Dynamic Scene Understanding using Deep Neural Networks

Ye Lyu

DYNAMIC SCENE UNDERSTANDING USING DEEP NEURAL NETWORKS

D I S S E R T A T I O N

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof. dr. ir. A. Veldkamp,
on account of the decision of the Doctorate Board,
to be publicly defended
on Wednesday 8th September 2021 at 14.45

by

Ye Lyu
born on the 21st of July, 1991
in Wuhan, China

This dissertation is approved by:

Prof. dr. ir. M.G. Vosselman , promoter
Dr.-Ing. M.Y. Yang , co-promoter

ITC dissertation number 400
ITC, P.O. Box 217, 7500 AE Enschede, The Netherlands

ISBN 978-90-365-5223-3
DOI 10.3990/1.9789036552233

Cover designed by Ye Lyu and Job Duim
Printed by CTRL-P, Hengelo

UNIVERSITY OF TWENTE.

FACULTY OF GEO-INFORMATION SCIENCE AND EARTH OBSERVATION

Graduation committee

Chair / secretary

Prof. dr. F.D. van der Meer	University of Twente (ITC)

Promoter

Prof. dr. ir. M.G. Vosselman	University of Twente (ITC)

Co-promoter

Dr.-Ing. M.Y. Yang	University of Twente (ITC)

Members

Prof. dr. ir. A. Stein	University of Twente (ITC)
Prof. dr. C. Brune	University of Twente (EEMCS)
Prof. Dr.-Ing. B. Rosenhahn	Leibniz Universität Hannover
Prof. dr. A. Yilmaz	The Ohio State University

Acknowledgment

I still remember the first day when I came to the Netherlands in August 27th, 2017. I was in my T-shirt, and it was a little cold when the plane landed in the Netherlands before dawn. It was my first time being abroad, and I was excited and a bit nervous. I had never seen so many foreigners before, and I had to speak foreign language all the way. It was awkward as I almost went to another city due to my inaccurate pronunciation when I purchased the train ticket. As dawn was breaking, my Ph.D. journey begins after boarding the right train to Enschede. Thanks to my great supervisors, colleagues, friends, and all other staffs, it only took few days before my discomforts cleared up.

Four years have been a long period, and I have turned into a more experienced researcher. I would like to thank my awesome supervisors, George and Michael. Without you, I could hardly achieve what I have today. My promoter George Vosselman is always willing to help, and he always takes a serious and responsible attitude towards his students. He had helped my paper writing in such details, which is beyond my expectation. He has a sharp and critical mind towards research, and is always able to identify key issues in papers and ideas during reading classes and Ph.D. seminars. I appreciate every advice he has ever given me, and there is one advice from him I have been keeping in mind, which dates back to my proposal writing: think carefully before doing it.

I would thank my co-promoter Michael Yang greatly, with whom I communicate the most for all aspects of my research. He has very good vision towards possible research directions, and he is open to new topics and ideas. Michael always encourages and supports us to try new methods, which inspires me a lot. He also taught us an important thing: persevere with our work. Try rebuttal even though the paper is likely to be rejected as that is the full cycle of submission. I would also like to thank Michael for staying up late revising paper with us for those memorable nights.

I am also very grateful to Prof. Guisong Xia from Wuhan University. It is him who introduced me to Michael. It is an honor to work with him and get acquainted with his amazing Captain team. Guisong is so kind to help me connect with other outstanding students. I had a short visit to his group, and got impressed and inspired by his talented and high aiming team. I would like to thank Kunping Yang, Jian Ding, Fudong Wang, Tianzhu Xiang, Yang Long, and Nan Xue for their help.

For these four years, I am proud of being a member of EOS friends. My first great impression of my team is that people care about each other. My promoter George and our previous EOS secretary Teresa, they are so friendly and easy of approach. They had kindly shown me around our offices and introduced me to other

colleagues on my first day at ITC building to help me blend into our team. Michael often brought us chocolates and candies for different festivals to cheer us up. My awesome office mates, Caroline and Andrea were so nice to help me get used to my new life. Encouraged by Caroline, I had made my first yogurt cake for the bake-off competition in ITC, and I learned that Dutch people are so good at making cakes and their carrot cake is really tasty. Thanks to Ville for inviting us to his birthday party, it was nice to have fun with you. Joep was perhaps the one I had chatted the most during EOS coffee breaks, and I am grateful for his help regarding my coding issues. I enjoyed the moments chatting with Samer, Diogo, Phillipp, Shayan, and Zill during lunch in the early days. Many thanks to Sofia as well to help me understand Dutch language. Our new secretary Jenny has also kindly helped me with my writing for several times, towards which I am so grateful. Mengmeng Li, Jia Peng, and Fashuai Li, had kindly organized gatherings and dinners during Chinese festivals, which made us feel at home during our life abroad.

I am grateful for my friends in the scene understanding group (SUG) led by Michael. Zhenchao Zhang was the first person from ITC that I met, who had always been ready to help. I still remembered my first lunch here cooked and treated by him. Yaping Lin is also a very nice teammate, who would always like to try making special hometown food and share with us, she is really good at cooking. I would like to thank our senior teammate Wentong Liao for sharing job information with me, and he has set a good exemplary as a researcher for us. I appreciate the moments discussing papers and ideas with all my teammates, which are very enlightening.

There are many other awesome friends from ITC. Yanwen Wang, Wufan Zhao, Cai Wu, Bin Zhang, Ning Zhang, Shaoqin Dai, Siyang Chen, Qilei Huang, Wen Zhou, Ting Zhou, Yao Li, Ruosha Zeng, Ruodan Zhuang, Zhishuang Yang, Ze Li, Qianqian Han, Siqi Shi, Xu Zhang. We had lots of fun moments together. We rode roller coasters in Walibi park, embraced nature in De Hoge Veluwe national park, enjoyed brilliant artworks in various museums, and cheered for the Twente football team in the FC Twente stadium.

I am also grateful to have met great and interesting friends from other faculties in UT. It all started with five initiators, Kun Dai, Jie Wang, Tao Tian, Panfei Sun and me. I still remember the days when we crossed the border into Germany at night, only to buy some wines and chocolates. Our friend group grew as we would invite new comers to meet each other. One of the most memorable European travel was in Italy together with Pei Zhang, Weiqiu Chen, Minsi Li, Zhiguo Zhang, Zhen Jiao, Tao Tian, Kun Dai, and Keyan Chen, which dates back to Christmas in 2017. Those memories are so fresh as if they happened yesterday. In Netherlands, we had more funny, crazy, and even stupid moments together with more friends, Min Lin, Mengna Li, Juan Wang, Hao Wu, Jinmeng Hao, and Dan Hu.

Thanks to the Chinese Scholarship Council (CSC), which has supported my four year ph.D. life financially. It is an honor as a recipient of the scholarship, and it also means the responsibility we carry. I would like to share my knowledge and experience learned here with people in my country, which is far more than just scientific research. I would like to disseminate my impression of Dutch people, and show others how friendly and helpful Dutch people are, how they are enthusiastic towards life, how responsible they are towards their research and work, and how open they are towards different cultures. I treasure my four year experience here, and I would continuously learn and benefit from it in the following years.

Lastly, special thanks to my parents, relatives and Chenming, who have always been my strongest support all the time. Thousands of miles' distance never weaken our feelings for each other, and whenever we need each other, we will always be there.

Contents

List of Figures

List of Tables

Introduction 1

1.1 Background

Computer vision is an interdisciplinary field dating back to the 1960s, whereby computers acquire high-level understanding from digital images and videos. The goal is to make computers perceive and understand the world as humans do through the visual system, including describing the world and reconstructing its properties [174]. Scene understanding is a part of computer vision as it makes computers understand scenes in images and videos. A scene is a view of a real world environment that contains multiple objects, which are organized in a meaningful way. Scene understanding finds out which objects are in the scene, where they are, and their relationship. It also includes logic reasoning, to establish what is happening, why it is happening, and what will happen. Based on this information, scene understanding tries to assess how to react and what to do next [82]. The main objective of scene understanding is to make computers understand the world autonomously. Scene understanding incorporates numerous tasks including identifying objects in a scene, finding where the objects are localized in the real world[157], and determining attributes of objects of interest[127]. The spatial and semantic relationships between objects can also be characterized and described[211].

Scene understanding is also an important task in the field of earth observation, which serves as the foundation of complicated higher level tasks, such as hazard detection[147], traffic surveillance[46] and environment management[30]. Both computer vision and remote sensing researchers have been propelling researches in the field of scene understanding forward.

Compared to the static image data, video data contains time sequence information, which supports even more applications. Many real time applications, such as surveillance and monitoring[10], often require video data for anomaly detection. Video data also provides additional data consistency. As video frames are collected in sequence, subsequent frames have large overlaps with each other. The information captured in one frame can be propagated into subsequent frames to enhance the information, e.g., semi-supervised tracking relies on such facts. For other scene understanding tasks, such as detection and semantic segmentation, time sequence information could also help to boost the robustness of the algorithm.

1.2 The Advance of Deep Learning

Deep learning is currently the most effective tool for scene understanding. "Deep" refers to deep hierarchical concepts learned by computers [70], which are achieved most effectively by deep neural networks.

The deep neural network is a powerful non-convex regressor or classifier, which originated some time ago, but it had not received much attention as a result of limited computational power, lack of effective training methods and training data.

On the contrary, many other models are favoured, such as support vector machine, boosting algorithm and random forest algorithm. These models are carefully designed and deals more effectively with problems of small scale data, which was the typical setting in earlier research.

However, with the development of the algorithm, the improvement of computation power, and collection of more and more data, it was inevitable that deep learning started to thrive. For the first time deep learning achieved record-breaking results in both classification and localization tasks in the world's greatest computer vision competition, namely, the Large Scale Visual Recognition Challenge in 2012. This proved the capability of deep learning to extract semantic information from a large quantity of data. From then on, deep learning started to sweep all the first places in various scene understanding competitions, including object detection, segmentation, tracking. The success of deep learning can be attributed to its ability to learn rich feature representations automatically as opposed to hand-designed features used in traditional methods.

More and more research has arrived at the same conclusion: provided with enough training data, with enough computation resources, under the guidance of an effective training method, deep learning has the power to interpret complex semantic information, which is of great use for scene understanding.

At present, apart from being able to understand a single image, deep learning has also been wired to gain memory ability to infer object properties based on early information that has been acquired. Deep learning has the power to use memory to infer the context of a sentence and to infer information in a video frame based on previous frames to retrieve temporarily occluded objects. Time domain information of video is also exploited by deep learning models.

1.3 Fundamental Research Tasks

Scene understanding has a broad scope, and this dissertation only focuses on part of the fundamental tasks, which are object detection, semantic segmentation, and tracking in images and videos.

Detection is the union of object recognition and localization in the scene. Recognition differentiates the categories and properties of different objects, which are represented as labels for objects, while localization acquires the position and size of different objects, which are normally represented as bounding boxes. Examples of object detection are shown in Figure 1.1. The object detection could potentially support other tasks that are instance specific. As shown in Figure 1.2, object masks are determined in the instance segmentation task, and human keypoints can be inferred according to the humans detected.

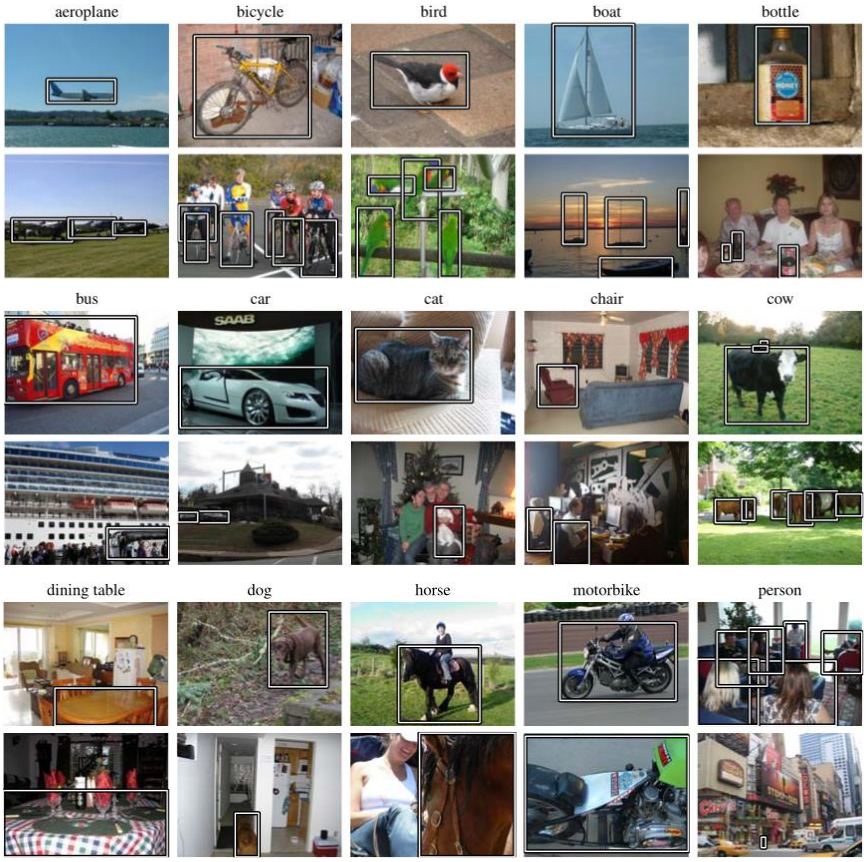

Figure 1.1 Examples of object detection from Pascal VOC benchmark [61]. Object detection simultaneously localizes and classifies the objects in images or videos.

Semantic segmentation, as another part of scene understanding, tries to classify the categories of each pixel in the image. Semantic segmentation complements the object detection and instance segmentation as not all things in a scene are objects that are countable, e.g., sky and ground. Semantic segmentation is normally a more spatially accurate recognition, but the limitation of semantic segmentation is that it cannot differentiate different objects of the same category if they are adjacent. An example of this limitation is shown in Figure 1.5.

Panoptic segmentation is a newly proposed task compared with semantic segmentation and instance segmentation. The limitations for semantic segmentation and instance segmentation are obvious. Semantic segmentation cannot differentiate overlapped objects of the same classes, while instance segmentation cannot handle amorphous background regions, such as sky and grass. Panoptic segmentation is brought out to tackle these problems. The goal of panoptic segmentation is to unify the instance segmentation and semantic segmentation as the two tasks are mutually complimentary. Examples of panoptic segmentation are shown in Figure1.4. The objects such as cars and pedestrians are marked in different colors as different

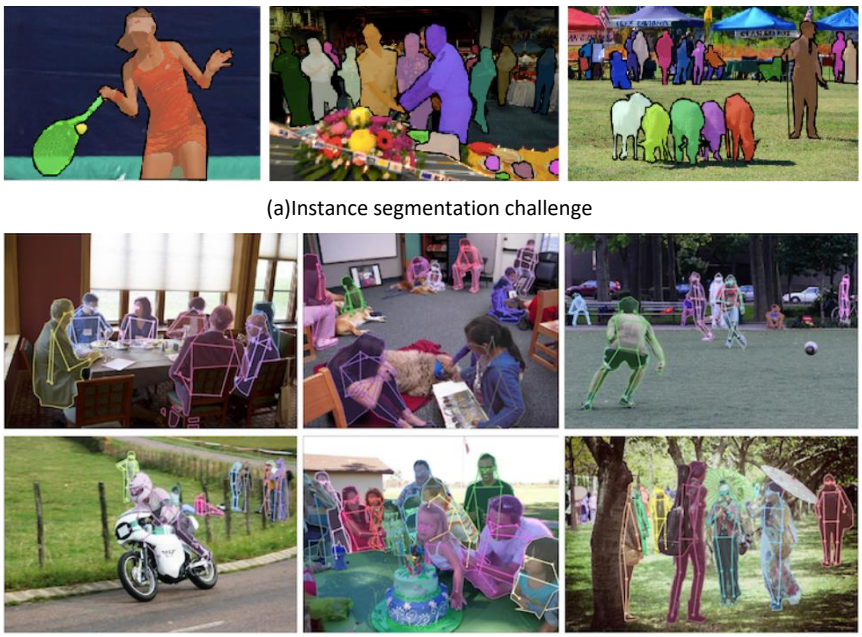

Figure 1.2 Examples of the instance segmentation challenge and the keypoint challenge from COCO benchmark [122]. The instance segmentation challenge is to retrieve the detailed mask of its individual object in the scene. The keypoint challenge is to infer the key points of the object, which describe the poses of the objects.

objects need to be differentiated.

Tracking extends the static scene understanding into dynamic scene understanding, which finds the correspondences at object level or pixel level between frames in a video. Single object tracking and multiple object tracking have been actively researched. They mainly focus on bounding box level tracking. In order to further boost the performance of tracking with more accurate object localization, it has been extended to pixel level tracking, upgrading the single object tracking (SOT) and multiple object tracking (MOT) to the video object segmentation (VOS) task and multi-object tracking and segmentation (MOTS) task. Tracking is more than an independent research task, as it could also help in other scene understanding tasks. For example, tracking could potentially improve the object localization and recognition for object detection in videos, or it could assist with the performance of the video instance segmentation task. Examples of tracking related tasks are shown in Figure 1.6.

My research is about dynamic scene understanding, which covers semantic segmentation, object detection, and tracking at both pixel level and object level. It covers different dynamic scene types, ranging from urban scenes captured by an unmanned aerial vehicle (UAV), common scenes where common objects are observed in daily life, and traffic scenes where cars and pedestrians are the main focus. The goal of the research is to explore the possibility of making the computer possess the

(a) COCO stuff segmentation challenge

(b) ADE20K semantic scene parsing challenge

(c) Cityscapes semantic labeling challenge

Figure 1.3 Examples of semantic segmentation. Semantic segmentation is to label each pixel with the object category it belongs to in an image. (a)(b)(c) are examples from COCO [122], ADE20K [226], and Cityscapes [47] datasets, respectively.

high level interpretation ability to understand the world, and to provide humans with insightful information, knowledge and guidance to make better decisions.

1.4 Scene Understanding with Consistency

In the last section, some fundamental scene understanding tasks have been introduced. Compared to a single image, video captures more information. Consecutive frames from a video have very strong correlations, most area show the same objects, and there is valuable time consistent information that should be well employed.

Frame-by-frame recognition, ignoring temporal information, often yields jittering across frames, especially at object boundaries. As a result, it is preferable to utilize a batch of frames to achieve consistent semantic segmentation in time sequence. Consistency will boost the stability and accuracy of the algorithm.

To leverage on temporal information, conditional random field(CRF) and deep neural network(DNN) based methods are often used to gain temporal consistency.

