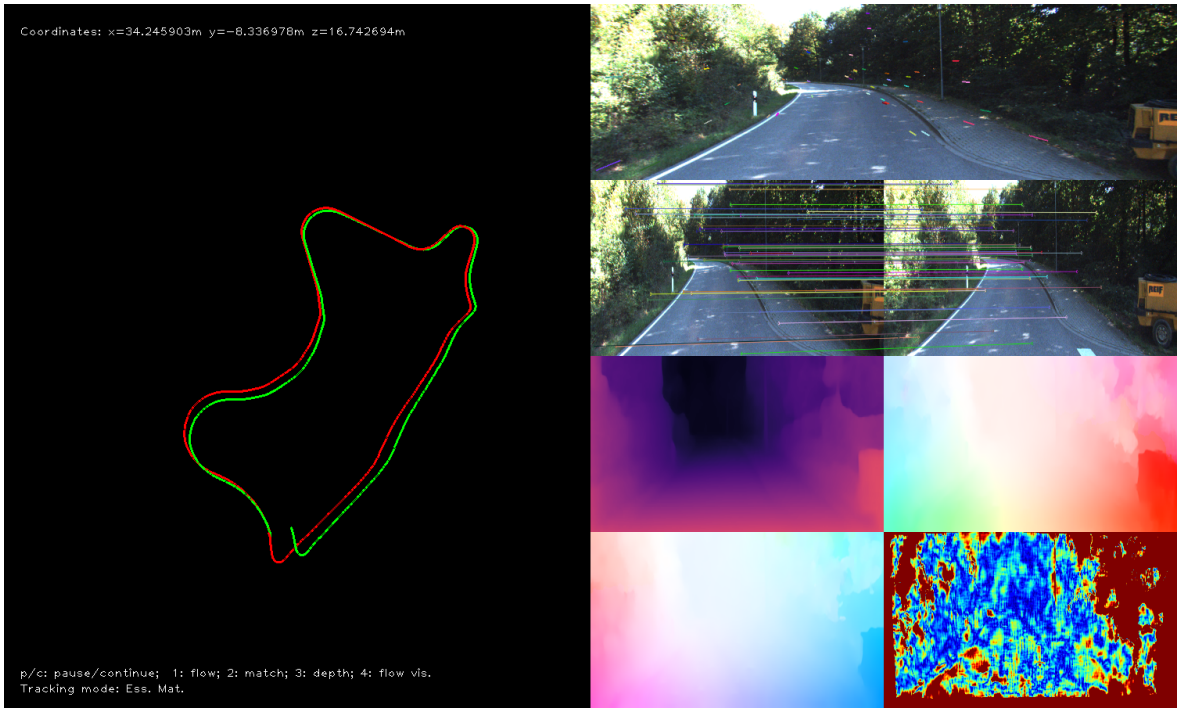

Figure 5.5: Leverage Geometry-based Method and RSTNet's Map on KITTI Odometry Sequence 09.

4.12.

5.4 Conclusions

To sum up, this chapter serves as a supplement and extension to the previous RSTNet according to depth and ego-motion refinements. This part implements the leveraging semantic segmentation into the RSTNet depth estimation and leveraging geometry-based method on the ego-motion estimation. For the fairness and comparability of the experiments, we have to implement the semantic segmentation network on the same RSTNet visual odometry data set. Considering the hardware memory limitation, we re-implement our RSTNet in a lightweight version for exploring the semantic segmentation's improvement of depth estimation. Although there is a limitation on KITTI semantic segmentation ground truth data, we coalesce the semantic segmentation into RSTNet, and its performance is reasonably good. In the future, we can extend the semantic segmentation to large semantic data sets for further improvement.

Furthermore, we leverage the geometry-based method and deep learning networks to estimate the ego-motion track map following the current research diversion. The ego-motion

Table 5.2: Ego-motion estimation results on KITTI odometry dataset from Sequence 03-07 and 09-10 with our proposed RSTNet. We compare our leveraging geometry method with the RSTNet's ego-motion estimation. The best results are made in bold.

	Sequence	03	04	05	06	07	09	10
RSTNet	$t_{\text{rel}}(\%)$	4.67	2.28	2.97	3.81	2.83	4.75	5.54
	$r_{\text{rel}}(^\circ/100m)$	3.08	0.69	1.04	1.13	1.77	2.22	2.72
Leveraging	$t_{\text{rel}}(\%)$	**2.26**	**1.47**	**1.53**	**1.35**	**0.69**	**2.48**	**1.89**
Method	$r_{\text{rel}}(^\circ/100m)$	**0.39**	**0.33**	**0.27**	**0.42**	**0.30**	**0.28**	**0.39**

- t_{rel}: average translational RMSE drift $(\%)$ on length of 100m-800m.
- r_{rel}: average rotational RMSE drift $(^\circ/100m)$ on length of 100m-800m.

estimation of the fusing method has a significant advantage compared with the pure deep learning method. Through our evaluation experiments, we can see that the leveraging geometric method has high accuracy. Therefore, the leveraging geometry and deep learning method achieve both advantages, such as robustness and accuracy requirements. In the future, we can explore more lightweight deep learning networks for feature point extraction such as depth maps to get accurate, robust, and real-time performance.

The above three chapters are my main work and contributions. The next chapter is the last part of the thesis to discuss the conclusions and future work.