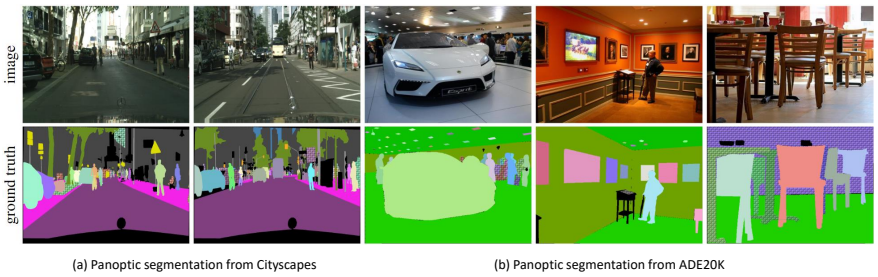

Figure 1.4 Examples of panoptic segmentation. Panoptic segmentation is to unify the semantic segmentation task and the instance segmentation task. The two images on the left are from the Cityscapes benchmark [47]. The three images on the right are from the ADE20k benchmark [226].

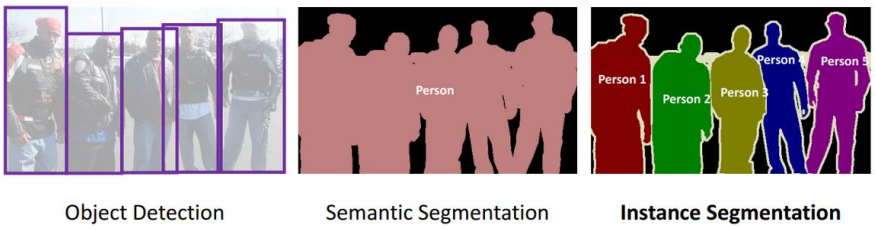

Figure 1.5 Example of the limitation of the semantic segmentation. Semantic segmentation cannot differentiate different objects when they are adjacent.

There are three common ways to ensure consistency across frames.

1) Matching. Consistency may come from correspondences of low level features. A straight forward way is to learn optical flow [42], which provides pixel to pixel matching. Feature propagation [93] or probabilistic graphical model [94], such as markov random field, could also refine the segmentation prediction by smoothing pixel predictions that are close in low level feature space, such as color, texture, and location. Matching of objects through object tracking could also be used to improve the object recognition.

2) Evolution. Another way to preserve time consistency is through online learning, which could be understood as evolution of the deep neural networks. The model is fine-tuned based on the confident prediction in order to adapt to the upcoming frames. Some good results have been reported on the DAVIS video object segmentation benchmark [148, 24, 136, 183].

3) Memory. Memory modules have been successfully used to model sequential data [81, 44]. It should be beneficial for video related tasks since their sequential property. In order to handle 2D image data, convolutional gated recurrent unit (Conv-GRU) [8] and convolutional long short term memory (Conv-LSTM) [203] have been proposed, it would be promising to apply these memory modules in order to achieve consistent prediction. [177] is an example, which uses Conv-GRU to handle the video object segmentation task.

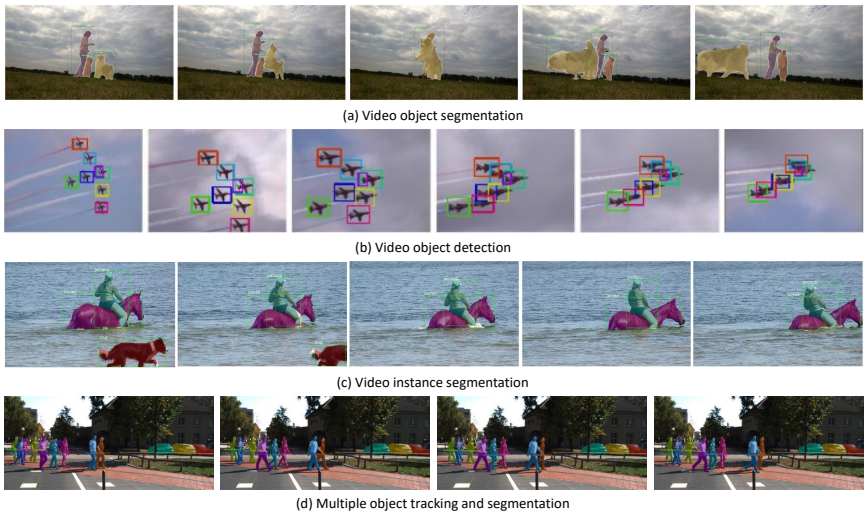

Figure 1.6 Examples of tracking related tasks. Video object segmentation task tackles object tracking at pixel level, example images are from DAVIS dataset [151]. Video object detection, video instance segmentation, and multi-object tracking and segmentation need to detect objects first, so tracking could be used to link objects across frames and improve the performance of object recognition. Example images for the video object detection, video instance segmentation, and multi-object tracking and segmentation are from ImageNet VID dataset [165], Youtube VIS dataset [208], and KITTI MOTS dataset [67], respectively.

1.5 Research Gap

For dynamic urban scene understanding, there is still no light-weighted aerial drone based solution, which uses images with very high spatial resolutions in oblique view. A new semantic segmentation dataset for the UAV imagery is needed serving as one fundamental benchmark. As oblique viewing brings larger object scale variation, better models that are adaptive to different object scales need to be designed. In order to bridge the gap between static scene understanding and dynamic scene understanding, research investigating the relations between object recognition and object tracking could be explored. Specifically, we will explore feature sharing and multi-task learning that combine both object recognition and object tracking. The aim is to simplify the pipeline for enhanced speed while maintaining performance. These directions still lack research as most works explore object recognition and object tracking independently.

1.6 Research Objectives

This Ph.D. thesis focuses on research for dynamic scene understanding based on deep learning methods. Both segmentation and detection in videos are explored. The relevant research tasks include semantic segmentation, video object segmentation,

video object detection, and multi-object tracking and segmentation in videos. For each research task, we have addressed different sub-problems, which all focus on dynamic scene understanding.

Details of the research objectives are as follows,

(i) Semantic segmentation of UAV images for the dynamic urban scene. In this objective, we will explore the possibility of dynamic scene understanding from static images. The static context information could also provide dynamic information to some degree, e.g., cars in the parking lot are more likely to be static, while cars in the middle of roads are more likely to be moving. Several sub-objectives are included. Firstly, establishing the UAV image dataset for the semantic segmentation task. Secondly, providing baseline methods for the semantic segmentation benchmark. Finally, exploring novel methods to extract multi-scale context information from the UAV images. We also explore how to achieve consistent predictions for semantic segmentation in videos based on the high correlations of the frames in a video.

(ii) Video object segmentation for common objects. The major objective is learning to preserve segmentation consistency across multiple frames for multiple objects simultaneously. We explore how the features learned for instance segmentation could be adapted for pixel-level video object tracking.

(iii) In the video object detection task, the goal is to maintain consistent detection results for multiple objects across multiple frames. We explore backbone feature sharing for object detection and object tracking to speed up the pipeline. As labeling video data is quite expensive, we also investigate how to tackle the problem of the lack of training videos by leveraging on a class-agnostic tracker.

(iv) For the task of multi-object tracking and segmentation in videos, the major objective is learning to detect, segment, and track multiple objects simultaneously. We explore multi-task learning, which unifies detection, segmentation, and tracking.

Different objectives focus on different tasks. In order to show the relations between different objectives, relations between different tasks are presented in Figure 1.7. The connections shown in the figure illustrate the relations between the tasks, which are only valid in our method design, which may not hold for other methods.

1.7 Outline

Objectives (i), (ii), (iii), (iv) are addressed in chapters 2, 3, 4, 5, 6. The structure of the dissertation is as follows,

Chapter1 introduces the research background, the research gap, the research objectives, and the outline of this dissertation.

Chapter2 explores how to establish the semantic segmentation benchmark for the UAV images, which includes data collection, data labeling, dataset construction, and performance evaluation with baseline deep neural networks, such as fcn8s [128], U-Net [161] and dilation net [216]. A novel multi-scale dilation net is introduced serving as an improved baseline method. Conditional random field (CRF) with

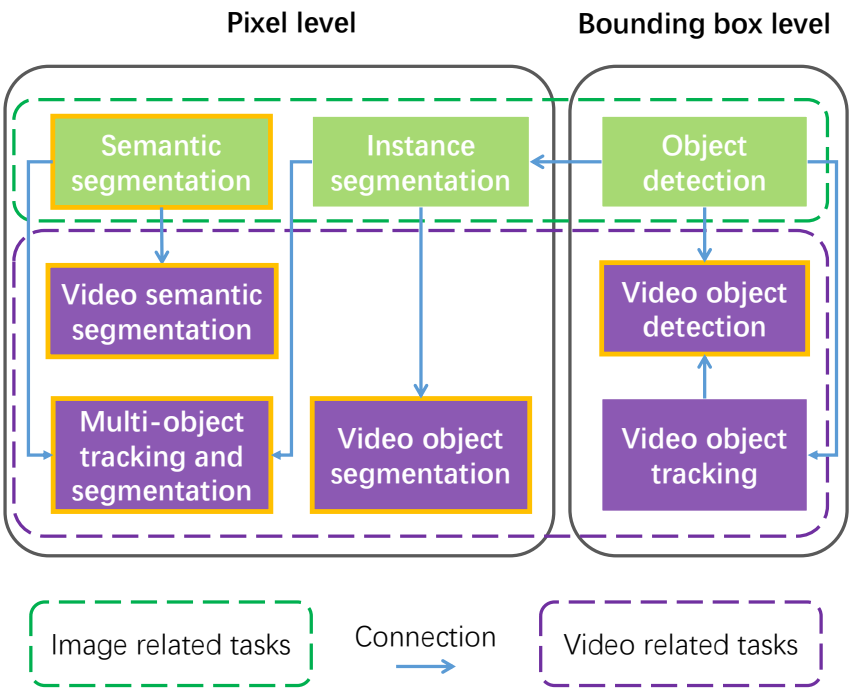

Figure 1.7 Relations between different tasks. The tasks for different objectives are high-lighted in orange boxes. The connections between different tasks are shown with blue arrows, which is only valid in our method design.

feature space optimization (FSO) [107] is used to achieve consistent semantic segmentation prediction in videos.
Chapter3 explores how to better extract the scene context information for improved object recognition performance. By proposing the novel bidirectional multi-scale attention networks, objects could be recognized in proper scales for better performance.
Chapter4 explores how to simultaneously segment multiple objects across multiple frames by combining memory modules with instance segmentation networks.
Chapter5 explores how to improve the performance of well-trained object detectors with a light weighted and efficient plug&play tracker for video object detection. This chapter also explores how the proposed model performs when lacking video training data.
Chapter6 explores how to improve the performance of detection, segmentation, and tracking by jointly considering top-down and bottom-up inference.
Chapter7 synthesizes the work, and draws conclusions for each research objective with reflections on current research trend and recommendations for future work.

UAVid: A Semantic Segmentation Benchmark for UAV Imagery

2

Abstract

Semantic segmentation has been one of the leading research interests in computer vision recently. It serves as a perception foundation for many fields, such as robotics and autonomous driving. The fast development of semantic segmentation attributes enormously to the large scale datasets, especially for the deep learning related methods. There already exist several semantic segmentation datasets for comparison among semantic segmentation methods in complex urban scenes, such as the Cityscapes and CamVid datasets, where the side views of the objects are captured with a camera mounted on the driving car. There also exist semantic labeling datasets for the airborne images and the satellite images, where the top views of the objects are captured. However, only a few datasets capture urban scenes from an oblique Unmanned Aerial Vehicle (UAV) perspective, where both of the top view and the side view of the objects can be observed, providing more information for object recognition. In this chapter, we introduce our UAVid dataset, a new high-resolution UAV semantic segmentation dataset as a complement, which brings new challenges, including large scale variation, moving object recognition and temporal consistency preservation. Our UAV dataset consists of 30 video sequences capturing 4K high-resolution images in oblique views. In total, 300 images have been densely labeled with 8 classes for the semantic labeling task. We have provided several deep learning baseline methods with pre-training, among which the proposed Multi-Scale-Dilation net performs the best via multi-scale feature extraction, reaching a mean intersection-over-union (IoU) score around 50% and outperforming the others by more than 1.6%. We have also explored the influence of spatial-temporal regularization for sequence data by leveraging on feature space optimization (FSO) and 3D conditional random field (CRF), which improves the mean IoU scores by around another 0.5%. Our UAVid website and the labeling tool have been published [1].

[1] https://uavid.nl/

[*] This chapter is based on:

Y. Lyu, G. Vosselman, G.-S. Xia, A. Yilmaz, and M. Y. Yang. Uavid: A semantic segmentation dataset for uav imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 165:108 – 119, 2020

2.1 Introduction

Visual scene understanding has been advancing in recent years, which serves as a perception foundation for many fields such as robotics and autonomous driving. The most effective and successful methods for scene understanding tasks adopt deep learning as their cornerstone, as it can distill high-level semantic knowledge from the training data. However, the drawback is that deep learning requires a tremendous number of samples for training to make it learn useful knowledge instead of noise, especially for real-world applications. Semantic segmentation, as part of scene understanding, is to assign labels for each pixel in the image. In order to make the best of deep learning methods, a large number of densely labeled images are required.

At present, there are several public semantic segmentation datasets available, which focus only on common objects in natural images. They all capture images from the ground. The Ms-COCO [122] and the Pascal VOC [60] datasets provide semantic segmentation tasks for common object recognition in common scenes. They focus on classes like person, car, bus, cow, dog, and other objects. In order to help semantic segmentation models generalize better across different scenes, the ADE20K dataset [226] spans more diverse scenes. Objects from much more different categories are labeled, bringing more variability and complexity for object recognition. The above datasets are often used for common object recognition.

There are more semantic segmentation datasets designed using street scenes for autonomous driving scenarios [20, 100, 47, 166, 218, 67]. Images are captured with cameras mounted on vehicles. The objects of interest include pedestrians, cars, roads, lanes, traffic lights, trees, and other surrounding objects near the streets. Especially, the CamVid [20] and the Highway Driving [100] datasets provide continuously labeled driving frames, which can be used for video semantic segmentation with temporal consistency evaluation. The Cityscapes dataset [47] focuses more on the data variation. It is larger in the number of images and the size of each image. Images are collected from 50 cities, making it closer to real-world complexity.

Regarding the remote sensing platforms, the number of datasets for semantic segmentation is much smaller, and the images are often captured in the nadir view, in which only the top of the objects can be seen. For the airborne imagery, the ISPRS 2D semantic labeling benchmarks [162] provide Vaihingen and the Potsdam datasets targeting on semantic labeling for the urban scenes. There are 6 classes defined for the semantic segmentation task, including impervious surfaces, building, low vegetation, tree, car ,and background clutter. The Vaihingen and the Potsdam datasets are 9 cm and 5 cm resolutions, respectively. Houston dataset [50] provides hyperspectral images (HSIs) and Light Detection And Ranging (LiDAR) data, both of which have 2.5m spatial resolutions, for the pixel level region classification. Zeebruges [27] provides a dataset with 7-tiles. There are 8 classes defined for both the land cover and the object classification. Besides the same 6 class types as in the ISPRS 2D semantic labeling datasets, additional boat and water classes are included. The RGB images are of 5 cm resolutions. For the satellite imagery, the DeepGlobe benchmarks [51] provide a semantic labeling dataset for the land cover classification with a pixel resolution of 50 cm. The images are of the sub-meter resolution, covering 7 classes, i.e., urban, agriculture, rangeland, forest, water, barren, and unknown. GID dataset [178] offers 4m resolution multispectral (MS)

satellite images from Gaofen-2 (GF-2) imagery for the land use classification. The target classes include 15 fine classes belonging to 5 major categories, which are built-up, farmland, forest, meadow, and water.

All the datasets above have had high impacts on the development of current state-of-the-art semantic segmentation methods. However, there are few high-resolution semantic segmentation datasets [92] based on UAV imagery with oblique views, which is supplemented with our UAVid dataset. The unmanned aerial vehicle (UAV) platform is more and more utilized for the earth observation. Compact and light-weighted UAVs are a trend for future data acquisition. The UAVs make image retrieval in large areas cheaper and more convenient, which allows quick access to useful information around a certain area. Distinguished from collecting images by satellites and airplanes, UAVs capture images from the sky with flexible flying schedules and higher spatial resolution, bringing the possibility to monitor and analyze landscape at specific locations and time swiftly.

The inherently fundamental applications for UAVs are surveillance [167, 150] and monitoring [199] in the target area. They have already been used for smart farming [129], precision agriculture [32], and weed monitoring [138], but few researches have been done for urban scene analysis. The semantic segmentation research for urban scenes could be the foundation for applications such as traffic monitoring, e.g., traffic jams and car accidents, population density monitoring and urban greenery monitoring, e.g., vegetation growth and damage. Although there are existing UAV datasets for detection, tracking, and behavior analysis [229, 59, 139, 160], to the best of our knowledge, there exists only one low altitude UAV semantic segmentation dataset before our UAVid, namely the Aeroscapes [92] dataset. Our UAVid dataset is comprised of much larger images that capture scenes in much larger range and with more scene complexity regarding the number of objects and object configurations, which make our UAVid dataset better for UAV urban scene understanding than the Aeroscapes dataset.

In this chapter, a new UAVid semantic segmentation dataset with high-resolution UAV images in oblique views has been brought out, which is designed for the semantic segmentation of urban scenes. We have brought out several challenges for the new dataset: the large scale variation between objects in different distances or of different categories, the moving object (separation of moving cars and static cars) recognition in the urban street scene and the preservation of the temporal consistency for better predictions across frames. These challenges mark the uniqueness of our dataset. In total, 300 high-resolution images from 30 video sequences are labeled with 8 object classes. The size of our dataset is ten times of the Vaihingen dataset [162], five times of the CamVid dataset [20] and twice of the Potsdam dataset [162] regarding the labeled number of pixels. All the labels are acquired with our in-house video labeler tool. Besides the provided image-label pairs, which are acquired with 0.2 FPS, unlabeled images are also provided with 20 FPS for users. The additional images are provided to aid the object recognition potentially. To provide performance references for the semantic labeling task and to test the usability of our dataset, several typical deep neural networks (DNNs) are utilized, including FCN-8s [128], Dilation net [216] and U-Net [161], which are widely used and stable for semantic segmentation task across different datasets. In addition, we propose a novel multi-scale-dilation net, which is useful to handle the problem of large scale variation that is prominent in the UAVid dataset. In order to benefit from

preserving the consistent prediction across the frames, an existing spatial-temporal regularization method (FSO [107]) is applied for post-processing. All the DNNs combined with FSO are evaluated as baselines.

By bringing the urban scene semantic segmentation task for the UAV platform, researchers could gain more insights for the visual understanding task in the UAV scenes, which could be the main foundations for higher level smart applications. As the data from UAVs has its own specialties, semantic segmentation task using UAV data deserves more attention.

Figure 2.1 Example images and labels from UAVid dataset. First row shows the images captured by UAV. Second row shows the corresponding ground truth labels. Third row shows the prediction results of MS-Dilation net+PRT+FSO model as in Tab. 2.1.

The rest of the chapter is organized as follows. Section 2.2 details how the UAVid dataset is built for the urban scene semantic segmentation, including the data specification, the class definition, the annotation methods, and the dataset splits. Section 2.3 presents the semantic labeling task for the UAVid dataset. The section involves the task illustration and the baseline methods for the task. Section 2.4 shows the corresponding experiment results with the analysis for the baseline methods. Lastly, section 2.5 provides the concluding remarks and the prospects for the UAVid dataset.

2.2 Dataset

Designing a UAV dataset requires careful thought about the data acquisition strategy, UAV flying protocol, and object class selection for annotation. The whole process is designed considering the usefulness and effectiveness for the UAV semantic segmentation research. In this section, the way to establish the dataset is illustrated. Section 2.2.1 shows the data acquisition strategy. Section 2.2.4 and 2.2.5 describe the classes for the task and the annotation methods respectively. Section 2.2.6 illustrates the data splits for the semantic segmentation task.

2.2.1 Data Specification

Our data acquisition and annotation methodology is designed for UAV semantic segmentation in complex urban scenes, featuring on both static and moving object recognition. In order to capture data that contributes the most towards researches on UAV scene understanding, the following features for the dataset are taken into consideration.

- Oblique view. For the UAV platform, it is natural to record images or videos in an oblique view or a nadir view style. Nadir view is more common for satellite images as the distance between the camera and the ground is often far. Nadir view brings invariance to the representation of objects in the image as only the top can be observed. However, the limited representation may also bring confusion in recognition among different objects. In contrast, an oblique view gives a more diverse representation of objects, which can be helpful for recognition. It also observes a larger area with details when flying close to the ground, which causes large scale variation across an image. In order to observe in an oblique view, the camera angle is set to around 45 degrees to the vertical direction.

- High resolution. We adopt 4K resolution video recording mode with safe flying height around 50 meters. The image resolution is either 4096×2160 or 3840×2160. In this setting, it is visually clear enough to differentiate most of the objects. Objects that are horizontally far away could also be detected. In addition, it is even possible to detect humans that are near to the UAV.

- Consecutive labeling. Our dataset is designed for the semantic segmentation task. We prefer to label images in a sequence, where the prediction stability could be evaluated. As it is too expensive to label densely in the temporal space, we label 10 images with 5 seconds interval in each sequence.

- Complex and dynamic scenes with diverse objects. Our dataset aims at achieving real-world scene complexity, where there are both static and moving objects. Scenes near streets are chosen for the UAVid dataset as they are complex enough with more dynamic human activities. A variety of objects appear in the scene such as cars, pedestrians, buildings, roads, vegetation, billboards, light poles, traffic lights, and so on. We fly UAVs with an oblique view along the streets or across different street blocks to acquire such scenes.

- Data variation. In total, 30 small UAV video sequences are captured in 30 different places to bring variance to the dataset, relieving learning algorithms from over-fitting. Data acquisition is performed in good weather conditions with sufficient illumination. We believe that data acquired in dark environments or other weather conditions like snowing or raining require special processing techniques, which are not the focus of our current dataset.

DJI phantom3 pro and DJI phantom4 are used for flying, which are light weighted modern drones. The UAVs fly steadily with a maximum flying speed of 10 m/s, preventing potential blurry effect caused by motions. The default cameras mounted on the UAVs are used for video acquisition with only RGB channels (see Fig. 2.1).

2.2.2 Scene Complexity

The scene complexity [47] of the new UAVid dataset is higher than the other existing UAV semantic segmentation dataset [92] regarding the number of objects and the different object configurations. We should note that both datasets lack the instance labeling for the quantitative scene complexity calculation [47]. It is still qualitatively evident that our UAVid dataset has much higher scene complexity. We have, on average, 9 times more car objects and 3 times more human objects per unit of image area by manually counting in a random subset of images from the two datasets. Examples of the street scenes are shown in Fig. 2.2.

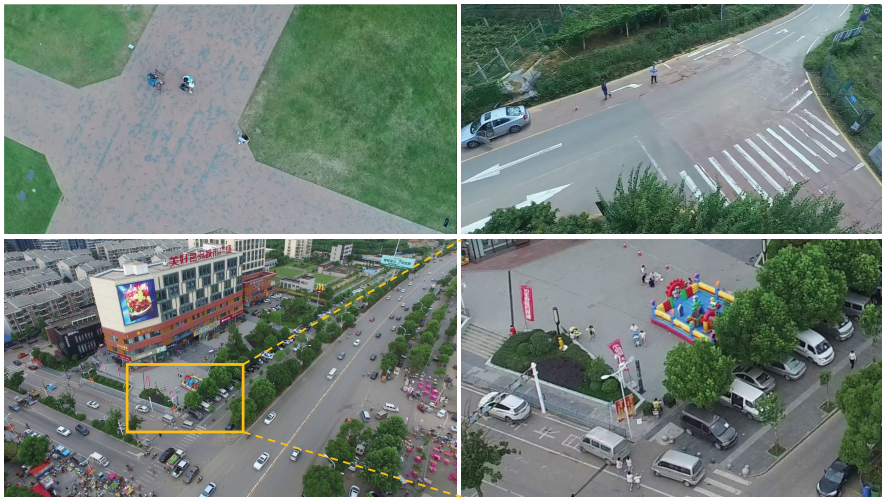

Figure 2.2 Comparison between the Aeroscapes dataset [92] and the UAVid dataset. The first row shows the examples from Aeroscapes dataset. The second row shows the examples from the UAVid dataset, in which the right column shows an image crop at the original scale, where detailed object can be clearly seen. Regarding the number of objects and different object configurations, the UAVid dataset has higher scene complexity.

2.2.3 Dataset Size

Our UAVid dataset has 300 images and each of size 4096×2160 or 3840×2160. To compare the sizes of different datasets for semantic segmentation fairly, we should consider not only the number of images, but also the size of each image. A more fair metric is to compare the number of labeled pixels in total. We select several well-known semantic segmentation datasets for comparisons. The CamVid dataset [20] has 701 images of size 960×720, which is only one fifth of our dataset in terms of the number of labeled pixels. The giant Cityscapes dataset [47] has $5,000$ images of size 2048×1024, which is 4 times the size of our UAVid dataset. However, many objects in our images are smaller than theirs, providing more object variance in the same number of pixels, which compensate for the object recognition task in a degree. Compared to the ISPRS 2D semantic labeling datasets,

the Vaihingen and the Potsdam datasets [162] have even fewer images, 33 and 38 images respectively, but the size of each image is quite large, e.g., 6000×6000 for the Potsdam dataset. Regarding the total number of the labeled pixels, the Vaihingen and the Potsdam datasets are only one tenth and one half the size of our UAVid dataset, respectively. The state-of-the-art DeepGlobe Land Cover Classification dataset [51] has 1146 satellite images for rural areas of size 2448×2448, which is about 2.5 times the size of our dataset. However, the scene complexity of the rural area is much lower than it is in our UAVid dataset. In conclusion, our UAVid dataset has a moderate size, and it is bigger than several well-known semantic segmentation datasets. Section 2.4 has shown that deep learning methods can achieve satisfactory qualitative and quantitative results for experimental purposes, which further proves the usability of our UAVid dataset.

2.2.4 Class Definition and Statistical Analysis

Fully label all types of objects in the street scene in a 4K UAV image is very expensive. As a consequence, only the most common and representative types of objects are labeled for our current dataset. In total, 8 classes are selected for the semantic segmentation, i.e., building, road, tree, low vegetation, static car, moving car, human, and clutter. Example instances from different classes are shown in Fig. 2.3. The definition of each class is described as follows.

1. building: living houses, garages, skyscrapers, security booths, and buildings under construction. Freestanding walls and fences are not included.
2. road: road or bridge surface that cars can run on legally. Parking lots are not included.
3. tree: tall trees that have canopies and main trunks.
4. low vegetation: grass, bushes and shrubs.
5. static car: cars that are not moving, including static buses, trucks, automobiles, and tractors. Bicycles and motorcycles are not included.
6. moving car: cars that are moving, including moving buses, trucks, automobiles, and tractors. Bicycles and motorcycles are not included.
7. human: pedestrians, bikers, and all other humans occupied by different activities.
8. clutter: all objects not belonging to any of the classes above.

We deliberately divide the car class into moving car and static car classes. Moving car is such a special class designed for moving object segmentation. Other classes can be inferred from their appearance and context, while the moving car class may need additional temporal information in order to be appropriately separated from static car class. Achieving high accuracy for both static and moving car classes is one possible research goal for our dataset.

The number of pixels in each of the 8 classes from all 30 sequences is reported in Fig. 2.4. It clearly shows the unbalanced pixel number distribution of different classes. Most of the pixels are from classes like building, tree, clutter, road, and low vegetation. Fewer pixels are from moving car and static car classes, which are both fewer than 2% of the total pixels. For human class, it is almost zero, fewer

Figure 2.3 Example instances from different classes. The first row shows the cropped instances. The second row shows the corresponding labels. From left to right, the instances are building, road, static car, tree, low vegetation, human and moving car respectively.

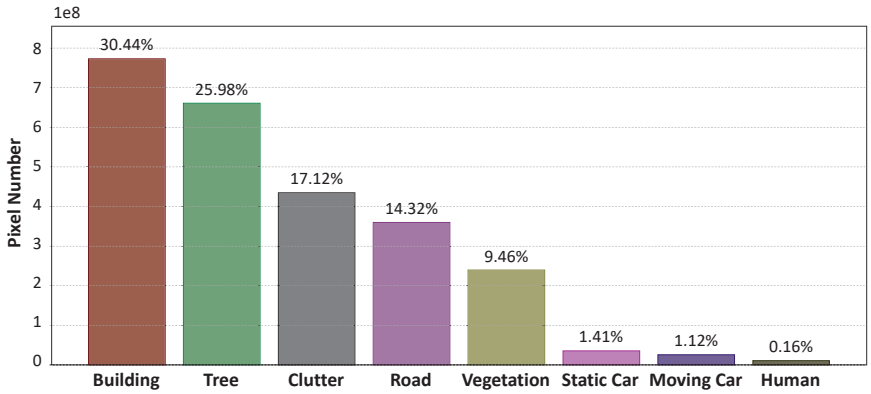

Figure 2.4 Pixel number histogram.

than 0.2% of the total pixels. Smaller pixel number is not necessarily resulted by fewer instances, but the size of each instance. A single building can take more than 10k pixels, while a human instance in the image may only take fewer than 100 pixels. Normally, classes with too small pixel numbers are ignored in both training and evaluation for semantic segmentation task [47]. However, we believe humans and cars are important classes that should be kept in street scenes rather than being ignored.

2.2.5 Annotation Method

We have provided densely labeled fine annotations for high-resolution UAV images. All the labels are acquired with our own labeler tool. It takes approximately 2 hours to label all pixels in one image. Pixel level, super-pixel level, and polygon level annotation methods are provided for annotators, as illustrated in Fig.2.5. For super-pixel level annotation, our method employs a similar strategy as the COCO-Stuff [25] dataset. We first apply SLIC method [2] to partition the image into super-pixels, each of which is a group of pixels that are spatially connected and share similar characteristics, such as color and texture. The pixels within the same super-pixel

are labeled with the same class. Super-pixel level annotation can be useful for the objects with sawtooth boundaries like trees. We offer super-pixel segmentation of 4 different scales for annotators to best adjust to objects of different scales. Polygon annotation is more useful to annotate objects with straight boundaries like buildings, while pixel level annotation serves as a basic annotation method. Our tool also provides video play functionality around certain frames to help to inspect whether certain objects are moving or not. As there might be overlapping objects, we label the overlapping pixels to be the class that is closer to the camera.

Figure 2.5 Annotation methods. Left shows pixel level annotation, middle shows super-pixel level annotation, and right shows polygon level annotation.

2.2.6 Dataset Splits

The whole 30 densely labeled video sequences are divided into training, validation, and test splits. We do not split the data completely randomly, but in a way that makes each split to be representative enough for the variability of different scenes. All three splits should contain all classes. Our data is split at the sequence level, and each sequence comes from a different scene place. Following this scheme, we get 15 training sequences (150 labeled images) and 5 validation sequences (50 labeled images) for training and validation splits, respectively, whose annotations will be made publicly available. The test split consists of the left 10 sequences (100 labeled images), whose labels are withheld for benchmarking purposes. The size ratios among training, validation and test splits are 3:1:2.

2.3 Semantic Labeling Task

In this section, the semantic labeling task for our dataset is introduced. The task details and the evaluation metric for the UAVid dataset are introduced first in section 2.3.1. The following sections (from 2.3.2 to 2.3.5) introduce the baseline methods for the task. The baseline methods are presented in company with the task to offer performance references and to test the usability of the dataset for the task. Section 2.3.2 and section 2.3.3 introduce the deep neural networks in the baseline methods. Section 2.3.4 and section 2.3.5 introduce the pre-training and the spatial-temporal regularization respectively, which boost the performance for all baseline methods.

2.3.1 Task and Metric

The task defined on the UAVid dataset is to predict pixel level semantic labeling for the UAV images. The image-label pairs are provided for each sequence together with the unlabeled images. Currently, the UAVid dataset only supports image level semantic labeling without instance level consideration. The semantic labeling performance is assessed based on the standard mean IoU metric [60]. The goal for this task is to achieve as a high mean IoU score as possible. For the UAVid dataset, the clutter class has a relatively large pixel number ratio and consists of meaningful objects, which is taken as one class for both training and evaluation rather than being ignored.

2.3.2 Deep neural networks for baselines

In order to offer performance references and to test the usability of our UAVid dataset for the semantic labeling task, we have tested several deep learning models for the single image prediction. Although static cars and moving cars cannot be differentiated by their appearance from only one image, it is still possible to infer based on their context. The moving cars are more likely to appear in the center of the road, while the static cars are more likely to be at the parking lots or to the side of the roads. As the UAVid dataset consists of very complex street scenes, it requires powerful algorithms like deep neural networks for the semantic labeling task. We start with 3 widely used deep fully convolutional neural networks, they are FCN-8s [128], Dilation net [216] and U-Net [161].

FCN-8s [128] has often been a good baseline candidate for semantic segmentation. It is a giant model with strong and effective feature extraction ability, but yet simple in structure. It takes a series of simple 3x3 convolutional layers to form the main parts for high-level semantic information extraction. This simplicity in structure also makes FCN-8s popular and widely used for semantic segmentation.

Dilation net [216] has a similar front end structure with FCN-8s, but it removes the last two pooling layers in VGG16. Instead, convolutions in all following layers from the conv5 block are dilated by a factor of 2 due to the ablated pooling layers. Dilation net also applies a multi-scale context aggregation module in the end, which expands the receptive field to boost the performance for prediction. The module is achieved by using a series of dilated convolutional layers, whose dilation rate gradually expands as the layer goes deeper.

U-Net [161] is a typical symmetric encoder-decoder network originally designed for segmentation on medical images. The encoder extracts features, which are gradually decoded through the decoder. The features from each convolutional block in the encoder are concatenated to the corresponding convolutional block in the decoder to acquire features of higher and higher resolution for prediction gradually. U-Net is also simple in structure but good at preserving object boundaries.

2.3.3 Multi-Scale-Dilation Net

For a high-resolution image captured by a UAV in an oblique view, the sizes of objects in different distances can vary dramatically. Figure 2.7 illustrates such a scale problem in the UAVid dataset. The large scale variation in a UAV image

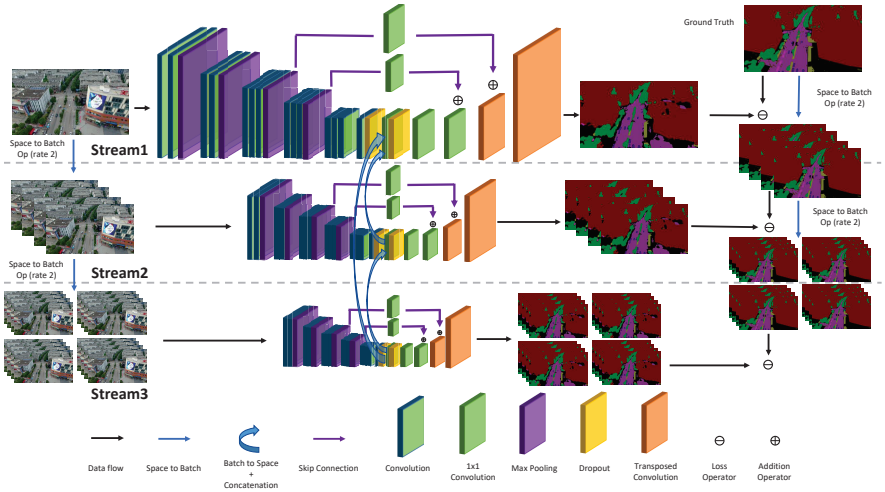

Figure 2.6 Structure of the proposed Multi-Scale-Dilation network. Three scales of images are achieved by Space to Batch operation with rate 2. Standard convolutions in stream2 and stream3 are equivalent to dilated convolutions in stream1. The main structure for each stream is FCN-8s [128], which could be replaced by any other networks. Features are aggregated at conv7 layer for better prediction on finer scales.

can affect the accuracy of prediction. In a network, each output pixel in the final prediction layer has a fixed receptive field, which is formed by pixels in the original image that can affect the final prediction of that output pixel. When the objects are too small, the neural network may learn the noise from the background. When the objects are too big, the model may not acquire enough information to infer the label correctly. The above is a long-standing notorious problem in computer vision. To reduce such a large scale variation effect, we propose a novel multi-scale-dilation net (MS-Dilation net) as an additional baseline.

One way to expand the receptive field of a network is to use dilated convolution [216]. Dilated convolution can be implemented in different ways, one of which is to leverage on space to batch operation (S2B) and batch to space operation (B2S), which is provided in Tensorflow API. Space to batch operation outputs a copy of the input tensor where values from the height and width dimensions are moved to the batch dimension. Batch to space operation does the inverse. A standard 2D convolution on the image after S2B is the same as a dilated convolution on the original image. A single dilated convolution can be performed as $S2B- > convolution- > B2S$. This implementation for dilated convolution is efficient when there is a cascade of dilated convolutions, where intermediate S2B and B2S cancel out. For instance, 2 consecutive dilated convolution with the same dilation rate can be performed as $S2B- > convolution- > convolution- > B2S$.