Chapter 6

Conclusions and Future Work

The final chapter presents a summary of my research work and outlines the significant research contributions. In addition, this chapter also enumerates the journal and conference publications of my research work and extends to discuss the future work.

6.1 Research Summary

My Ph.D. research is irritated by sensor-less challenging navigation tasks for mobile robots with videos only from the camera. Visual simultaneous localization and mapping (VSLAM) techniques are a solution to these challenging navigation tasks. Remarkably, my work focused on the initial and vital component of VSLAM: visual odometry (VO), which directly deals with feeding images as the front end of VSLAM systems.

The traditional visual odometry is based on the geometry-based method by estimating the rigid ego-motion between two points by the positions in three-dimensional space. The relationship between them is the artificial, rigid transformation. Deep learning technology extends another path to estimate ego-motion through training deep neural networks. Unlike the traditional geometric method or sensor's measuring depth, the depth or the ego-motion is the assumed known as the deep learning network's initial output, although they are wrong. Then the network is trained on the amount of data converging them towards the target output through minimizing the loss functions with the supervised ground truth labels or the unsupervised view synthesis images. No matter what methods are, there are three primary variables of visual odometry (depth information, ego-motion, and optical flow) to settle. Therefore, my research target is to estimate them by deep learning technology and process high-level

semantic segmentation tasks.

In learning-based methods for visual odometry, depth and ego-motion can support each other via the design of loss functions and network architectures. There are stereo and monocular depth estimation methods. Some datasets provide the depth and ego-motion ground truth, and some do not. There are supervised and unsupervised training methods. After comparing the previous works, we decided to explore the harsh condition under which the monocular depth and ego-motion could be estimated with unsupervised learning methods.

Inspired by related work and popular generative adversarial network (GAN), we propose the SGANVO based on the stacked GAN to estimate the depth and ego-motion at the bottom layer. The state-of-the-art performance of estimations is achieved compared with the previous work. However, the SGANVO is too bloated and contains a considerable size of networks which costs too much memory of GPU with long computation time. To make the visual odometry network more nimble, we explore the internal structure of the networks to improve the nimble network learning ability of spatial-temporal features. In this condition, we successfully design the recurrent spatial-temporal (RST) layer, which records the spatial-temporal features.

Furthermore, we propose the efficient RSTNet to estimate the depth maps and ego-motions for three consecutive input images each time. From the evaluation of the baseline dataset, the RSTNet can estimate the more accurate depth and ego-motion with a smaller network scale. In addition, we refine the PWC-Net by the RST layer to estimate optical flow to detect dynamic objects and mask the dynamic information during the network's training process.

At last, we expand our RSTNet to deal with semantic segmentation and also leverage geometry-based methods. From the experiment results, our RSTNet performs well on the semantic segmentation task. After replacing the geometry-based feature extraction with our deep learning network, our visual odometry has a broad practical prospect with state-of-art performance.

6.2 Research Contributions

The significant contributions of my research are briefly outlined as follows:

(1) **Stacked Generative Adversarial Networks based Visual Odometry (Chapter 3)**

- Propose a novel stacked generative adversarial network to estimate the depth and ego-motion to reconstruct the fake image with the previous input image for gaming with the actual input image.
- The upper stacked structure of the networks can learn the temporal features of three consecutive frames for estimation tasks.
- The training dataset chooses the stereo images to learn to rebuild the stereo image, making the depth estimation more accurate. The trained SGANVO can work on the monocular dataset.

(2) **Recurrent Spatial-Temporal Encoder and Decoder Layers (Chapter 4.3)**

- Design a novel RST layer with 3D convolution space expansion and LSTM temporal extension, which records the spatial-temporal features in one time step.
- Integrated sub-pixel layer into the RST-encoder layer and RST-decoder layer to achieve a higher precision output.
- The RST-encoder and RST-decoder layers can be flexibly used as plug-in units by other networks.

(3) **Recurrent Spatial-Temporal Network for Estimating Depth, Ego-motion, and Dynamic Object (Chapter 4.4)**

- Propose a novel depth estimation network with an encoder-decoder structure consisting of multiple RST-encoder and RST-decoder layers.
- Feed the different features from images, depth maps, and optical flow into an RNN to estimate the ego-motion.
- Propose the auto-mask to move out the relatively static and dynamic information for refining the network's training process.
- Utilize the difference between the full optical flow from pre-trained PWC-Net and the rigid optical flow synthesized from depth and ego-motion estimations to estimate dynamic objects through the pre-trained SegNet.

(4) **Semantic Segmentation and Geometry Method are coalesced with the RSTNet (Chapter 5)**

- Coalesce high-level semantic segmentation into the RSTNet depth and ego-motion estimation training in a single stage.
- Leverage the geometry-based feature extraction by the RSTNet optical flow network.

6.3 Academic Publications

Some academic publications have been published or submitted from my Ph.D. research.

Journals

- Tuo Feng, Dongbing Gu. "SGANVO: Unsupervised deep visual odometry and depth estimation with stacked generative adversarial networks." IEEE Robotics and Automation Letters 4.4 (RA-L) (2019): 4431-4437.
- Tuo Feng, Dongbing Gu. "RSTNet: Recurrent Spatial-Temporal Networks for Estimating Depth, Ego-Motion, and Dynamic Objects." IEEE Transactions on Cybernetics. (T-Cyb), 2021. (Under Review)

Conference

- Tuo Feng, and Dongbing Gu. "SGANVO: Unsupervised deep visual odometry and depth estimation with stacked generative adversarial networks." IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019.