Space to batch operation can also be taken as a kind of nearest neighbor down-sampling operation, where the input is the original image while the outputs are down-sampled images with slightly different spatial shifts. The nearest neighbor down-sampling operation is nearly equivalent to space to batch operation, where the only difference is the number of output batches. With the above illustration, it

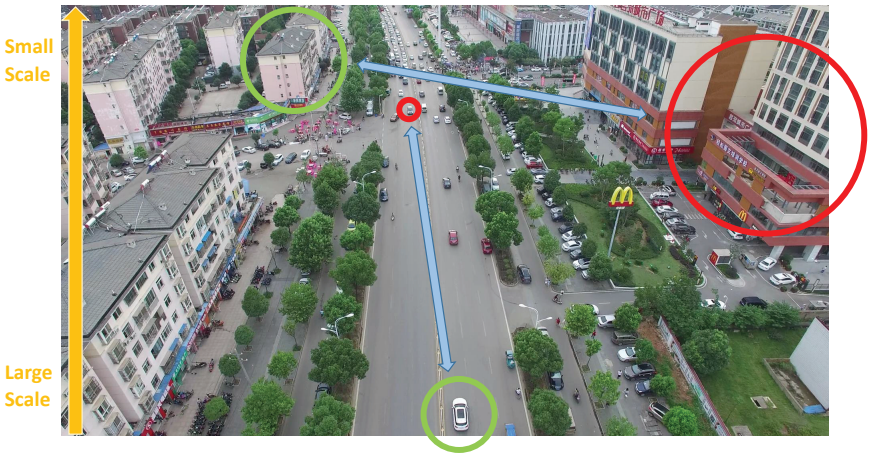

Figure 2.7 Illustration of the scale problem in an UAV image. The scales of the objects vary greatly from the bottom to the top of the image. The green circles mark the objects in proper scales while the red circles mark the objects in either too large or too small scales.

is easy to draw the connection between the dilated convolution and the standard convolution on down-sampled images.

By utilizing space to batch operation and batch to space operation, semantic segmentation can be done on different scales. In total, three streams are created for three scales, as shown in Fig. 2.6. For each stream, a modified FCN-8s is used as the main structure, where the depth for each convolutional block is reduced due to the memory limitation. Here, filter depth is sacrificed for more scales. In order to reduce detail loss in feature extraction, the pooling layer in the fifth convolutional block is removed to keep a smaller receptive field. Instead, features with larger receptive fields from other streams are concatenated to higher resolution features through skip connection in conv7 layers. Note that these skip connections need batch to space operation to retain spatial and batch number alignment. In this way, each stream handles feature extraction in its own scale, and features from larger scales are aggregated to boost prediction for higher resolution streams.

Multiple scales may also be achieved by down-sampling images directly [3]. However, there are 3 advantages to our multi-scale processing. First, every pixel is assigned to one batch in space to batch operation, and all the labeled pixels shall be used for each scale with no waste. Second, there is strict alignment between image-label pairs in each scale as there is no mixture of image pixels nor a mixture of label pixels. Finally, the concatenated features in the conv7 layer are also strictly aligned.

For each scale, corresponding ground truth labels can also be generated through space to batch operation in the same way as the generation for input images in different streams. With ground truth labels for each scale, deeply supervised training can be done. The losses in three scales are all cross entropy loss. The loss in stream1 is the target loss, while the losses in stream2 and stream3 are auxiliary. The final loss to be optimized is the weighted mean of the three losses, shown in the equations

below. $m1, m2, m3$ are the numbers of pixels of an image in each stream. n is the batch index, and t is the pixel index. p is the target probability distribution of a pixel, while q is the predicted probability distribution.

$$CE_1 = \frac{1}{m_1} \sum_{t=1}^{m_1} -p_t \log(q_t) \tag{2.1}$$

$$CE_2 = \frac{1}{4m_2} \sum_{n=1}^{4} \sum_{t=1}^{m_2} -p_t^n \log(q_t^n) \tag{2.2}$$

$$CE_3 = \frac{1}{16m_3} \sum_{n=1}^{16} \sum_{t=1}^{m_3} -p_t^n \log(q_t^n) \tag{2.3}$$

$$Loss = \frac{w_1 \times CE_1 + w_2 \times CE_2 + w_3 \times CE_3}{w_1 + w_2 + w_3} \tag{2.4}$$

It is also interesting to note that every layer becomes a dilated version for stream2 and stream3, especially for the pooling layer and the transposed convolutional layer, which turn into a dilated pooling layer and a dilated transposed convolutional layer respectively. Compared to layers in stream1, layers in stream2 are dilated by rate of 2, and layers in stream3 are dilated by rate of 4. Theses 3 streams together form the MS-Dilation net.

2.3.4 Fine-tune Pre-trained Networks

Due to the limited size of our UAVid dataset, training from scratch may not be enough for the networks to learn diverse features for better label prediction. Pre-training a network has been proved to be very useful for various benchmarks [126, 24, 37, 224], which boosts the performance by utilizing more data from other datasets. To reduce the effect of limited training samples, we also explore how much pre-training a network can boost the score for the UAVid semantic labeling task. We pre-train all the networks with the cityscapes dataset [47], which comprises many more images for training.

2.3.5 Spatial-temporal Regularization for Semantic Segmentation

For semantic labeling task, we further explore how a spatial-temporal regularization can improve the prediction. Taking advantage of temporal information is valuable for label prediction for sequence data. Normally, deep neural networks trained on individual images cannot provide completely consistent predictions spanning several frames. However, different frames provide observations from different viewing positions, through which multiple clues can be collected for object prediction. To utilize temporal information in the UAVid dataset, we adopt feature space optimization (FSO) [107] method for sequence data prediction. It smooths the final label

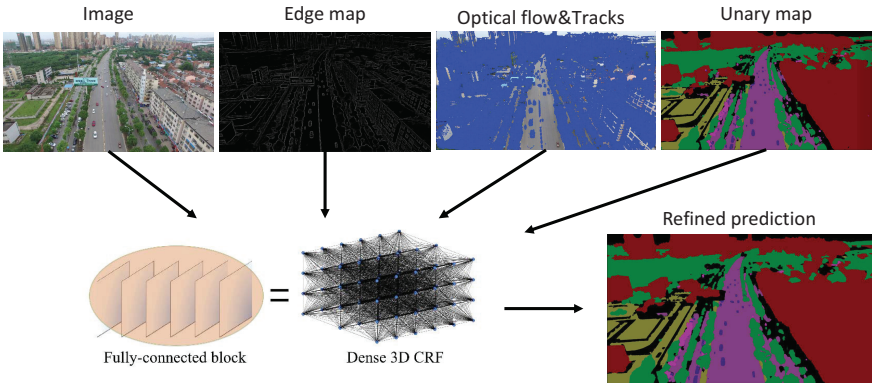

Figure 2.8 The data inputs for the FSO [107] post-processing method. The image, the edge map, the unary map, the optical flow and tracks are required for the method. The edge map shows the probability of each pixel to be an edge. The blue points in the image of optical flow&Tracks mark the points being tracked. The unary map is the class probabilities for each pixel predicted by the deep neural networks.

prediction for the whole sequence by applying 3D CRF covering both spatial and temporal domains. The method takes advantage of optical flow and tracks to link the pixels in the temporal domain. The whole post-processing requires multiple data inputs, including the image, the unary map from the deep neural networks, the edge map, the optical flow, and the tracks as shown in Fig. 2.8.

2.4 Experiments

Our experiments are divided into 3 parts. Firstly, we compare semantic segmentation results by training deep neural networks from scratch. These results serve as the basic baselines. Secondly, we analyze how pre-trained models can be useful for the UAVid semantic labeling task. We fine-tune deep neural networks with UAVid dataset after they are pre-trained on the cityscapes dataset [47]. Finally, we explore the influence of spatial-temporal regulation by using the FSO method for semantic video segmentation.

The size of our UAV images is very large, which requires too much GPU memory for intermediate feature storage in deep neural networks. As a result, we crop each UAV image into 9 evenly distributed smaller overlapped images that cover the whole image for training. Each cropped image is of size 2048×1024. We keep such a moderate image size in order to reduce the ratio between the zero padding area and the valid image area. Bigger image size also resembles a larger batch size when each pixel is taken as a training sample. During testing, the average prediction scores are used for the overlapped area. Fig. 2.9 illustrates the way of cropping.

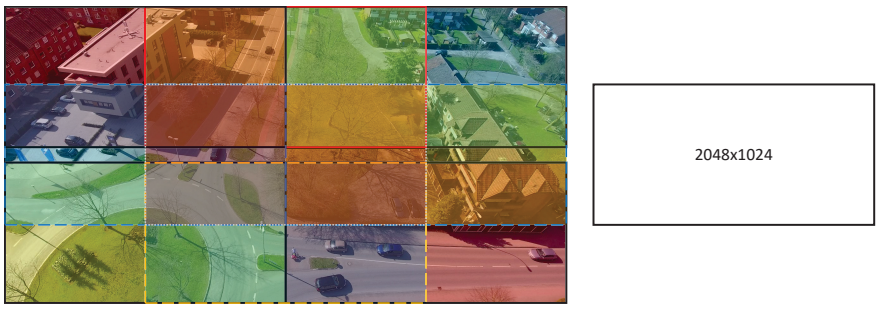

Figure 2.9 Image cropping illustration. The 4K image is cropped to 9 evenly distributed smaller overlapped images before processing.

Table 2.1 IoU scores for different models. IoU scores are reported in percentage and best results are shown in bold. **PRT** stands for pre-train and **FSO** stands for feature space optimization [107].

Model	Building	Tree	Clutter	Road	Low Vegetation	Static Car	Moving Car	Human	mean IoU
FCN-8s	64.3	63.8	33.5	57.6	28.1	8.4	29.1	0.0	35.6
Dilation Net	72.8	66.9	38.5	62.4	34.4	1.2	36.8	0.0	39.1
U-Net	70.7	67.2	36.1	61.9	32.8	11.2	**47**.5	0.0	40.9
MS-Dilation (ours)	**74**.3	**68**.1	**40**.3	**63**.5	**35**.5	**11**.9	42.6	0.0	**42**.0
FCN-8s+PRT	77.4	72.7	44.0	63.8	45.0	19.1	49.5	0.6	46.5
Dilation Net+PRT	**79**.8	73.6	44.5	64.4	44.6	24.1	53.6	0.0	48.1
U-Net+PRT	77.5	73.3	44.8	64.2	42.3	**25**.8	**57**.8	0.0	48.2
MS-Dilation (ours)+PRT	79.7	**74**.6	**44**.9	**65**.9	**46**.1	21.8	57.2	**8**.0	**49**.8
FCN-8s+PRT+FSO	78.6	73.3	45.3	64.7	46.0	19.7	49.8	0.1	47.2
Dilation Net+PRT+FSO	80.7	74.0	45.4	65.1	45.5	24.5	53.6	0.0	48.6
U-Net+PRT+FSO	79.0	73.8	**46**.4	65.3	43.5	**26**.8	56.6	0.0	48.9
MS-Dilation (ours)+PRT+FSO	**80**.9	**75**.5	46.3	**66**.7	**47**.9	22.3	**56**.9	**4**.2	**50**.1

2.4.1 Train from Scratch

To have a fair comparison among different networks, we re-implement all the networks with Tensorflow [1], and all networks are trained with an Nvidia Titan X GPU. In order to accommodate the networks into 12G GPU memory, depth of some layers in the Dilation net, U-Net, and MS-Dilation net are reduced to fit into the memory maximally. The model configuration details of different networks are shown in Fig. 2.10.

The neural networks share similar hyper-parameters for training from scratch. All models are trained with the Adam optimizer for 27K iterations (20 epochs). The base learning rate is set to 10^{-4} exponentially decaying to 10^{-7}. Weight decay for all weights in convolutional kernels is set to 10^{-5}. Training is done with one image per batch. For data augmentation in training, we apply left-to-right flip randomly. We also apply a series of color augmentation, including random hue operation, random contrast operation, random brightness operation, random saturation operation.

Auxiliary losses are used for our MS-Dilation net. The loss weights for three streams are set to 1.8, 0.8, and 0.4 empirically. The loss weights for stream2 and stream3 are set smaller than stream1 as the main goal is to minimize the loss in stream1. For the Dilation net, the basic context aggregation module is used and initialized as it is in [216]. All networks are trained end-to-end, and their mean IoU scores are reported in percentage, as shown in Tab. 2.1.

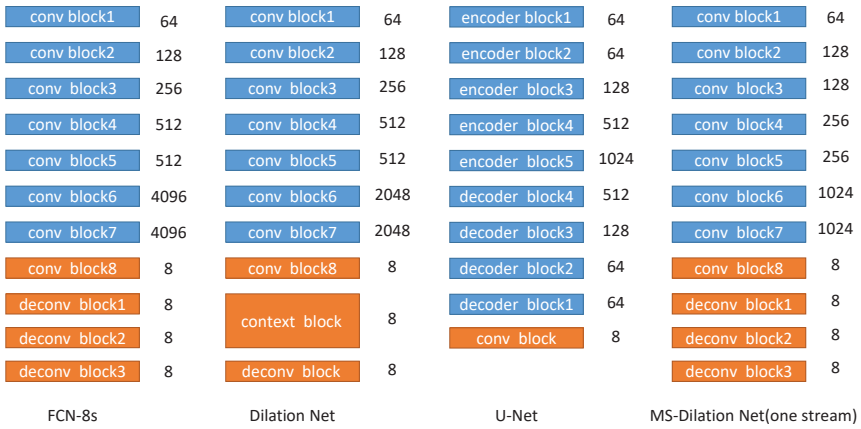

Figure 2.10 The configuration of different models. The blue blocks are the feature extraction part, while the orange blocks are the context aggregation and the prediction part for the corresponding 8 classes in UAVid dataset.

For all the four networks, they are better at discriminating building, road, and tree classes, achieving IoU scores higher than 50%. The scores for car, vegetation, and clutter classes are relatively lower. All four networks completely fail to discriminate human class. Normally, classes with larger pixel number have relatively higher IoU scores. However, the IoU score for the moving car class is much higher than the static car class, even though the two classes have similar pixel numbers. The reason may be that static cars may appear in various contexts like parking lots, garages, sidewalks, or partially blocked under the trees, while moving cars often run in the middle of roads with a very clear view.

The Dilation net and the U-Net perform similarly, and they both outperform the FCN-8s. The FCN-8s extracts features on a single scale, while the Dilation net and U-Net benefit from features in better scales from the context blocks and multiple decoders in multiple scales, respectively. Our Multi-Scale-Dilation net differs as it extracts features in multiple scales from very early and shallow layers, and it achieves the best mean IoU score and the best IoU scores for most of the classes among the four networks. It shows the effectiveness of multi-scale feature extraction.

2.4.2 Fine-tune Pre-trained Models

For fine-tuning, all the networks are pre-trained with cityscapes dataset [47]. Finely annotated data from both training and validation splits are used, which is of 3,450 densely labeled images in total. Hyper-parameters and data augmentation are set the same as they are in section 2.4.1, except that the iteration is set to 52K. Next, all the networks are fine-tuned with data from the UAVid dataset. As there is still large heterogeneity between these two datasets, all layers are trained for all networks. We only initialize feature extraction parts of the networks with pre-trained models,

Table 2.2 IoU scores for different training strategies. IoU scores are reported in percentage and best results are shown in bold. **w** stands for with and **w/o** stands for without.

Model	Building	Tree	Clutter	Road	Low Vegetation	Static Car	Moving Car	Human	mean IoU
fine-tune w/o auxiliary loss	78.5	72.2	44.0	65.3	43.5	17.4	51.5	1.2	46.7
fine-tune w auxiliary loss	79.2	72.5	**44**.8	64.6	44.3	17.0	52.8	3.4	47.3
fine-tune w+w/o auxiliary loss	**79**.4	**73**.1	43.7	**65**.5	**45**.3	**21**.3	**55**.8	**6**.3	**48**.8

while the prediction parts are initialized the same as training from scratch. The learning rate is set to 10^{-5} exponentially decaying to 10^{-7} for FCN-8s, and 10^{-4} exponentially decaying to 10^{-7} for other 3 networks as they are more easily stuck at a local minimum with initial learning rate to be 10^{-5} during training. The rest of the hyper-parameters are set the same as training from scratch. The performance is also shown in Tab. 2.1.

To find out whether auxiliary losses are important, we have fine-tuned MS-Dilation net with 3 different training plans. For the first plan, we fine-tune the MS-Dilation net without auxiliary losses for 30 epochs by setting loss weights to 0 in stream2 and stream3. For the second plan, we fine-tune the MS-Dilation net with auxiliary losses for 30 epochs. For the final plan, we fine-tune the MS-Dilation net with auxiliary losses for 20 epochs and without auxiliary losses for another 10 epochs. The IoU scores for three plans are shown in Tab. 2.2. As it is shown, the best mean IoU score is achieved by the third plan. The better result for MS-Dilation net+PRT in Tab. 2.1 is achieved by fine-tuning 20 epochs without auxiliary losses after fine-tuning 20 epochs with auxiliary losses.

Clearly, auxiliary losses are very important for the MS-Dilation net. However, neither purely fine-tuning the MS-Dilation net with auxiliary losses nor without achieves the best score. It is the combination of these two fine-tuning processes that brings the best score. Auxiliary losses are important as they can guide the multi-scale feature learning process, but the network needs to be further fine-tuned without auxiliary losses to get the best multi-scale filters for prediction.

By fine-tuning the pre-trained models, the performance boost is huge for all networks across all classes except human class. The networks still struggle to differentiate human class. Nevertheless, the improvement is evident for the MS-Dilation net with 8% improvement. Decoupling the filters with different scales can be beneficial when objects appear in large scale variation.

In order to see the effect of multiple-scale processing, the qualitative performance comparisons among FCN-8s, Dilation net, U-Net, and MS-Dilation Net are presented in Fig. 2.11. By utilizing features in multiple scales, the MS-Dilation Net gives relatively better prediction for the roundabout. Locally, the road may be wrongly classified to be building due to the simple texture. However, by aggregating information from multiple scales in MS-Dilation Net, the relatively better label can be predicted.

2.4.3 Spatial-temporal Regularization for Semantic Segmentation

For spatial-temporal regularization, we apply methods used in feature space optimization (FSO) [107]. As FSO process a block of images simultaneously, a block of 5 consecutive frames with a gap of 10 frames are extracted from the provided video files, and the test image is located at the center in each block. The gap between

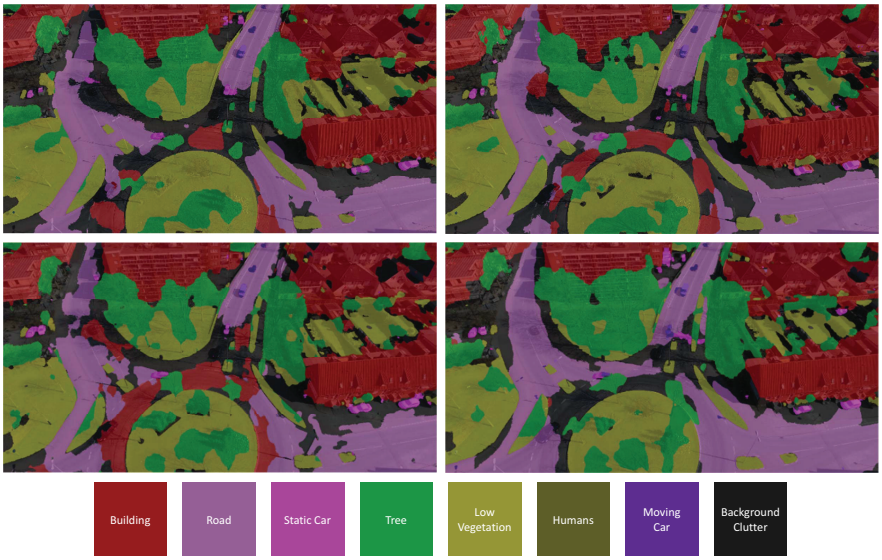

Figure 2.11 Prediction example of FCN8s (top left), Dilation Net (top right), U-Net (bottom left) and MS-Dilation Net (bottom right).

consecutive frames is not too big in order to get good flow extraction. It is better to have longer sequences to gain longer temporal regularization, but due to memory limitation, it is not possible to support more than 5 images in a 30G memory without sacrificing the image size.

The FSO process in each block requires several ingredients. Contour strength for each image is calculated according to [56]. The unary for each image is set as the softmax layer output from each fine-tuned network. Forward flows and backward flows are calculated according to [22, 23]. As the computation speed for optical flow at the original image scale is extremely low, the images to be processed are downsized by 8 times for both width and height, and the final flows at the original scale are calculated through bicubic interpolation and magnification. Then, points trajectories can be calculated according to [173] with the forward and backward flows. Finally, a dense 3D CRF is applied after feature space optimization as described in [107].

The IoU scores for FSO post-processing with unaries from different fine-tuned networks are reported in Tab. 2.1. For each model, there is around 1% IoU score improvement for each individual class except for human and moving car classes. FSO favors more for the class whose instance covers more image pixels. The IoU score improves less for the class with smaller instances like static car class, and it drops for moving car class and human class. The IoU score of the human class for MS-Dilation net drops by a large margin, nearly 4%. An example of refinement is shown in Fig. 2.12.

In addition, qualitative prediction examples of different configurations across different time index are shown in Fig.2.13. Temporal consistency can be evaluated by viewing one row of the figure. Different model settings can be evaluated by

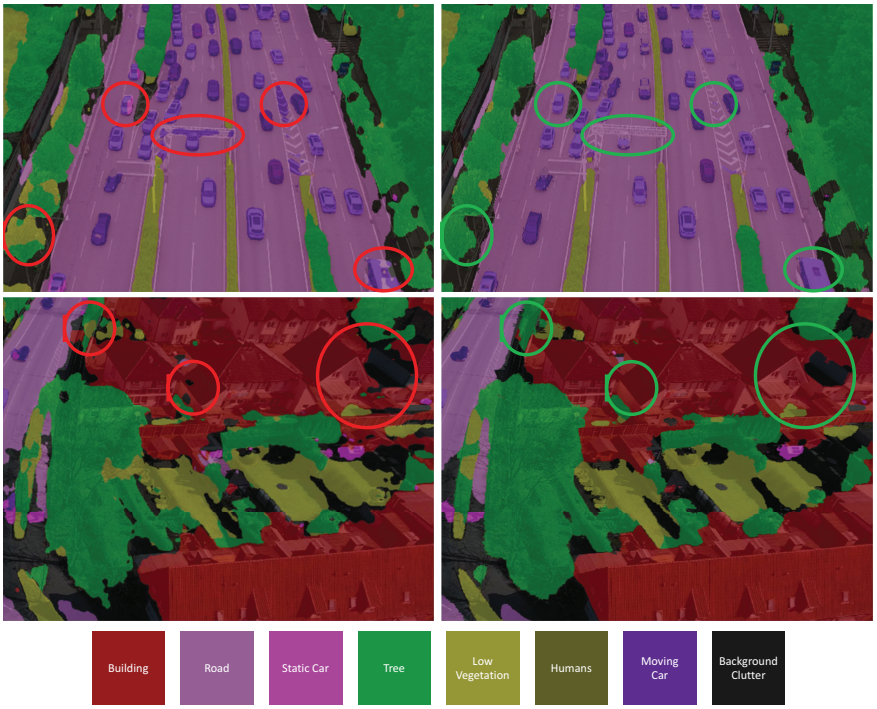

Figure 2.12 Examples of spatial-temporal regularization for UAVid image semantic segmentation. The left column shows the prediction without FSO plus 3D CRF refinement. The right column shows the corresponding refined prediction with FSO plus 3D CRF refinement. The most obvious improvements are high-lighted with circles. The spatial-temporal regularization achieves a more coherent prediction for different objects.

viewing one column of the figure.

2.5 Conclusions

In this chapter, we have presented a new UAVid dataset to advance the development of semantic segmentation in urban street scenes from UAV images. Our dataset has brought out several challenges for the semantic segmentation task, including the large scale variation for different objects, the moving object recognition in the street scenes, and the temporal consistency across multiple frames. Eight classes for the semantic labeling task have been defined and labeled. The usability of our UAVid dataset has also been proved with several deep convolutional neural networks, among which the proposed Multi-Scale-Dilation net performs the best via multi-scale feature extraction. It has also been shown that pre-training the network and applying the spatial-temporal regularization are beneficial for the UAVid semantic labeling task. Although the UAVid dataset has some limitations in the size and the number of classes compared to the biggest dataset in the computer vision community,

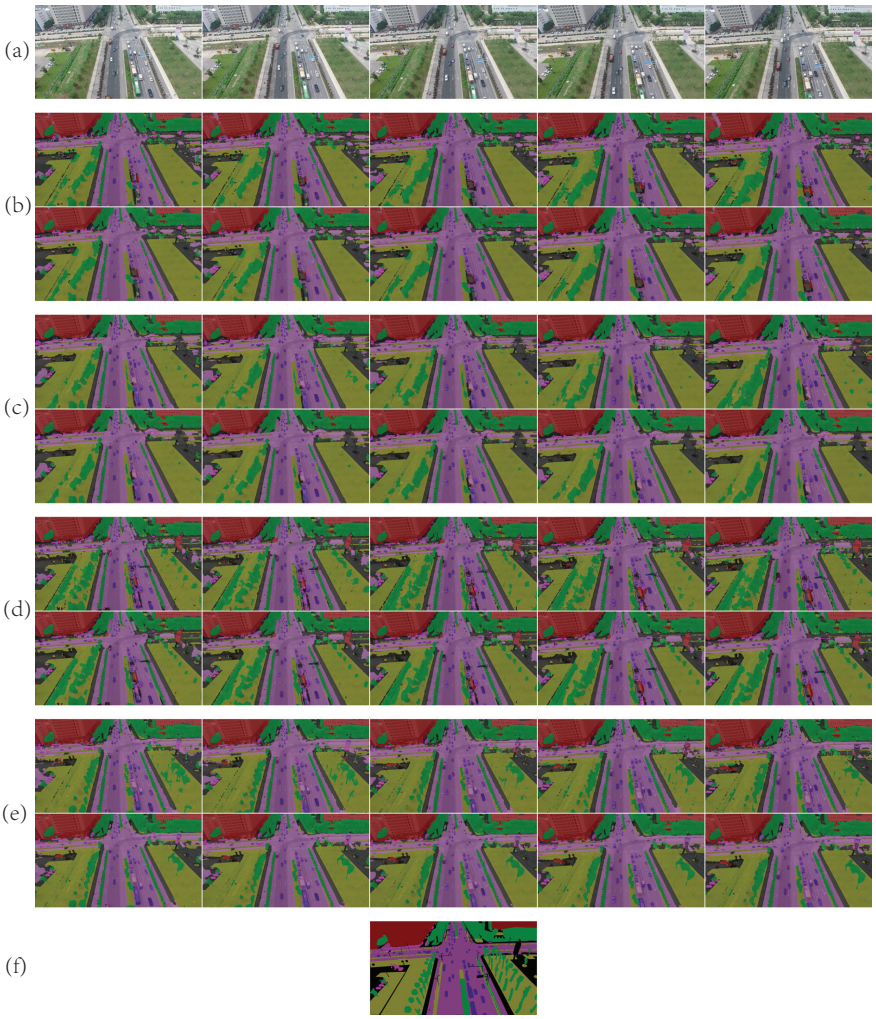

Figure 2.13 Example prediction for sequence images. The 1st row block (a) presents the original images in sequential order from left to right. The last row block (f) presents the ground truth label for the test image located in the middle of the sequence. The 2nd, 3rd, 4th and 5th row blocks (b,c,d,e) present the prediction results of different models as it is in Tab. 2.1. Two rows from block (b) present prediction of FCN-8s+PRT and FCN-8s+PRT+FSO respectively. Two rows from block (c) present prediction of Dilation Net+PRT and Dilation Net+PRT+FSO respectively. Two rows from block (d) present prediction of U-Net+PRT and U-Net+PRT+FSO respectively. Two rows from block (e) present prediction of MS-Dilation Net+PRT and MS-Dilation Net+PRT+FSO respectively. PRT and FSO are defined the same as in Tab. 2.1.

the UAVid dataset can already be used for benchmark purposes. In the future, we would like to further expand the dataset in size and the number of categories to make it more challenging and useful to advance the semantic segmentation research for the UAV imagery.

Bidirectional Multi-scale Attention Networks for Semantic Segmentation

3

Abstract

Semantic segmentation for aerial platforms has been one of the fundamental scene understanding task for the earth observation. Most of the semantic segmentation research focused on scenes captured in nadir view, in which objects have relatively smaller scale variation compared with scenes captured in oblique view. The huge scale variation of objects in oblique images limits the performance of deep neural networks (DNN) that process images in a single scale fashion. In order to tackle the scale variation issue, in this chapter, we propose the novel bidirectional multi-scale attention networks, which fuse features from multiple scales bidirectionally for more adaptive and effective feature extraction. The experiments are conducted on the UAVid2020 dataset and have shown the effectiveness of our method. Our model achieved the state-of-the-art (SOTA) result with a mean intersection over union (mIoU) score of 70.80%.

3.1 Introduction

Semantic segmentation has been one of the most fundamental research tasks for scene understanding. It is to assign each pixel within an image with the class label it belongs to. There have been many works for semantic segmentation on the remote sensing images and the aerial images [51, 162], which are captured in nadir view style. The spatial resolutions in such images are approximately the same for all pixels. Oblique views have a much larger land coverage if the platforms are at the same flight height. For example, the unmanned aerial vehicle (UAV) platform has been used to for urban scene observation [133, 92]. The images of different viewing directions are shown in Figure 3.1. The left image of nadir view is from the Vaihingen dataset [162], while the right image of oblique view is from the

* This chapter is based on:

Y. Lyu, G. Vosselman, G.-S. Xia, and M. Y. Yang. Bidirectional multi-scale attention networks for semantic segmentation of oblique uav imagery. In *ISPRS Congress*, 2021

UAVid2020 dataset [133]. Compared with the images in nadir view style, the images in oblique view have very large spatial resolution variation across the entire image.

The state-of-the-art methods for semantic segmentation all rely on powerful deep neural networks, which can effectively extract high-level semantic information to determine the class types for all pixels. Deep neural networks serve as non-linear functions, which map an image input to a label output. Due to its non-linear property, the label output will not scale linearly as the image input scales. When designing the deep neural networks, there is usually a performance trade-off for objects in different scales. For example, the semantic segmentation of a small car in a remote sensing image is better handled in higher resolution where finer details can be observed, such as wheels. For larger objects like roads and buildings, it is better to have more global context to recognize the objects since their whole shapes can be observed for semantic segmentation.

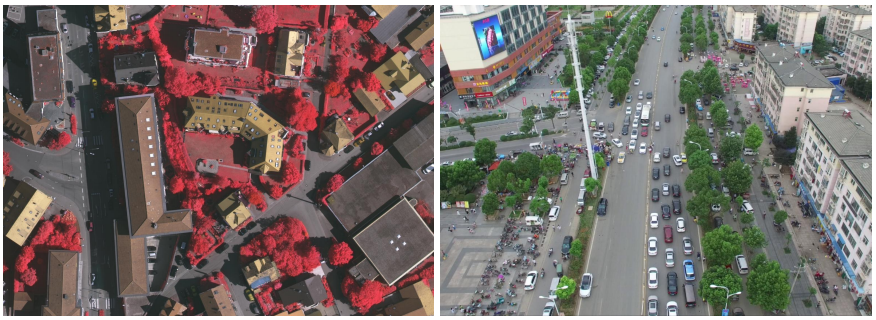

Figure 3.1 Example of images in different viewing style. The left image from Vaihingen dataset[162] is captured in nadir view. The right image from UAVid2020 dataset[133] is captured in oblique view.

When objects in an image dataset have very large scale variation, the semantic segmentation performance of deep neural networks will drop if this multi-scale problem is not considered in the network design. A simple strategy is to apply multi-scale inference [224], i.e., a well-trained deep neural networks predict the score maps of the same image in multiple different scales, and the score maps are averaged to determine the final label prediction. Such strategy generally provides better performance. However, a good prediction from a proper scale could be undermined by those worse predictions from other scales, which limits the model performance. Max-pooling selects one score map prediction of multiple scales for each pixel, but the optimal output could be the interpolation of the prediction of multiple scales. A smarter way of fusing the output score maps is to leverage on an attention model [36], which determines the weights when fusing the score maps of different scales. The strategy has been extended to a hierarchical structure for better performance [175].

With respect to the design of deep neural networks, there are several strategies to relieve the multi-scale problem. The first strategy is to gradually refine features from coarse scales to fine scales [128, 161, 133]. The second strategy is to design a multi-scale feature extractor module in the middle of the deep neural networks [224,

35, 37, 220]. Self-attention [66, 90, 219] and graph networks [118, 117] have also been applied to aggregate information globally to reinforce the features for each pixel.

In this chapter, we propose the bidirectional multi-scale attention networks (BiMSANet) to address the multi-scale problem in the semantic segmentation task. Our method is inspired by the multi-scale attention strategy [36, 175] and the feature level fusion strategy [35, 224], and jointly fuses the features guided by the attention of different scales in bidirectional pathways, i.e, coarse-to-fine and fine-to-coarse. Our method is tested on the new UAVid2020 dataset [133]. One of its challenges is the huge inter-class and intra-class scale variance for different objects due to its oblique viewing style. Our method achieves a new state-of-the-art result with a mIoU score of 70.8%. Compared with the currently top ranked method [175], which features on handling the multi-scale problem, our methods outperforms by almost 0.8%.

The contributions of this chapter are summarized as follows,

- We have proposed a novel bidirectional multi-scale attention networks (BiMSANet) to handle the multi-scale problem for the semantic segmentation task.
- We have visualized in multiple perspectives and analyzed the bidirectional multi-scale attentions in details.
- We have achieved state-of-the-art result on the UAVid2020 benchmark, and the code will be made public.

3.2 Related Work

In this section, we will discuss other works that are related to our work. In order to handle the multi-scale problem for the semantic segmentation, a number of deep neural networks have been designed.

Multi-scale feature fusion. The first basic type of method is to aggregate features of multiple scales from deep neural networks. FCN [128] and U-Net [161] have adopted skip connections between encoder and decoder to gradually fuse the information from multiple scales. MSDNet [133] has extended the connection across scales to further increase the performance. ZipZagNet [55] uses a more complex zip-zag architecture between the backbone and the decoder for intermediate multi-scale feature aggregation. HRNet [188] proposes a multi-scale backbone to exchange information between branches of coarse scale and fine scale. BiSeNet [215] proposes a dual branch structure for better performance, one branch for higher spatial resolution, while the other for richer semantic features.

Multi-scale context extraction. Another method is to aggregate multi-scale context from the same feature maps with a module. PSPNet [224] has adopted pyramid pooling module, which has pooling modules of multiple scales to pool context features for the object recognition. DeepLabv3 [35, 37] has utilized atrous spatial pyramid pooling module, which assembles multi-scale features with convolutions of multiple atrous rates. OCNet [220] proposes pyramid object context (Pyramid-OC) module and atrous spatial pyramid object context (ASP-OC) module to extract object context in multiple scales.

Context by relations. With the creation of self-attention mechanism [180] for natural language processing, better semantic segmentation results have also been achieved when self-attention is applied to reason the relation between pixels. Self-attention refines the features in a non-local style, which aggregates information for each pixel globally. DANet [66] has utilized dual attention module, position attention and channel attention, to extract information globally. CCNet [90] has applied the criss-cross attention module to reduce the computational complexity of the self-attention. OCRNet [219] has used explicit class attention to reinforce the features. However, these types of methods are normally intensive in memory and computation as there are too many pixels, resulting in very dense connections between them. Graph reasoning is another way to include relations among objects. Instead of adopting dense pixel relations, sparse graph structure makes the context relation reasoning less intensive in memory and computation. [118] proposes the symbolic graph reasoning (SGR) layer for context information aggregation through knowledge graph. [117] transforms a 2D image into a graph structure, whose vertices are clusters of pixels. Context information is propagated across all vertices on the graph.

Inference in multi-scale. Multi-scale inference is widely used to provide more robust prediction, which is orthogonal to previously discussed methods as those networks can be regarded as a trunk for multi-scale inference. Average pooling and max pooling on score maps are mostly used, but they limit the performance. [36] propose to apply attentions for fusing score maps across multiple scales. The method is more adaptive to objects in different scales as the weights for fusing score maps across multiple scales can vary. [175] further improve the multi-scale attention method by introducing a hierarchical structure, which allows different network structures during training and testing to improve the model design. Our work also focuses on the multi-scale inference. We have further improved the multi-scale attention mechanism by introducing feature level bidirectional fusion.

3.3 Preliminary

In this section, we first go through some network architecture design to better help understand the newly proposed bidirectional multi-scale attention networks.

3.3.1 Multi-Scale-Dilation Net

The multi-scale-dilation net [133] is proposed as the first attempt to tackle the multi-scale problem for the UAVid dataset. The basic idea shares the philosophy of multi-scale image inputs, where the input images are scaled by the scale to batch operation and batch to scale operation. The intermediate features are concatenated from coarse to fine scales, which are used to output the final semantic segmentation output. The structure is shown in Figure 3.2. The feature extraction part is named as trunk in the following figures.