6.4 Potential Applications

SGANVO and RSTNet perform well in the laboratory, but real-time requirements are sacrificed in pursuit of accuracy and performance, and extensive networks require a lot of computing time. These two networks can work on a solid computing power platform such as cloud service and vehicle for real-time applications. If applications' real-time requirements are not high, our networks can also run offline.

In the future, we can also convert the network to a lightweight version such as MobileNets [202] to be competent for real-time applications of computer vision platforms with limited resources.

6.5 Future Work

Based on current research in the visual odometry area, three main directions are worthy of further exploration: robustness, accuracy, and real-time.

- **Robustness** Deep learning technology provides a pathway to estimate the track for visual odometry robustly. Future work can expand the RSTNet on the same dataset but in different seasons or periods to train the network's robustness.

- **Accuracy** Fusing the geometry-based method and deep learning can gain both accurate and robust benefits. The next step is to explore the feature points extracted from different networks, such as the RSTNet depth network and the RSTNet ego-motion network. In addition, the output of the ego-motion network can be used as the initial estimation, which can be refined by the geometry-based method as residual adjustment.

- **Real-time** Design smaller and more efficient networks to reduce the computation time or replace ego-motion estimation with a real-time geometry-based method.

According to high-level segmentation tasks, three new fields are worthy of future research: instance segmentation, panoptic segmentation, and dynamic object segmentation.

- **Instance Segmentation** Instance segmentation is a more detailed task to detect different objects belonging to the same class. Therefore, the network needs to be added with new layers and be trained on the instance ground-truth dataset.

- **Panoptic Segmentation** Panoptic segmentation can be combined with semantic segmentation or instance segmentation. This task can process by one stage from a single designed network like [203] or two stages from respective networks.

- **Dynamic Object Segmentation** Our RSTNet only deals with the dynamic object estimation but cannot detect different object classes and instances. Benefit from semantic segmentation and instance segmentation. Future work can detect different dynamic object instances.

When visual odometry is integrated into an entire SLAM system, the rest should include loop closure detection, bundle adjustment, and 3D mapping.

- **Loop Closure Detection** The estimation of location is a recursive process in which the pose of the current frame is calculated from the pose of the previous frame. So the cumulative error or drift of estimation is transmitted frame by frame in visual odometry. Finding out the historical frames that can establish this location constraint can be achieved using the loop closure detection technique, which helps to reduce the cumulative error. In the future, our RSTNet can add the loop closure detection at the back end.

- **Bundle Adjustment** Bundle adjustment is another optimization operation when keyframes are needed to reduce the drift. This operation also can be added to the front end.

- **3D-Mapping** Building a dense 3D point cloud map becomes more and more popular in practical applications. Future work can build the 3D point cloud map through our RSTNet depth map estimation. In addition, subsequent step work can build a 3D point semantic map combined with semantic segmentation.

To sum up, the thesis research proposes some innovative deep learning networks for estimating depth, ego-motion, optical flow, dynamic objects, and semantic segmentation for visual odometry. The state-of-art performance has been achieved compared with related work. It contributes to further progress in the area of SLAM research.

Bibliography

[1] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised Monocular Depth Estimation with Left-Right Consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 270–279.

[2] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8934–8943.

[3] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[4] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised Learning of Depth and Ego-Motion from Video," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1851–1858.

[5] J.-W. Bian, Z. Li, N. Wang, H. Zhan, C. Shen, M.-M. Cheng, and I. Reid, "Unsupervised Scale-Consistent Depth and Ego-Motion Learning from Monocular Video," *arXiv preprint arXiv:1908.10553*, 2019.

[6] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into Self-Supervised Monocular Depth Estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3828–3838.

[7] V. Guizilini, R. Ambrus, S. Pillai, A. Raventos, and A. Gaidon, "3D Packing for Self-Supervised Monocular Depth Estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2485–2494.

[8] P. Z. Ramirez, M. Poggi, F. Tosi, S. Mattoccia, and L. Di Stefano, "Geometry Meets Semantics for Semi-Supervised Monocular Depth Estimation," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 298–313.

[9] H. Durrant-Whyte and T. Bailey, "Simultaneous Localization and Mapping: Part I," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[10] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual Simultaneous Localization and Mapping: A Survey," *Artificial intelligence review*, vol. 43, no. 1, pp. 55–81, 2015.

[11] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[12] F. Fraundorfer and D. Scaramuzza, "Visual Odometry: Part II: Matching, Robustness, Optimization, and Applications," *IEEE Robotics & Automation Magazine*, vol. 19, no. 2, pp. 78–90, 2012.

[13] J. Civera, A. J. Davison, and J. M. Montiel, "Inverse Depth Parametrization for Monocular SLAM," *IEEE transactions on robotics*, vol. 24, no. 5, pp. 932–945, 2008.

[14] N. H. Khan and A. Adnan, "Ego-Motion Estimation Concepts, Algorithms and Challenges: An Overview," *Multimedia Tools and Applications*, vol. 76, no. 15, pp. 16 581–16 603, 2017.

[15] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[16] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *European conference on computer vision*. Springer, 2014, pp. 834–849.

[17] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast Semi-Direct Monocular Visual Odometry," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 15–22.

[18] A. J. Davison, "Real-Time Simultaneous Localisation and Mapping with A Single Camera," in *Computer Vision, IEEE International Conference on*, vol. 3. IEEE Computer Society, 2003, pp. 1403–1403.