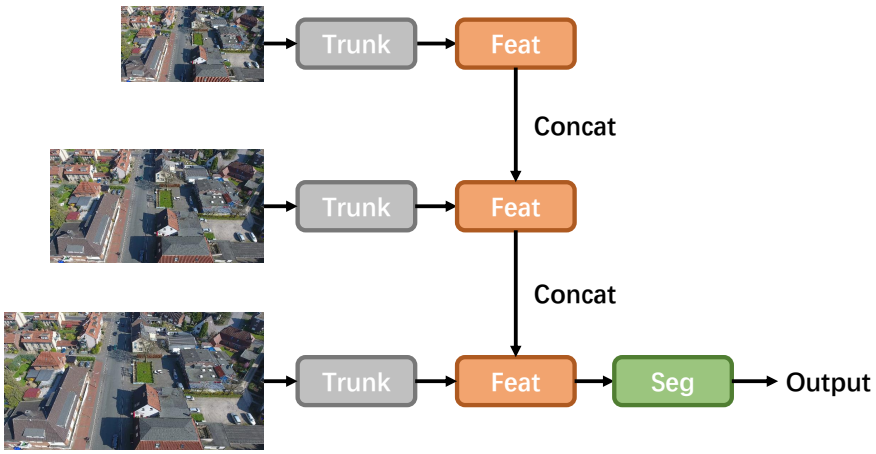

Figure 3.2 Architecture of the multi-scale dilation net. Features are aggregated from coarse to fine scales with concatenation.

3.3.2 Hierarchical Multi-Scale Attention Net

The hierarchical multi-scale attention net [175] is proposed to learn to fuse semantic segmentation outputs of adjacent scales by a hierarchical attention mechanism. The deep neural networks learn to segment the images while predicting the weighting masks for fusing the score maps. This method ranks as the top method in the Cityscapes pixel-level semantic labeling task [47], which focuses on the multi-scale problem. The hierarchical mechanism allows different network structures during training and inference, e.g., the networks have only two branches of two adjacent scales during training, while the networks could have three branches of three adjacent scales during testing as shown in Figure 3.3.

3.3.3 Feature Level Hierarchical Multi-Scale Attention Net

One limitation of the hierarchical multi-scale attention networks is that the fused score maps are the linear interpolation of the score maps in adjacent scales, whereas the best score maps could be acquired with the interpolated features instead. A simple solution that we propose is to move the segmentation head to the end of the fused features as shown in Figure 3.4.

3.4 Bidirectional Multi-scale Attention Networks

In this section, the structure of the proposed bidirectional multi-scale attention networks will be introduced.

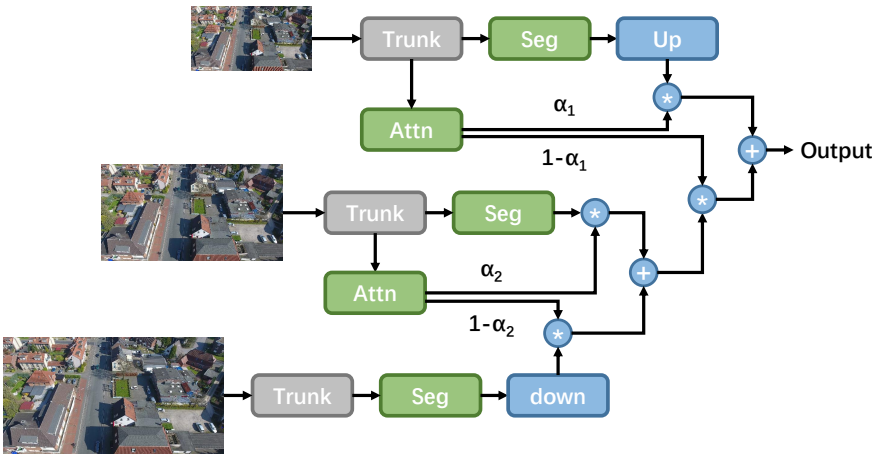

Figure 3.3 Architecture of the hierarchical multi-scale attention networks. In addition to the predicted score maps, extra weighting masks are predicted from the attention sub-networks for fusing the score maps of adjacent scales. $\oplus$, $\circledast$ stand for element-wise addition and multiplication, respectively.

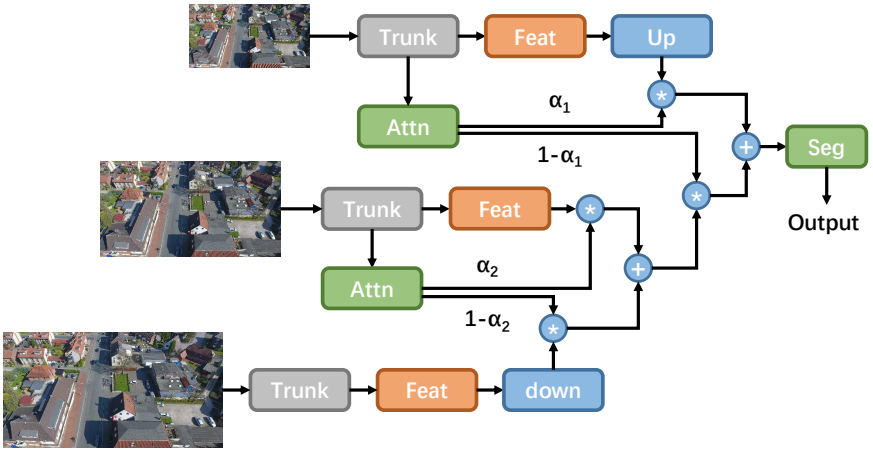

Figure 3.4 Architecture of the hierarchical multi-scale attention networks with feature level fusion. Segmentation head is moved to the end of the fused features. $\oplus$, $\circledast$ stand for element-wise addition and multiplication, respectively.

3.4.1 Overall Architecture

Our design also takes the hierarchical attention mechanism and the feature level fusion into account. The overall architecture is shown in Figure 3.5. For the input image I of size $H \times W$, the image pyramid is built by adding two extra images $I_{2\times}$ and $I_{0.5\times}$, which are acquired by bi-linear up-sampling I to size of $2H \times 2W$ and

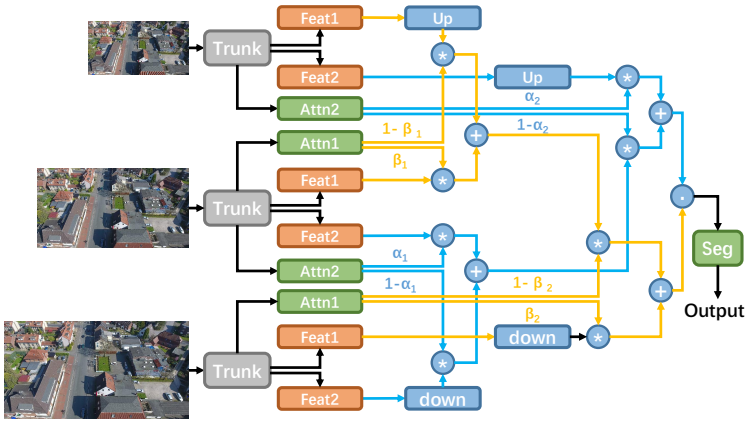

Figure 3.5 Architecture of the bidirectional multi-scale attention networks. The structure is the combination of two feature level hierarchical multi-scale attention nets corresponding to two pathways, where they share the same trunks. The coarse to fine pathway and the fine to coarse pathway are marked with the yellow and the blue arrows, respectively. $\oplus$, $\circledast$ stand for element-wise addition and multiplication, respectively. $\odot$ stands for concatenation in channel dimension.

bi-linear down-sampling I to $\frac{1}{2}H \times \frac{1}{2}W$. The bidirectional multi-scale attention networks have two pathways for feature fusing in a hierarchical manner. For each pathway, the structure is the same as the feature level hierarchical multi-scale attention nets. The design of the two pathways allows the feature fusion from both directions, and the fusion weights can be better determined in a better scale. The reason to use feature level fusion is that we need distinct features for two pathways. If the score maps are used for fusion, the feat1 and the feat2 in the two pathways would be the same, which limits the representation power of the two pathways. The two pathways take advantage of their own attention branches and features. Attn1 branch and Feat1 are for the coarse to fine pathway, while Attn2 branch and Feat2 are for the fine to coarse pathway. The Feat1 and the Feat2 from two pathways are fused hierarchically across scales, and the final feature is the concatenation of the features from the two pathways.

The Feat1 and Feat2 are reduced to the half number of channels as the Feat in feature level hierarchical multi-scale attention net. This setting is to provide fair comparisons between these two types of networks, since it leads to features with the same number of channels before the segmentation head.

The parameter sharing is also applied in the design. Three branches corresponding to the three scales share the same network parameters for Trunk, Attn1 and Attn2. Feat1 and Feat2 in the three branches are different as they are the output of different image inputs through the same trunk.

3.4.2 Module Details

In this section, we will illustrate the details of each component we applies.

Trunk. In order to effectively extract information from each single scale, we have adopted the deeplabv3+ [37] as the trunk. We apply the wide residual networks [222] as the backbone, namely the WRN-38, which has been pre-trained on the imagenet dataset [165]. The ASPP module in the deeplabv3+ has convolutions with atrous rate of $1, 6, 12$, and 18. The features f_b from the deeplabv3+ are further refined with a sequence of modules as follows, $Conv3 \times 3(256) -> BN -> ReLU -> Conv3 \times 3(256) -> BN -> ReLU -> Conv1 \times 1(nc)$ (numbers in the brackets are the numbers of output channels), which corresponds to the feature transformation in the Seg of the hierarchical multi-scale attention net before the final classification.

The trunk T transforms an image input I into feature maps f with nc channels, i.e., $f = T(I)$. $nc = n_{class} \times d$, where n_{class} is the total number of classes for the semantic segmentation task. d is the expansion rate for the channels. d is set to 4 in our case. The first $\frac{1}{2}nc$ channels are for the Feat1, while the second $\frac{1}{2}nc$ channels are for the Feat2.

Attention head. The Attn1 and the Attn2 share the same structure, but with different parameters. The attention heads map the features f_b from the deeplabv3+ to the attention weights α, β (ranging from 0.0 to 1.0 with $\frac{1}{2}nc$ channels) for the two pathways. For each attention head, the structure is comprised of a sequence of modules as follows, $Conv3 \times 3(256) -> BN -> ReLU -> Conv3 \times 3(256) -> BN -> ReLU -> Conv1 \times 1(\frac{1}{2}nc) -> Sigmoid$ (numbers in the brackets are the output channels).

Segmentation head. The segmentation head Seg converts the fused input feature maps f_{fused} into score maps l ($8channels$ for the UAVid2020 dataset), which correspond to the class probabilities for all the pixels, i.e., $l = Seg(f_{fused})$. The segmentation head is simply a 1×1 convolution, $Conv1 \times 1(n_{class})$. Argmax operation along the channel dimension outputs the final class labels for all the pixels.

Auxiliary semantic head. As in [175], we apply auxiliary semantic segmentation heads for each branch during training, which consists of only a 1×1 convolution, $Conv1 \times 1(n_{class})$.

3.4.3 Training and inference

As our model follows the hierarchical inference mechanism, it allows our model to be trained with only 2 scales, while to infer with 3 scales ($0.5\times, 1\times, 2\times$). Such design makes it possible for our network to adopt a large trunk such as deeplabv3+ with WRN-38 backbone for better performance. We use RMI loss [225] for the main semantic segmentation head and cross entropy loss for the auxiliary semantic head.

3.5 Experiments

In this section, we will illustrates the implementation details for the experiments and compare the performance of different models on the UAVid2020 dataset.

3.5.1 Dataset and Metric

Our experiments are conducted on the public UAVid2020 dataset[2] [133]. The UAVid2020 dataset focuses on the complex urban scene semantic segmentation task for 8 classes. The images are captured in oblique views with large spatial resolution variation. There are 420 high quality images of $4K$ resolutions (4096×2160 or 3840×2160) in total, split into training, validation and testing sets with 200, 70 and 150 images, respectively. The performance of different models are evaluated on the test set of the UAVid2020 benchmark. The performance for the semantic segmentation task is assessed based on the standard mean intersection-over-union(mIoU) metric [60].

Table 3.1 Performance comparisons in intersection-over-union (IoU) metric for different models. The top ranked scores are marked in colors. Red for the 1st place, green for the 2nd place, and blue for the 3rd place.

Methods	mIoU(%)	Clutter	Building	Road	Tree	Vegetation	Moving Car	Static Car	Human
MSDNet	56.97	57.04	79.82	73.98	74.44	55.86	62.89	32.07	19.69
DeepLabv3+	67.36	66.68	87.61	80.04	79.49	62.00	71.69	68.58	22.76
HMSANet	70.03	69.32	88.14	82.12	79.42	61.21	77.33	72.52	30.17
FHMSANet	70.33	69.36	87.95	82.69	80.06	62.66	76.88	72.90	30.12
BiMSANet	70.80	69.94	88.63	81.60	80.38	61.64	77.22	75.62	31.34

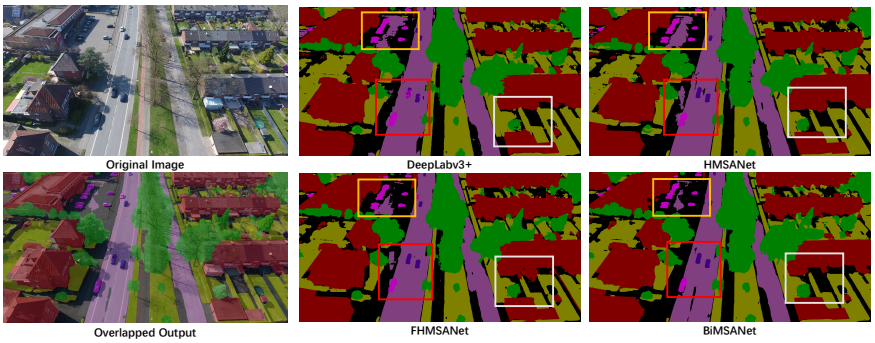

Figure 3.6 Qualitative comparisons of different models on the UAVid2020 test set. The example image is from the test set (seq30, 000400). Bottom left image shows the overlapped result of the BiMSANet output and the original image. Three example regions for comparisons are marked in red, orange, and white boxes.

3.5.2 Implementation

Training. All the models in the experiments are implemented with pytorch [146], and trained on a single Tesla V100 GPU of $16G$ memory with a batch of 2 images. Mixed precision and synchronous batch normalization are applied for the model. Stochastic gradient decent with a momentum 0.9 and weight decay of $5e^{-4}$ is applied as the optimizer for training. "Polynomial" learning rate policy is adopted [125] with a poly exponent of 2.0. The initial learning rate is set to $5e{-}3$. The

[2] https://uavid.nl/

model is trained for 175 epochs by random image selection. We apply random scaling for the images from $0.5\times$ to $2.0\times$. Random cropping is applied to acquire image patches of size of 896×896.

Testing. As the $4K$ image is too large to fit into the GPU, we apply cropping during testing as well. The image is partitioned into overlapped patches for evaluation as in [133] and the average of the score maps are used for the final output in the overlapped regions. The crop size is set to 896×896 with an overlap of 512 pixels in both horizontal and vertical directions.

3.5.3 Model Comparisons

In this section, we will presents the semantic segmentation results on the test set of UAVid2020 dataset for multi-scale-dilation net (MSDNet) [133], deeplabv3+ [37], hierarchical multi-scale attention net (HMSANet) [175], feature level hierarchical multi-scale attention net (FHMSANet), and our proposed bidirectional multi-scale attention networks (BiMSANet). MSDNet is included as reference, which uses an old trunk FCN-8s [128] in each scale. The major comparisons are among DeepLabv3+, HMSANet, FHMSANet, and BiMSANet.

The mIoU scores and the IoU scores for each individual class are shown in Table 3.1. Among all the compared models, the BiMSANet performs the best regarding the mIoU metric. Our BiMSANet has a more balanced prediction ability for both large and small objects.

For the evaluation of each individual class, the BiMSANet ranks the first for classes of clutter, building, tree, static car, and human. The most distinct improvement is for the static car, which is 2.72% higher than the second best score. With only the context information, our method could achieve decent scores for classes of both moving car and static car.

For human class, the scores of HMSANet, FHMSANet and BiMSANet are all significantly higher than the DeepLabv3+, which shows the superiority of multi-scale attention mechanism in handling the small objects. Thanks to the bidirectional multi-scale attention design, BiMSANet achieves the best performance for the human class.

Qualitative comparisons are shown in Figure 3.6. The example image is selected from the test set (seq30, 000400). As the ground truth label is reserved for benchmark evaluation, the overlapped output is shown instead in Figure 3.6. Three example regions are marked in red, orange, and white boxes.

In the red box region, it could be seen that the deeplabv3+ struggles to give coherent predictions for cars in the middle of the road, while the other three models have better results due to the multi-scale attention. The HMSANet and the FHMSANet wrongly classify part of the sidewalks, which is outside the road, as road class. BiMSANet handles better in this area. However, part of the road near the lane-mark are wrongly classified as clutter by the BiMSANet. In the orange box region, the parking lot, which belongs to the clutter class, is predicted as the road by all four models, and the BiMSANet makes the least error. In the white box region, the ground in front of the entrance door is wrongly classified as building by all models except the BiMSANet. This is benefited from the bidirectional multi-scale attention design.

We have also shown the performance for human class segmentation in Figure 3.7. The example image is from the test set (seq22, 000900). The zoomed in images in the middle and the right columns correspond to the patches in the white boxes of the overlapped output. The four patches are from different context, which is very complex in some local regions. Even though the humans in the image are quite small and in many different poses, such as standing, sitting, and riding, our model can still effectively detect and segment most of the humans in the image.

Figure 3.7 Qualitative example of human class segmentation by the BiMSANet. The example image is from the test set (seq22, 000900). The left column shows the original full image and the overlapped output. The middle and the right columns show the image patches cropped from the overlapped output (marked by white boxes), which all focus on the human class. The red circles mark some missing segmentation.

Table 3.2 Ablation study for models. The performance gains could be observed by gradually adding components.

Methods	mIoU(%)	mIoU Gains(%)	Trunk	Multi-Scale Attention	Feature Level Fusion	Bidirection
DeepLabv3+	67.36	-	✓	-	-	-
HMSANet	70.03	+2.67	✓	✓	-	-
FHMSANet	70.33	+0.30	✓	✓	✓	-
BiMSANet	70.80	+0.47	✓	✓	✓	✓

3.5.4 Ablation Study

In this section, we will compare the performance gains by gradually adding the components. The corresponding results are shown in Table 3.2. It is easy to see that the multi-scale processing is useful for the oblique view UAV images. The mIoU score has increased by 2.67% by including the multi-scale attention into the networks. The feature level fusion is also proved to be useful as it helps the networks to improve the mIoU score by 0.3%. By further adding the bidirectional attention mechanism, the networks improve the mIoU score by another 0.47%.

3.5.5 Analysis of Learned Multi-Scale Attentions

In this section, we will analyze the learned multi-scale attentions from the BiMSANet to better understand how the attentions work. We explore from mainly

three perspectives: attentions of different channels, different scales, and different directions. The example image is from the test set (seq25,000400). Attentions from both Attn1 branch and Attn2 branch are used, noted as α and β in Figure 3.5. α is for the fine to coarse pathway, while β is for the coarse to fine pathway.