[19] A. J. Davison, Y. G. Cid, and N. Kita, "Real-Time 3D SLAM with Wide-Angle Vision," *IFAC Proceedings Volumes*, vol. 37, no. 8, pp. 868–873, 2004.

[20] J. Meltzer, R. Gupta, M.-H. Yang, and S. Soatto, "Simultaneous Localization and Mapping using Multiple View Feature Descriptors," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 2. IEEE, 2004, pp. 1550–1555.

[21] W. Jeong and K. M. Lee, "CV-SLAM: A New Ceiling Vision-Based SLAM Technique," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 3195–3200.

[22] P. Smith, I. Reid, and A. Davison, "Real-Time Monocular SLAM with Straight Lines," in *Proc. British Machine Vision Conference*, Edinburgh, 2006, pp. 17–26.

[23] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[24] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE, 2007, pp. 225–234.

[25] D. Migliore, R. Rigamonti, D. Marzorati, M. Matteucci, and D. G. Sorrenti, "Use A Single Camera for Simultaneous Localization and Mapping with Mobile Object Tracking in Dynamic Environments," in *ICRA Workshop on Safe navigation in open and dynamic environments: Application to autonomous vehicles*, 2009, pp. 12–17.

[26] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D Mapping with An RGB-D Camera," *IEEE transactions on robotics*, vol. 30, no. 1, pp. 177–187, 2013.

[27] M. Li and A. I. Mourikis, "High-Precision, Consistent EKF-Based Visual-Inertial Odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[28] D. Weikersdorfer, D. B. Adrian, D. Cremers, and J. Conradt, "Event-Based 3D SLAM with A Depth-Augmented Dynamic Vision Sensor," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 359–364.

[29] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-Based Visual-Inertial Odometry Using Nonlinear Optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[30] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-Manifold Preintegration for Real-Time Visual-Inertial Odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2016.

[31] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[32] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-Time Globally Consistent 3D Reconstruction Using On-The-Fly Surface Reintegration," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, p. 1, 2017.

[33] R. Mur-Artal and J. D. Tardós, "Visual-Inertial Monocular SLAM with Map Reuse," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, 2017.

[34] D. Schlegel, M. Colosi, and G. Grisetti, "ProSLAM: Graph SLAM from A Programmer's Perspective," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3833–3840.

[35] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 965–972, 2018.

[36] H. Liu, M. Chen, G. Zhang, H. Bao, and Y. Bao, "ICE-BA: Incremental, Consistent and Efficient Bundle Adjustment for Visual-Inertial SLAM," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1974–1982.

[37] T. Qin, P. Li, and S. Shen, "VINS-MONO: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[38] P. Geneva, J. Maley, and G. Huang, "An Efficient Schmidt-EKF for 3D Visual-Inertial SLAM," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 105–12 115.

[39] Y. Li, N. Brasch, Y. Wang, N. Navab, and F. Tombari, "Structure-SLAM: Low-Drift Monocular SLAM in Indoor Environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6583–6590, 2020.

[40] J. Cheng, Z. Wang, H. Zhou, L. Li, and J. Yao, "DM-SLAM: A Feature-Based SLAM System for Rigid Dynamic Scenes," *ISPRS International Journal of Geo-Information*, vol. 9, no. 4, p. 202, 2020.

[41] H. Kim, S. Leutenegger, and A. J. Davison, "Real-Time 3D Reconstruction and 6-DoF Tracking with An Event Camera," in *European Conference on Computer Vision*. Springer, 2016, pp. 349–364.

[42] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An Efficient Alternative to SIFT or SURF," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[43] G. Silveira, E. Malis, and P. Rives, "An Efficient Direct Approach to Visual SLAM," *IEEE transactions on robotics*, vol. 24, no. 5, pp. 969–979, 2008.

[44] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense Tracking and Mapping in Real-Time," in *2011 international conference on computer vision*. IEEE, 2011, pp. 2320–2327.

[45] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-Time Dense Surface Mapping and Tracking," in *2011 10th IEEE international symposium on mixed and augmented reality*. IEEE, 2011, pp. 127–136.

[46] T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. Leonard, "Kintinuous: Spatially Extended KinectFusion," in *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Jul. 2012.

[47] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald, "Robust Real-Time Visual Odometry for Dense RGB-D Mapping," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 5724–5731.

[48] D. Weikersdorfer, R. Hoffmann, and J. Conradt, "Simultaneous Localization and Mapping for Event-Based Vision Systems," in *International Conference on Computer Vision Systems*. Springer, 2013, pp. 133–142.

[49] C. Kerl, J. Sturm, and D. Cremers, "Dense Visual SLAM for RGB-D Cameras," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 2100–2106.

[50] J. Engel, J. Sturm, and D. Cremers, "Semi-Dense Visual Odometry for A Monocular Camera," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1449–1456.

[51] J. Engel, J. Stückler, and D. Cremers, "Large-Scale Direct SLAM with Stereo Cameras," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 1935–1942.

[52] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust Visual Inertial Odometry Using A Direct EKF-Based Approach," in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2015, pp. 298–304.

[53] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "ElasticFusion: Real-Time Dense SLAM and Light Source Estimation," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016.

[54] H. Rebecq, T. Horstschäfer, G. Gallego, and D. Scaramuzza, "EVO: A Geometric Approach to Event-Based 6-DoF Parallel Tracking and Mapping in Real Time," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 593–600, 2016.

[55] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[56] R. Wang, M. Schworer, and D. Cremers, "Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3903–3911.

[57] L. Von Stumberg, V. Usenko, and D. Cremers, "Direct Sparse Visual-Inertial Odometry Using Dynamic Marginalization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2510–2517.

[58] Y. Zhou, G. Gallego, H. Rebecq, L. Kneip, H. Li, and D. Scaramuzza, "Semi-Dense 3D Reconstruction with A Stereo Event Camera," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 235–251.

[59] T. Schops, T. Sattler, and M. Pollefeys, "BAD SLAM: Bundle Adjusted Direct RGB-D SLAM," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 134–144.

[60] Y.-S. Shin, Y. S. Park, and A. Kim, "DVL-SLAM: Sparse Depth Enhanced Direct Visual-Lidar SLAM," *Autonomous Robots*, vol. 44, no. 2, pp. 115–130, 2020.

[61] J. Zubizarreta, I. Aguinaga, and J. M. M. Montiel, "Direct Sparse Mapping," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1363–1370, 2020.

[62] R. A. Newcombe and A. J. Davison, "Live Dense Reconstruction with A Single Moving Camera," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 1498–1505.

[63] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2016.

[64] S. H. Lee and J. Civera, "Loosely-Coupled Semi-Direct Monocular SLAM," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 399–406, 2018.

[65] F. Schenk and F. Fraundorfer, "RESLAM: A Real-Time Robust Edge-Based SLAM System," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 154–160.

[66] F. Tang, H. Li, and Y. Wu, "FMD Stereo SLAM: Fusing MVG and Direct Formulation Towards Accurate and Fast Stereo SLAM," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 133–139.

[67] Q. Liu, Z. Wang, and H. Wang, "SD-VIS: A Fast and Accurate Semi-Direct Monocular Visual-Inertial Simultaneous Localization and Mapping (SLAM)," *Sensors*, vol. 20, no. 5, p. 1511, 2020.

[68] X. Zhao, L. Liu, R. Zheng, W. Ye, and Y. Liu, "A Robust Stereo Feature-Aided Semi-Direct SLAM System," *Robotics and Autonomous Systems*, vol. 132, p. 103597, 2020.

[69] Z. Liang and C. Wang, "A Semi-Direct Monocular Visual SLAM Algorithm in Complex Environments," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 1, pp. 1–19, 2021.

[70] D. Eigen, C. Puhrsch, and R. Fergus, "Depth Map Prediction from A Single Image Using A Multi-Scale Deep Network," *arXiv preprint arXiv:1406.2283*, 2014.

[71] B. Li, C. Shen, Y. Dai, A. Van Den Hengel, and M. 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*, 2015, pp. 1119–1127.

[72] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 2024–2039, 2015.

[73] K. R. Konda and R. Memisevic, "Learning Visual Odometry with A Convolutional Network." in *VISAPP (1)*, 2015, pp. 486–490.

[74] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A Convolutional Network for Real-Time 6-DoF Camera Relocalization," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2938–2946.

[75] G. Costante, M. Mancini, P. Valigi, and T. A. Ciarfuglia, "Exploring Representation Learning with CNNs for Frame-to-Frame Ego-Motion Estimation," *IEEE robotics and automation letters*, vol. 1, no. 1, pp. 18–25, 2015.

[76] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4040–4048.

[77] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-End Learning of Geometry and Context for Deep Stereo Regression," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 66–75.

[78] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, "VINet: Visual-Inertial Odometry as A Sequence-to-Sequence Learning Problem," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[79] S. Wang, R. Clark, H. Wen, and N. Trigoni, "DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2043–2050.

[80] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-Time Dense Monocular SLAM with Learned Depth Prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6243–6252.

[81] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep Ordinal Regression Network for Monocular Depth Estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2002–2011.

[82] F. Xue, Q. Wang, X. Wang, W. Dong, J. Wang, and H. Zha, "Guided Feature Selection for Deep Visual Odometry," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 293–308.

[83] F. Xue, X. Wang, S. Li, Q. Wang, J. Wang, and H. Zha, "Beyond Tracking: Selecting Memory and Refining Poses for Deep Visual Odometry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8575–8583.

[84] C. Chen, S. Rosa, Y. Miao, C. X. Lu, W. Wu, A. Markham, and N. Trigoni, "Selective Sensor Fusion for Neural Visual-Inertial Odometry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 542–10 551.

[85] J. M. Facil, B. Ummenhofer, H. Zhou, L. Montesano, T. Brox, and J. Civera, "CAM-Convs: Camera-Aware Multi-Scale Convolutions for Single-View Depth," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 826–11 835.

[86] R. Garg, V. K. Bg, G. Carneiro, and I. Reid, "Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue," in *European conference on computer vision*. Springer, 2016, pp. 740–756.

[87] Y. Kuznietsov, J. Stuckler, and B. Leibe, "Semi-Supervised Deep Learning for Monocular Depth Map Prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6647–6655.

[88] M. Poggi, F. Tosi, and S. Mattoccia, "Learning Monocular Depth Estimation with Unsupervised Trinocular Assumptions," in *2018 International conference on 3d vision (3DV)*. IEEE, 2018, pp. 324–333.