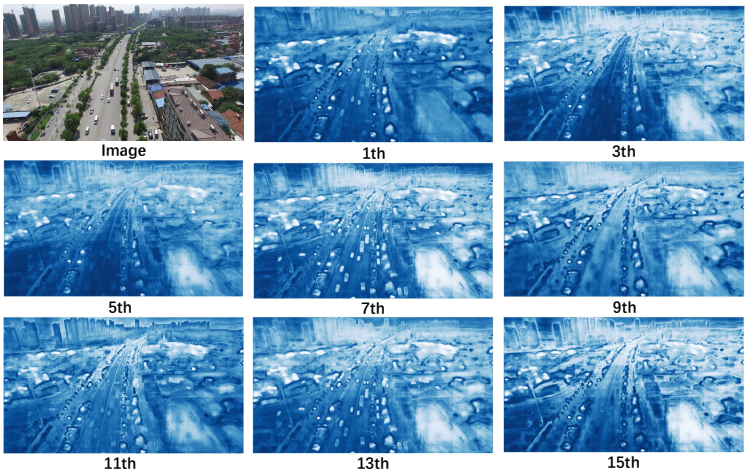

Figure 3.8 Attention analysis of different channels. Example attentions are of $1\times$ scale from Attn1 branch. The image on the top left shows the image adopted. The other 8 images are the attention maps from different channels. Channel indices are presented below the images. Brighter color means higher value. Best visualized with zoom in.

3.5.5.1 Attention of different channels

The multi-scale attentions in our BiMSANet have $\frac{1}{2}nc$ channels (16 in our case), which is different from the HMSANet [175], whose attention has only one single channel for all classes. The attentions guide the fusion of features across scales. Example attentions of different channels in $1\times$ scale branch are shown in Figure 3.8. Different channels have different attentions focusing on different parts of the image. It is obvious that different channels have different focus for different classes, e.g., $1th$ channel more focus on trees, $3th$ channel less focus on roads, and $7th$ channel have the most focus on moving cars.

3.5.5.2 Attention of different scales

In order to analyze the difference of attentions in different scales, we have selected 4 attentions from each of the Attn1 branch and the Attn2 branch as shown in Figure 3.9. The superscripts are the channel index of the attentions. By comparing the α_1 with α_2, which are predicted in $1\times$ and $0.5\times$ scales, we could see that attentions in different scales have different focus. The difference of the same channel between α_1 and α_2 are more worth of comparisons. The same applies for β_1 and β_2.

From α_1^1 and α_2^1, it could be noted that the recognition of cars in closer distance are more based on context, since the values of α_2^1 are larger than α_1^1. The recognition

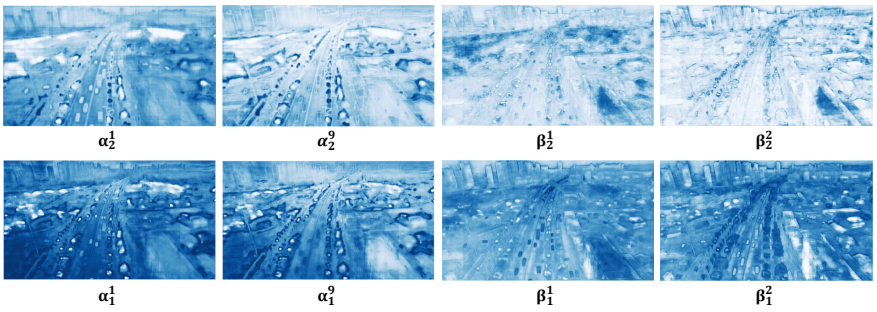

Figure 3.9 Attention analysis of different scales. We select 4 attentions from each of the Attn1 branch and the Attn2 branch. α, β are of the same meaning as in Figure 3.5. The superscripts are the channel index of the attentions. α_2, β_2 correspond to the attentions predicted in the $0.5\times$ scale and the $2\times$ scale. α_1, β_1 are predicted in $1\times$ scale. Brighter color means higher value. Best visualized with zoom in.

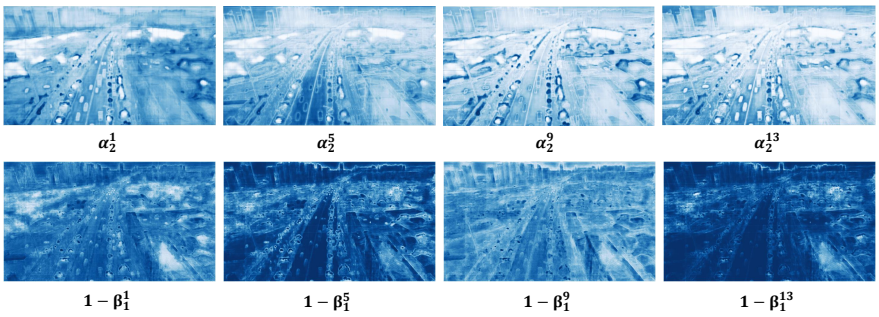

Figure 3.10 Attention analysis of different directions. The figure shows the attentions for fusing features of scale $0.5\times$ and scale $1\times$. α_2 is for the fine to coarse pathway, while $1-\beta_1$ is for the coarse to fine pathway. Brighter color means higher value. Best visualized with zoom in.

of road that are closer to the camera also relies more on the coarser level features, which is reasonable as the road area is large and requires more context for recognition. It is also interesting to note that the middle lane-marks is even brighter than other parts of the road in α_2^1, which means the recognition requires more context. It is reasonable as the color and the texture of the lane-marks are quite different compared to other parts of the road. The distant buildings near the horizon relies more on the coarser level features as well.

We have also noticed that the α_2 ($0.5\times$ scale) and β_2 ($2\times$ scale) have larger values on average compared with α_1 and β_1 ($1\times$ scale), which means that features with context information and fine details are both valuable for object recognition.

3.5.5.3 Attention of different directions

In our bidirectional design, both the coarse to fine pathway and the fine to coarse pathway fuse the features from three scales ($0.5\times, 1\times, 2\times$). In this section, we

analyze if the feature fusion in two pathways has the same attention pattern. Attention examples are shown in Figure 3.10. Attentions α_2 and $1 - \beta_1$ from two pathways are both for the feature fusion across scale $0.5\times$ and $1\times$. Although the attention values of same pixels can not be directly compared as the feature sources are different (Feat1 and Feat2), it is still evident that the attention densities on average are quite different. There are more activation in α_2 than $1 - \beta_1$, showing that the two pathways play different roles for feature fusion across same scales.

3.6 Conclusion

In this chapter, we have proposed the bidirectional multi-scale attention networks (BiMSANet) for the semantic segmentation task. The hierarchical design adopted from [175] allows the usage of larger trunk for better performance. The feature level fusion and the bidirectional design allows the model to more effectively fuse the features from both of the adjacent coarser scale and the finer scale. We have conducted the experiments on the UAVid2020 dataset [133], which have large variation in spatial resolution. The comparisons among different models have shown that our BiMSANet achieves better results by balancing the performance of small objects and large objects. Our BiMSANet achieves the state-of-art result with a mIoU score of 70.80% for the UAVid2020 benchmark.

Learning Instance Propagation for Video Object Segmentation

4

Abstract

In recent years, the task of segmenting foreground objects from background in a video, i.e., video object segmentation (VOS), has received considerable attention. In this chapter, we propose a single end-to-end trainable deep neural network, convolutional gated recurrent Mask-RCNN, for tackling the semi-supervised VOS task. We take advantage of both the instance segmentation network (Mask-RCNN) and the visual memory module (Conv-GRU) to tackle the VOS task. The instance segmentation network predicts masks for instances, while the visual memory module learns to selectively propagate information for multiple instances simultaneously, which handles the appearance change, the variation of scale and pose and the occlusions between objects. After offline and online training under purely instance segmentation losses, our approach is able to achieve satisfactory results without any post-processing or synthetic video data augmentation. Experimental results on DAVIS 2016 dataset and DAVIS 2017 dataset have demonstrated the effectiveness of our method for video object segmentation task.

4.1 Introduction

Video object segmentation (VOS) aims at segmenting foreground objects from background in a video with coherent object identities. Such visual object tracking task serves for many applications including video analysis and editing, robotics and autonomous cars. Compared to the video object tracking task in bounding box level, this task is more challenging as pixel level segmentation is more detailed description of an object.

The VOS task is defined as a semi-supervised problem if ground truth annotations are given for the first several frames. It is otherwise an unsupervised problem if no annotation is provided. The ground truth annotations are masks that mark the objects that need to be tracked through the whole video. In our work, we focus on

* This chapter is based on:

Y. Lyu, G. Vosselman, G.-S. Xia, and M. Ying Yang. Lip: Learning instance propagation for video object segmentation. In *ICCVw*, Oct 2019

Figure 4.1 Example predictions by our method from DAVIS [151, 149] dataset. Top row: Parkour sequence. An example of large appearance change over time. One every 20 frames shown of 100 in total. Middle row: Drift-straight sequence. An example of large scale and pose variation over time. One every 10 frames shown of 50 in total. Bottom row: Dogs-jump sequence. An example of occlusions between objects. One every 5 frames shown of first 20.

semi-supervised video object segmentation task, where the ground truth annotations are provided only for the first frame.

There are several challenges that make VOS a difficult task. First, both the appearance of the target objects and the background surroundings may change significantly over time. Second, there could be a large pose and scale variation over time. Third, there could be occlusions between different objects, which hinder the performance of tracking. Examples of the above three challenges are shown in Fig. 4.1. A notable and challenging dataset for the VOS task is the DAVIS 2016 dataset [149], which is designed for single-object video segmentation. Later the DAVIS 2017[151] is brought out focusing on segmentation of multiple video objects. Both of the datasets are provided with mask annotations of extremely high accuracy.

Most of the current methods for the VOS task, such as VPN [93], MSK [148] and RGMP [198], are based on the pixel level mask propagation. However, those methods fail to give a coherent label within an instance. In this chapter, we introduce a single end-to-end trainable network to predict masks on instance level, namely the convolutional gated recurrent Mask-RCNN. It integrates instance segmentation network (Mask-RCNN [76]) with visual memory module (Conv-GRU [8]). Instance segmentation network is designed for foreground object segmentation, which is extended with visual memory for foreground object segmentation in a video. The incorporated visual memory helps to propagate information across frames to handle the appearance change, the pose and scale variation and the occlusions between objects. Our network gives a coherent label to a detected instance and assigns one label to only one detected instance. The model structure is shown in Fig. 4.2.

Our **Contributions** are:

- We propose a novel convolutional gated recurrent Mask-RCNN to learn instance propagation (**LIP**) for video object segmentation (VOS) task. Our model simultaneously segments all the target objects in the images.
- We design a single end-to-end trainable network for VOS task, enabling both mask propagation in the long term and bottom-up path augmentation.

- A strategy to successfully train the model for VOS task has been brought out. All the training processes are guided by the instance segmentation losses only.

4.2 Related work

In this section, we will discuss some relevant work.

Object detectors. Object detection starts with box level prediction and has a great improvement over the years. Single-stage detectors [156, 157, 124, 65, 121] have faster running speed while two-stage networks [69, 159] are more accurate in general. Later, Mask-RCNN [76] merges object detection with semantic segmentation by combining Faster-RCNN [159] and FCN [128], which form a conceptually simple, flexible yet effective network for instance segmentation task. Mask-RCNN network is suitable for instance segmentation on static images, but lacks the ability for temporal inference. Our work is to further extend Mask-RCNN with Conv-GRU module to solve video object segmentation task.

Recurrent neural networks (RNNs). RNNs [83, 164] are widely used for tasks with sequential data, such as image captioning [97], image generation [72] and speech recognition [71]. The key for RNNs is the hidden state, which selectively accumulates information from current input and the previous hidden state over time. However, RNN has its limitation as it fails to propagate information for a long sequence due to the problem of gradient vanishing or explosion in training [80, 145]. Two RNN variants, LSTM [81] and GRU [44] are more effective for the long term prediction by taking advantage of gating mechanism. To further encode spatial information, they are extended to Conv-LSTM [203] and Conv-GRU [8] respectively and have been used for video prediction [64] and action recognition [8].

Methods for VOS. Conv-GRU has already been used for video object segmentation. It serves as visual memory in [177] and has been proved to boost the performance for VOS task. However, their model performs binary semantic segmentation only, which is not suitable for video object segmentation task with multiple objects.

VPN [93], MSK [148] and RGMP [198] learn to propagate mask for the VOS task. VPN utilizes learnable bilateral filters to achieve video-adaptive information propagation across frames. MSK learns to utilize both current frame and mask estimation from the previous frame for mask prediction. RGMP utilizes the first frame and mask as reference for instant information propagation besides the usage of current frame and previous mask estimation. Both MSK and RGMP achieve good results, but they can only propagate information for instances one by one.

Specially, OSVOS [24], OSVOS+ [136] and OnAVOS [183] tackle video object segmentation from static images, achieving temporal consistency as a by-product. They learn a general object segmentation model from image segmentation datasets and transfer the knowledge for video object segmentation. They all rely on additional post processing for better segmentation result. OnAVOS further applies online adaptation to continuously fine-tune the model, which is very time consuming.

[98] explores the benefits from in-domain training data synthesis with the labelled frames of the test sequences. [198] synthesizes video training data from static image dataset to add to limited video training samples. [88, 170] explore fast prediction without online training through matching based method. CINM [9] achieves good prediction by spatial-temporal post-processing based on results from

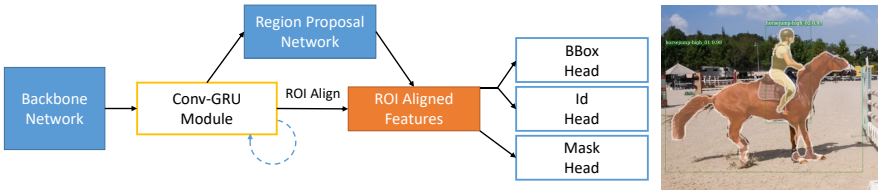

Figure 4.2 Overall model structure. The backbone network distills useful features from each input image. The features are then sent to Conv-GRU module (visual memory) for feature propagation. The output features from Conv-GRU module are utilized by region proposal network for proposal generation. Multiple heads finally take the ROI aligned features for video object segmentation. An example output is shown on the right, including bounding boxes, id predictions and object segments. The class of an instance is named by video sequence name plus object index.

OSVOS [24]. To handle the problem of long term occlusion, [116, 114] apply re-identification network to retrieve the missing objects, which complements their mask propagation methods. Recently, there are still many researches focusing on single-object video segmentation [201, 86, 45], which are not easily transformed for video segmentation of multiple objects. MaskRNN [87] is another method for instance level segmentation, but it only predicts for one instance at a time. The best results are achieved by ensemble of multiple specialized networks. PReMVOS [131] takes the 1st place of recent DAVIS2018 semi-supervised VOS task by utilizing complex pipeline with multiple specialized networks trained on multiple datasets.

4.3 Method

In this section, we first introduce the structure of our convolutional gated recurrent Mask-RCNN, which extracts and propagates information for multiple objects in a video. It is comprised of mainly three parts. They are the feature extraction backbone, the visual memory module and the prediction heads. The backbone network extracts features that are forwarded to visual memory module. The visual memory module then selectively remembers the new input features and forgets the old hidden states. On top of Conv-GRU, region proposal network (RPN), bounding box regression head, id classification head and mask segmentation head are constructed to solve the VOS task. The whole network is end-to-end trainable under the guidance of instance segmentation losses.

4.3.1 Mask-RCNN

Mask-RCNN [76] is one of the most popular framework designed for instance segmentation task. It is used for instance-wise object detection, classification and mask segmentation, which makes it naturally suitable for multiple video objects segmentation. Roles of different components in Mask-RCNN directly shift to fit VOS task as illustrated below.

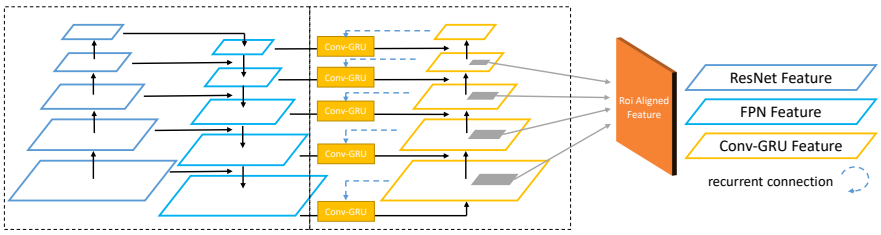

Figure 4.3 Model structure details. The left black dashed box shows the ResNet101-FPN backbone structure. The right black dashed box shows the Conv-GRU module. Our network brings bottom-up path augmentation for output features in Conv-GRU module. The augmented output features are used for both RPN and the prediction heads. All 5 layers are utilized for multi-level RPN, but only 4 bottom layers are used for multi-level ROIs.

Backbone: The backbone network still serves to extract features from images, but more focused on generating useful features adaptively for gates of Conv-GRU module. ResNet101-FPN [77, 120] with group normalization [197] is used as our backbone network. Detailed structure is shown in Fig.4.3.

RPN: Mask-RCNN is known as a two stage instance segmentation network. Bounding boxes of general objects are proposed in the first stage, while classes and masks are predicted instance-wisely in the second stage. Such two stage framework adopts the same philosophy as the training stages of OSVOS [24]. For OSVOS, the network first learns to segment binary mask for general objects in a class-agnostic manner. Then it learns to segment specific objects during online training. In Mask-RCNN, RPN learns to reject background objects and to propose foreground objects in the first stage, which is also class-agnostic. It is in the second stage that classes and masks of different objects are determined.

Bounding box regression head: This branch is used to refine the bounding box proposals. Each predicted box contains one object. The boxes serve to separate different objects in an image.

Classification head: This branch is used to assign the object a correct class label. However, class type is unknown for VOS task. Instead, different objects are associated with different ids, which need to be predicted coherently in a video sequence. Classification branch is naturally transformed into an id classification branch.

Mask segmentation head: This branch is used to extract a mask for each foreground object in the image, which is the main target of VOS task.

Clearly, for the components in Mask-RCNN, there is a direct responsibility mapping from instance segmentation task to VOS task.

4.3.2 Convolutional gated recurrent unit

One difficulty for video object segmentation is the problem of long term dependency. The ground truth is provided only for the first frame, but the objects still need to be predicted after tens or hundreds of frames based on the ground truth from the first frame. The appearance of different objects in the videos may vary greatly and the

objects sometimes get partially or even completely occluded, which makes coherent prediction more difficult.

In order to handle the above problem, we utilize the convolutional gated recurrent unit, serving as a visual memory to handle appearance morphing and occlusion. The memory module learns to selectively propagate the memorized features and to merge them with the newly observed ones. The key role for Conv-GRU module is to maintain a good feature over time for prediction of region proposal, bounding box regression, id classification and mask segmentation.