[89] F. Aleotti, F. Tosi, M. Poggi, and S. Mattoccia, "Generative Adversarial Networks for Unsupervised Monocular Depth Prediction," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[90] A. Pilzer, D. Xu, M. Puscas, E. Ricci, and N. Sebe, "Unsupervised Adversarial Depth Estimation Using Cycled Generative Networks," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 587–595.

[91] C. Wang, J. M. Buenaposada, R. Zhu, and S. Lucey, "Learning Depth from Monocular Videos Using Direct Methods," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2022–2030.

[92] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. Reid, "Unsupervised Learning of Monocular Depth Estimation and Visual Odometry with Deep Feature Reconstruction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 340–349.

[93] R. Li, S. Wang, Z. Long, and D. Gu, "UnDeepVO: Monocular Visual Odometry through Unsupervised Deep Learning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 7286–7291.

[94] Y. Wang, P. Wang, Z. Yang, C. Luo, Y. Yang, and W. Xu, "UNOS: Unified Unsupervised Optical-Flow and Stereo-Depth Estimation by Watching Videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8071–8081.

[95] A. Pilzer, S. Lathuiliere, N. Sebe, and E. Ricci, "Refine and Distill: Exploiting Cycle-Inconsistency and Knowledge Distillation for Unsupervised Monocular Depth Estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9768–9777.

[96] F. Tosi, F. Aleotti, M. Poggi, and S. Mattoccia, "Learning Monocular Depth Estimation Infusing Traditional Stereo Knowledge," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9799–9809.

[97] P.-Y. Chen, A. H. Liu, Y.-C. Liu, and Y.-C. F. Wang, "Towards Scene Understanding: Unsupervised Monocular Depth Estimation with Semantic-Aware Representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2624–2632.

[98] X. Fei, A. Wong, and S. Soatto, "Geo-Supervised Visual Depth Prediction," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1661–1668, 2019.

[99] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki, "SfM-Net: Learning of Structure and Motion from Video," *arXiv preprint arXiv:1704.07804*, 2017.

[100] Z. Yang, P. Wang, W. Xu, L. Zhao, and R. Nevatia, "Unsupervised Learning of Geometry with Edge-Aware Depth-Normal Consistency," *arXiv preprint arXiv:1711.03665*, 2017.

[101] R. Mahjourian, M. Wicke, and A. 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*, 2018, pp. 5667–5675.

[102] Y. Zou, Z. Luo, and J.-B. Huang, "DF-Net: Unsupervised Joint Learning of Depth and Flow Using Cross-Task Consistency," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 36–53.

[103] Z. Yin and J. Shi, "GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1983–1992.

[104] A. Ranjan, V. Jampani, L. Balles, K. Kim, D. Sun, J. Wulff, and M. J. Black, "Competitive Collaboration: Joint Unsupervised Learning of Depth, Camera Motion, Optical Flow and Motion Segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 240–12 249.

[105] G. Wang, H. Wang, Y. Liu, and W. Chen, "Unsupervised Learning of Monocular Depth and Ego-Motion Using Multiple Masks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4724–4730.

[106] S. Li, F. Xue, X. Wang, Z. Yan, and H. Zha, "Sequential Adversarial Learning for Self-Supervised Deep Visual Odometry," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2851–2860.

[107] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of A Convolutional Neural Network," in *2017 International Conference on Engineering and Technology (ICET)*. Ieee, 2017, pp. 1–6.

[108] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent Neural Network Regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[109] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[110] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation," *arXiv preprint arXiv:1406.1078*, 2014.

[111] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," *arXiv preprint arXiv:1506.04214*, 2015.

[112] N. Ballas, L. Yao, C. Pal, and A. Courville, "Delving Deeper into Convolutional Networks for Learning Video Representations," 2016.

[113] D. L. Donoho, "Compressed Sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[114] Y. Wang, H. Yao, and S. Zhao, "Auto-Encoder Based Dimensionality Reduction," *Neurocomputing*, vol. 184, pp. 232–242, 2016.

[115] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," 2014.

[116] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial Transformer Networks," in *Advances in neural information processing systems*, 2015, pp. 2017–2025.

[117] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le, "Attention Augmented Convolutional Networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3286–3295.

[118] A. Geiger, P. Lenz, and R. Urtasun, "Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[119] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[120] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[121] J. Xie, M. Kiefel, M.-T. Sun, and A. Geiger, "Semantic Instance Annotation of Street Scenes by 3D to 2D Label Transfer," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[122] J. Xie, R. Girshick, and A. Farhadi, "Deep3D: Fully Automatic 2D-to-3D Video Conversion with Deep Convolutional Neural Networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 842–857.

[123] Y. Zhong, Y. Dai, and H. Li, "Self-Supervised Learning for Stereo Matching with Self-Improving Ability," *arXiv preprint arXiv:1709.00930*, 2017.

[124] A. Kendall and R. Cipolla, "Modelling Uncertainty in Deep Learning for Camera Relocalization," in *2016 IEEE international conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4762–4769.

[125] R. Li, S. Wang, and D. Gu, "DeepSLAM: A Robust Monocular SLAM System with Unsupervised Deep Learning," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 4, pp. 3577–3587, 2020.

[126] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova, "Unsupervised Monocular Depth and Ego-Motion Learning with Structure and Semantics," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[127] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova, "Depth from Videos in the Wild: Unsupervised Monocular Depth Learning from Unknown Cameras," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8977–8986.

[128] H. Li, A. Gordon, H. Zhao, V. Casser, and A. Angelova, "Unsupervised Monocular Depth Learning in Dynamic Scenes," *arXiv preprint arXiv:2010.16404*, 2020.

[129] Y. Almalioglu, M. R. U. Saputra, P. P. de Gusmao, A. Markham, and N. Trigoni, "GANVO: Unsupervised Deep Monocular Visual Odometry and Depth Estimation with Generative Adversarial Networks," in *2019 International conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 5474–5480.

[130] T. Feng and D. Gu, "SGANVO: Unsupervised Deep Visual Odometry and Depth Estimation with Stacked Generative Adversarial Networks," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4431–4437, 2019.

[131] G. Zhou, B. Bescos, M. Dymczyk, M. Pfeiffer, J. Neira, and R. Siegwart, "Dynamic Objects Segmentation for Visual Localization in Urban Environments," *arXiv preprint arXiv:1807.02996*, 2018.

[132] Y. Yang, A. Loquercio, D. Scaramuzza, and S. Soatto, "Unsupervised Moving Object Detection via Contextual Information Separation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 879–888.

[133] H. Zhang, H. Chang, B. Ma, N. Wang, and X. Chen, "Dynamic R-CNN: Towards High Quality Object Detection via Dynamic Training," in *European Conference on Computer Vision*. Springer, 2020, pp. 260–275.

[134] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, p. 674–679.

[135] J. Wulff and M. J. Black, "Efficient Sparse-to-Dense Optical Flow Estimation Using A Learned Basis and Layers," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 120–130.

[136] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1164–1172.

[137] R. Schuster, C. Bailer, O. Wasenmüller, and D. Stricker, "FlowFields++: Accurate Optical Flow Correspondences Meet Robust Interpolation," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 1463–1467.

[138] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning Optical Flow with Convolutional Networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.

[139] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of Optical Flow Estimation with Deep Networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2462–2470.

[140] A. Ranjan and M. J. Black, "Optical Flow Estimation Using A Spatial Pyramid Network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4161–4170.

[141] Z. Ren, O. Gallo, D. Sun, M.-H. Yang, E. B. Sudderth, and J. Kautz, "A Fusion Approach for Multi-Frame Optical Flow Estimation," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 2077–2086.

[142] T.-W. Hui, X. Tang, and C. C. Loy, "LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8981–8989.

[143] T.-W. Hui and C. C. Loy, "LiteFlowNet3: Resolving Correspondence Ambiguity for More Accurate Optical Flow Estimation," in *European Conference on Computer Vision*. Springer, 2020, pp. 169–184.

[144] J. Hur and S. Roth, "Iterative Residual Refinement for Joint Optical Flow and Occlusion Estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5754–5763.

[145] S. Meister, J. Hur, and S. Roth, "UnFlow: Unsupervised Learning of Optical Flow with A Bidirectional Census Loss," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[146] J. Janai, F. Guney, A. Ranjan, M. Black, and A. Geiger, "Unsupervised Learning of Multi-Frame Optical Flow with Occlusions," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 690–706.

[147] P. Liu, I. King, M. R. Lyu, and J. Xu, "DDFlow: Learning Optical Flow with Unlabeled Data Distillation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 8770–8777.

[148] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image Segmentation Using Deep Learning: A Survey," *arXiv preprint arXiv:2001.05566*, 2020.

[149] D. Ciresan, A. Giusti, L. Gambardella, and J. Schmidhuber, "Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images," *Advances in neural information processing systems*, vol. 25, pp. 2843–2851, 2012.

[150] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-Based Convolutional Networks for Accurate Object Detection and Segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 142–158, 2015.

[151] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[152] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *arXiv preprint arXiv:1506.01497*, 2015.

[153] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[154] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs," *arXiv preprint arXiv:1412.7062*, 2014.

[155] Chen, Liang-Chieh and Papandreou, George and Kokkinos, Iasonas and Murphy, Kevin and Yuille, Alan L, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[156] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking Atrous Convolution for Semantic Image Segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[157] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[158] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid Scene Parsing Network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[159] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual Attention Network for Scene Segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3146–3154.

[160] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "CCNet: Criss-Cross Attention for Semantic Segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 603–612.

[161] C. Yu, J. Wang, C. Gao, G. Yu, C. Shen, and N. Sang, "Context Prior for Scene Segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 416–12 425.

[162] Q. Hou, L. Zhang, M.-M. Cheng, and J. Feng, "Strip Pooling: Rethinking Spatial Pooling for Scene Parsing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4003–4012.

[163] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[164] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8759–8768.

[165] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang, "Mask Scoring R-CNN," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6409–6418.

[166] B. De Brabandere, D. Neven, and L. Van Gool, "Semantic Instance Segmentation with A Discriminative Loss Function," *arXiv preprint arXiv:1708.02551*, 2017.

[167] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[168] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[169] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-Time Instance Segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9157–9166.

[170] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li, "SOLO: Segmenting Objects by Locations," in *European Conference on Computer Vision*. Springer, 2020, pp. 649–665.

[171] A. Bak, S. Bouchafa, and D. Aubert, "Dynamic Objects Detection through Visual Odometry and Stereo-Vision: A Study of Inaccuracy and Improvement Sources," *Machine vision and applications*, vol. 25, no. 3, pp. 681–697, 2014.

[172] L. Xiao, J. Wang, X. Qiu, Z. Rong, and X. Zou, "Dynamic-SLAM: Semantic Monocular Visual Localization and Mapping Based on Deep Learning in Dynamic Environment," *Robotics and Autonomous Systems*, vol. 117, pp. 1–16, 2019.

[173] S. Pillai, R. Ambrus, and A. Gaidon, "SuperDepth: Self-Supervised, Super-Resolved Monocular Depth Estimation," *arXiv preprint arXiv:1810.01849*, 2018.

[174] A. C. Kumar, S. M. Bhandarkar, and M. Prasad, "Monocular Depth Prediction Using Generative Adversarial Networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2018, pp. 413–4138.

[175] K. Gwn Lore, K. Reddy, M. Giering, and E. A. Bernal, "Generative Adversarial Networks for Depth Map Estimation From RGB Video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1177–1185.

[176] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4580–4584.

[177] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1874–1883.

[178] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved Training of Wasserstein GANs," *arXiv preprint arXiv:1704.00028*, 2017.

[179] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-Normalizing Neural Networks," in *Advances in neural information processing systems*, 2017, pp. 971–980.

[180] T. Chavdarova and F. Fleuret, "SGAN: An Alternative Training of Generative Adversarial Networks," in *Proceedings of the IEEE international conference on Computer Vision and Pattern Recognition*, 2018.

[181] S. Wang, R. Clark, H. Wen, and N. Trigoni, "End-to-End, Sequence-to-Sequence Probabilistic Visual Odometry through Deep Neural Networks," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 513–542, 2018.

[182] Y. Luo, J. Ren, M. Lin, J. Pang, W. Sun, H. Li, and L. Lin, "Single View Stereo Matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 155–163.

[183] C. Wang, X. Bai, X. Wang, X. Liu, J. Zhou, X. Wu, H. Li, and D. Tao, "Self-Supervised Multiscale Adversarial Regression Network for Stereo Disparity Estimation," *IEEE Transactions on Cybernetics*, 2020.

[184] V. Prasad and B. Bhowmick, "SfMLearner++: Learning Monocular Depth & Ego-Motion Using Meaningful Geometric Constraints," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 2087–2096.

[185] L. Zhou, J. Ye, M. Abello, S. Wang, and M. Kaess, "Unsupervised Learning of Monocular Depth Estimation with Bundle Adjustment, Super-Resolution and Clip Loss," *arXiv preprint arXiv:1812.03368*, 2018.

[186] M. Klingner, J.-A. Termöhlen, J. Mikolajczyk, and T. Fingscheidt, "Self-Supervised Monocular Depth Estimation: Solving The Dynamic Object Problem by Semantic Guidance," in *European Conference on Computer Vision*. Springer, 2020, pp. 582–600.

[187] B. Wagstaff, V. Peretroukhin, and J. Kelly, "Self-Supervised Deep Pose Corrections for Robust Visual Odometry," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2331–2337.

[188] R. Ambrus, V. Guizilini, J. Li, and S. P. A. Gaidon, "Two Stream Networks for Self-Supervised Ego-Motion Estimation," in *Conference on Robot Learning*. PMLR, 2020, pp. 1052–1061.

[189] H. Jiang, L. Ding, Z. Sun, and R. Huang, "DIPE: Deeper Into Photometric Errors for Unsupervised Learning of Depth and Ego-Motion from Monocular Videos," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 061–10 067.

[190] W. Zhao, S. Liu, Y. Shu, and Y.-J. Liu, "Towards Better Generalization: Joint Depth-Pose Learning without PoseNet," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9151–9161.

[191] Y. Chang, C. Jung, and J. Sun, "Joint Reflection Removal and Depth Estimation From A Single Image," *IEEE Transactions on Cybernetics*, pp. 1–14, 2020.

[192] C. Li, R. Cong, S. Kwong, J. Hou, H. Fu, G. Zhu, D. Zhang, and Q. Huang, "ASIF-Net: Attention Steered Interweave Fusion Network for RGB-D Salient Object Detection," *IEEE transactions on cybernetics*, vol. 51, no. 1, pp. 88–100, 2020.

[193] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[194] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[195] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility To Structural Similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[196] S. Lee, S. Im, S. Lin, and I. S. Kweon, "Learning Residual Flow as Dynamic Motion from Stereo Videos," *arXiv preprint arXiv:1909.06999*, 2019.

[197] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox, "Flownet: Learning Optical Flow with Convolutional Networks," in *IEEE International Conference on Computer Vision (ICCV)*, 2015. [Online]. Available: http://lmb.informatik.uni-freiburg.de/Publications/2015/DFIB15

[198] H. Zhan, C. S. Weerasekera, J. W. Bian, and I. Reid, "Visual Odometry Revisited: What Should Be Learnt?" in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4203–4210.

[199] Z. Teed and J. Deng, "DeepV2D: Video to Depth with Differentiable Structure from Motion," *arXiv preprint arXiv:1812.04605*, 2018.

[200] H. Alhaija, S. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes," *International Journal of Computer Vision (IJCV)*, 2018.

[201] H. Zhan, C. S. Weerasekera, J.-W. Bian, R. Garg, and I. Reid, "DF-VO: What Should Be Learnt for Visual Odometry?" *ArXiv Preprint ArXiv:2103.00933*, 2021.

[202] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *ArXiv Preprint ArXiv:1704.04861*, 2017.

[203] D. de Geus, P. Meletis, and G. Dubbelman, "Single Network Panoptic Segmentation for Street Scene Understanding," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 709–715.