Compared to the instance segmentation task, where each training batch is comprised of multiple randomly sampled images, the batch in temporal training has less variation as consecutive images from one sequence are highly correlated. This is similar to the problem of small batch size. To relieve such effect, we further replace the bias term in Conv-GRU with the group normalization (GN) layer, which are proved to give consistent performance across different batch sizes [197]:

$$z_t = \sigma(GN(W_{hz} * h_{t-1} + W_{xz} * x_t)) \tag{4.1a}$$

$$r_t = \sigma(GN(W_{hr} * h_{t-1} + W_{xr} * x_t)) \tag{4.1b}$$

$$\hat{h_t} = \Phi(GN(W_h * (r_t \odot h_{t-1}) + W_x * x_t)) \tag{4.1c}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h_t}, \tag{4.1d}$$

where x_t is the input feature of time t, h_t is the hidden state of time t. z_t, r_t are update gate and reset gate respectively. W are convolutional filter parameters. σ and Φ are sigmoid function and tanh function respectively.$*$ and $\odot$ denote the convolution operation and element-wise multiplication respectively.

For each level of the feature pyramid network [120] (FPN), we create a corresponding Conv-GRU layer. The layers at different levels learn different transition functions for the hidden states. As bottom up path augmentation has been proved to be useful for instance segmentation [123], we easily achieve it by down-sampling and addition operation with output features from multi-level Conv-GRU module. The structure is shown in Fig. 4.3. The output features after path augmentation are used for RPN and prediction heads. Conv-GRU module is deliberately directly inserted after backbone network. In this way, information for both region proposal and instance prediction can be propagated through time.

4.3.3 Online inference

As our model predicts mask for each unique instance, there naturally exist constraints for prediction.

One maximum constraint. For each instance, there should be at most one object detected. This constraint is achieved by selecting highest id prediction score.

Location continuity constraint. If an instance is detected in the previous frame with high enough id prediction score, the location of the current detection should not be far from its previous location. To achieve this constraint, we suppress the prediction for the instance, whose boxes iou between consecutive frames is low.

As probability for id prediction decays over time, we further apply a very light weighted fine-tuning process for the last linear layer of the id head during online prediction. If there exists a target object detected with a high enough id prediction

score, its predicted bounding box is set as ground truth for fine-tuning the id head only. By saving and reusing intermediate tensors, the speed for fine-tuning is fast.

4.4 Training the network

In this section, we will describe our training strategy in detail. The training modality for video object segmentation can be divided into offline training and online training [98, 148]. During offline training, the model is trained with the training set only. During online training, the model is fine-tuned with the first frame from the test set. As the class types of the test set are not known and objects may never be seen during offline training, online fine-tuning is necessary to help the model to generalize better for test set.

Our network needs both offline training and online training. During offline training stage, our network learns the features to differentiate all the object instances and learns to predict class-agnostic boxes and masks. During online training stage, our network is fine-tuned to differentiate objects for each test sequence and trained with boxes and masks in a class-specific manner.

4.4.1 Class-agnostic offline training

To provide our model with as much generality as possible, we apply class-agnostic training for bounding box and mask through the whole offline training process. Offline training for our model can be divided into two steps. First, our model is trained with instance segmentation dataset. This step is to provide our model with general object detection ability. Then, we train the model with video dataset to learn to propagate information over time for video object segmentation.

4.4.1.1 Pre-train on instance segmentation dataset

Pre-training on additional dataset is a common practice [183, 24, 136, 114]. We initialize our model by pre-training on Microsoft COCO dataset [122]. Ms-COCO dataset has been widely used for object detection task. It targets common objects in context with annotations including boxes, classes and masks. By first training on Ms-COCO dataset, our model learns to extract useful features for general object detection. As the training is on static images, we set hidden states to be zeros without update for Conv-GRU module.

After this step, our model gains general region proposal ability, general bounding box prediction ability and general object segmentation ability. Our model also learns to differentiate general objects by classes defined on Ms-COCO dataset.

4.4.1.2 Fine-tune on VOS dataset

In this stage, we train all the modules except the backbone network. By fine-tuning our model on video object segmentation dataset, the Conv-GRU module learns to tune its gates to best propagate information. It should be noted that the class number has changed as the video object segmentation dataset does not share the class definition with instance segmentation dataset. Instead, we replace the last

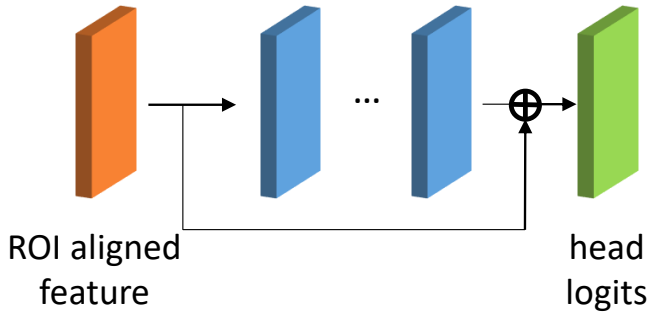

Figure 4.4 Shortcut in prediction head. In order to let output from Conv-GRU module have more direct influence towards final prediction, we add a shortcut connection between ROI aligned feature and head logits by simple addition operation.

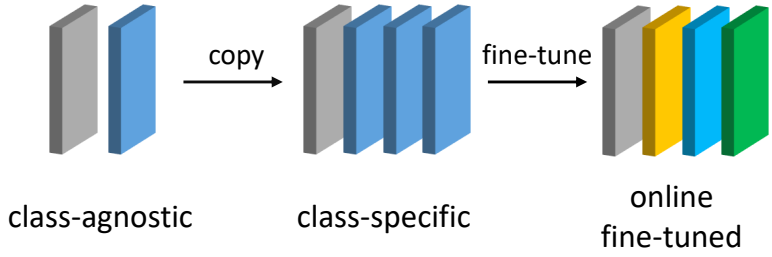

Figure 4.5 Transforming class-agnostic weights to class-specific weights. During online fine-tuning, the class-agnostic bounding box and mask predictions are altered to class-specific. The rectangles are weights in the last linear layer of bbox head or the last convolutional layer of mask head. The grey color marks weights for background and the blue for foreground. Foreground weights are copied for each foreground instance to be fine-tuned uniquely.

linear layer right before softmax layer in the class prediction head with a new one, which now predicts the ids in the dataset. The class prediction head turns into an id prediction head.

The network is trained purely with instance segmentation losses. The different losses guide our model to have different abilities. The mask loss helps our model to propagate mask segmentation. The losses from id head and bbox head help our model to propagate information differently for each instance. Although the mask head and bbox head are trained in a class-agnostic manner, the id head and bbox head provide a chance to learn to propagate class-specific information.

To facilitate the information propagation, we further add a shortcut connection between the ROI aligned feature and the head logits as shown in Fig. 4.4.

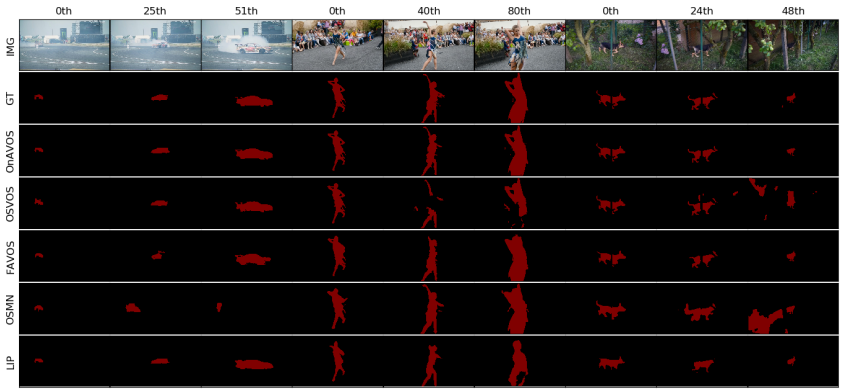

Figure 4.6 Qualitative results comparison of OnAVOS [183], OSVOS [24], FAVOS [41], OSMN [209] and LIP on DAVIS 2016 dataset [149]. The index of each image in a sequence is shown on the top.

4.4.2 Class-specific online fine-tuning

As the instances in test sequences are not the same as in training sequences, the last linear layer in id head needs to be re-initialized and trained to differentiate instances in the current sequence. We replace the last linear layer in the same way as in section 4.4.1.2. We also adopt focal loss [121] for id head to balance the training for multiple instances.

During online fine-tuning, the parameters in backbone network and Conv-GRU module are frozen to keep the learned propagation property. All other parts are fine-tuned for the new objects in test image. The class-agnostic prediction in mask head and bbox head are altered to be class-specific in order to have less competition for different instances. The process is illustrated in Fig. 4.5.

4.5 Experiments

To test how our model learns to propagate instance information in a long term sequence, we evaluate our model on both DAVIS 2016 [149] and DAVIS 2017 [151] datasets, which contain video sequences of high quality and accurate mask annotations of objects. DAVIS 2016 dataset focuses on single-object video segmentation. It has 30 training and 20 validation videos. As an extension to DAVIS 2016 dataset, DAVIS 2017 dataset brings 30 more video sequences for training set and 10 more for validation set. It also provides another 30 sequences for testing. As DAVIS 2017 dataset focuses on multiple object segmentation, it has been re-annotated for each individual target object.

4.5.1 Implementation Details

Our model is implemented with PyTorch [146] library. A Nvidia Titan X (Pascal) GPU with 12GB memory is used for experiments. Details of convolutional gated

recurrent Mask-RCNN are shown below.

Model structure. Our backbone network is a ResNet101-FPN [77, 120] with group normalization [197]. ResNet101 is initialized with weights pre-trained on Imagenet [165]. In Conv-GRU module, the channel number of each hidden state is 256. Kernels of all convolutions in Conv-GRU are of size 3×3 with 256 filters. We apply multi-level RPN and multi-level ROIs for the network. 5 RPN heads are created for 5 levels of the Conv-GRU module. Each RPN head proposes boxes for 1 scale of all 5 scales. ROI aligned features are extracted from 1 of the 4 bottom levels of the Conv-GRU module based on box scales ($< 112^2$, $112^2 - 224^2$, $224^2 - 448^2$, $> 448^2$). The ROI aligned feature resolution is 28×28 for mask head, and 7×7 for bbox head and id head. In all cases, we adopt image centric training [69].

Method	OnAVOS	FAVOS	OSVOS	LIP(Ours)	MSK	PML	SFL	OSMN	CTN	VPN
J&F Mean↑	85.5	81.0	80.2	78.5	77.6	77.4	76.1	73.5	71.4	67.9
J Mean↑	86.1	82.4	79.8	78.0	79.7	75.5	76.1	74.0	73.5	70.2
J Recall↑	96.1	96.5	93.6	88.6	93.1	89.6	90.6	87.6	87.4	82.3
J Decay↓	5.2	4.5	14.9	0.05	8.9	8.5	12.1	9.0	15.6	12.4
F Mean↑	84.9	79.5	80.6	79.0	75.4	79.3	76.0	72.9	69.3	65.5
F Recall↑	89.7	89.4	92.6	86.8	87.1	93.4	85.5	84.0	79.6	69.0
F Decay↓	5.8	5.5	15.0	0.06	9.0	7.8	10.4	10.6	12.9	14.4

Table 4.1 Results on DAVIS 2016 [149]. Left column shows different metrics. Up-arrow↑ means the higher the better. Down-arrow↓ means the lower the better. Methods are in descent order according to J&F mean from left to right.

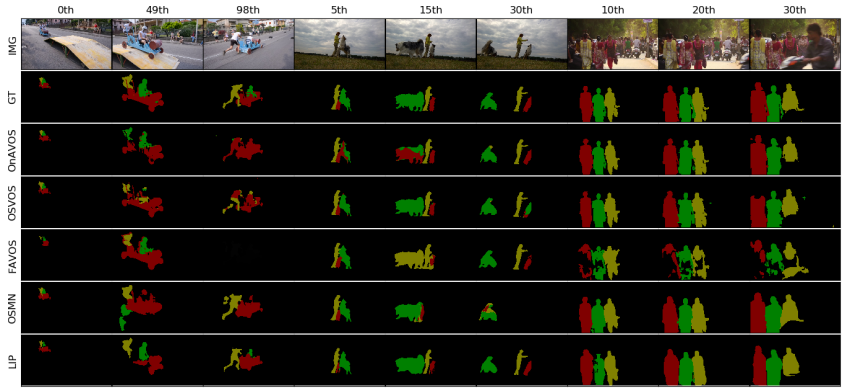

Figure 4.7 Qualitative results comparison of OnAVOS [183], OSVOS [24], FAVOS [41], OSMN [209] and LIP on DAVIS 2017 dataset [151]. The index of each image in a sequence is shown on the top.

Pre-train on Ms-COCO dataset. For each image, we randomly scale it to have its shorter side equal to 1 of 11 different lengths: 640, 608, 576, 544, 512, 480, 448, 416, 384, 352, 320 and its longer size to be maximumly 1333. We sample 512 ROIs with foreground-to-background ratio 1:3. RPN adopts 5 aspect ratios (0.2, 0.5, 1, 2, 5) and 5 scales (32^2, 64^2, 128^2, 256^2, 512^2). The model is trained with stochastic

gradient descent (SGD) for 270K iterations. We fix input hidden states to be zeros for Conv-GRU module, weight decay 0.0001, momentum 0.9. The initial learning rate is 0.02 and dropped by a factor of 10 at 210K and 250K. In the following cases, the configuration is kept the same unless otherwise stated.

Fine-tune on DAVIS dataset. We generate ground truth (GT) bounding boxes from GT masks of DAVIS dataset. The width and height of the boxes are expanded by 10% to prevent incomplete mask prediction caused by inaccurate box prediction. The sequences are randomly shuffled and scaled as in pre-training stage. As there is no causal reasoning in the task, we reverse each sequence for more training data. The backbone network is not trained to prevent over-fitting for DAVIS dataset. 128 ROIs are sampled from each image. The model is trained for 12K iterations with an initial learning rate of 0.002 and dropped by a factor of 10 at 8K and 10K. Due to the GPU memory limitation, it only allows to train with maximum recurrence of 4. We extend the length to 8 by stopping gradient back propagation between 4th and 5th frames.

Online fine-tuning. The network is fine-tuned with the GT of the first image for maximally 1000 iterations with early stopping. If the loss for a prediction head is smaller than an empirically chosen threshold, the loss is ignored. If all the losses are ignored, we stop the training. The thresholds for bbox regression loss, id cross-entropy loss and mask binary cross-entropy loss are 0.015, 0.010 and 0.15 respectively. We also stop the loss back-propagation in id head at its last fully connected layer, so the features to distinguish ids will not be affected by the newly initialized head. Focal loss [121] is used to balance id training, its hyper-parameters are set the same as in [121].

Online inference. For each id, we select 10 detected objects that have id score above 0.2 and apply one maximum constraint to select the best candidate. For the location continuity constraint, we suppress the object instance that has IOU lower than 0.3 with the detection from previous frame if the previous id score is higher than 0.4. To relieve the id score from decaying over time, we apply fine-tuning for id head for maximally 500 iterations with early stopping, and the early stopping threshold for id cross-entropy loss is set to 0.015.

Method	OnAVOS	LIP(Ours)	OSVOS	FAVOS	OSMN
J&F Mean↑	65.4	61.1	60.3	58.2	54.8
J Mean↑	61.6	59.0	56.6	54.6	52.5
J Recall↑	67.4	69.0	63.8	61.1	60.9
J Decay↓	27.9	16.0	26.1	14.1	21.5
F Mean↑	69.1	63.2	63.9	61.8	57.1
F Recall↑	75.4	72.6	73.8	72.3	66.1
F Decay↓	26.6	20.1	27.0	18.0	24.3

Table 4.2 Results on DAVIS 2017 [151]. Left column shows different metrics. Up-arrow↑ means the higher the better. Down-arrow↓ means the lower the better. Methods are in descent order according to J&F mean from left to right.

4.5.2 Comparison with other methods

We compare our method with some state-of-the-art methods on both the DAVIS 2016 benchmark and the DAVIS 2017 benchmark [2] by using standard evaluation metrics J and F [149, 151]. The evaluation on DAVIS 2016 benchmark shows the performance for single-object video segmentation, while the evaluation on DAVIS 2017 benchmark shows the performance for video segmentation of multiple objects. It should be noted that our method does not apply any post-processing, but has been pre-trained on Ms-COCO dataset [122]. Among the several top methods, we remove CINM [9] and RGMP [198] to avoid unfair comparison. CINM [9] is built upon OSVOS [24] and further adopts a refinement CNN and MRF for post-processing. The better initial prediction, the better its result. RGMP [198] cannot be successfully trained with static image dataset and DAVIS dataset alone for mask propagation. It has created a large number of synthetic video training data from Pascal VOC [61, 61], ECSSD [169] and MSRA10K [43] datasets. It is not fair to compare with RGMP as the quality of video training data are not the same and cannot be controlled. For DAVIS 2017 benchmark, we exclude PReMVOS [131] and OSVOS+ [136] as they both use multiple specialized networks in multiple processes to refine their results.

For DAVIS 2016, we compare with OnAVOS [183], FAVOS [41], OSVOS [24], MSK [148], PML [39], SFL [42], OSMN [209], CTN [94] and VPN [93]. We detect multiple objects and evaluate in the way for single-object. Our method ranks the 4th among the compared methods as shown in Table 4.1. It should be noted that our results are better than FAVOS and OSVOS if they are without post-processing. FAVOS achieves J mean and F mean of 77.9% and 76% respectively without tracker and CRF [41]. OSVOS achieves J mean and F mean of 77.4% and 78.1% respectively without boundary snapping post-processing [24]. OnAVOS achieves J mean of 82.8% without CRF post-processing [183]. In addition, we compare our method with another visual memory (Conv-GRU) based VOS method [177]. Both of the methods are trained with additional image dataset, but we achieve 4.5% gain in J&F mean without optical flow and CRF post-processing.

For DAVIS 2017, we compare LIP with OnAVOS [183], OSVOS [24], FAVOS [41] and OSMN [209] as shown in Table 4.2. LIP has relatively better performance as it is better at separating different instances and keeping coherent label within an instance.

Qualitative results on DAVIS 2016 and DAVIS 2017 are shown in Fig. 4.6 and Fig. 4.7, respectively. Fig. 4.6 shows that our LIP can track single object well on instance level and preserve good mask extent for an instance. OSMN [209] and OSVOS [24] fail to keep the mask within an instance.

In Fig. 4.7, it is obvious that the information of an instance in our LIP helps segment multiple objects. All the other methods either assign one label to multiple objects or assign multiple labels to one object, while LIP handles those issues better.

4.5.3 More qualitative results comparison

Here we show more qualitative results comparison of OnAVOS [183], OSVOS [24], FAVOS [41], OSMN [209] and LIP on DAVIS 2016 and DAVIS 2017 datasets in Fig. 4.8, 4.9 and Fig. 4.10, 4.11.

[2] https://davischallenge.org/

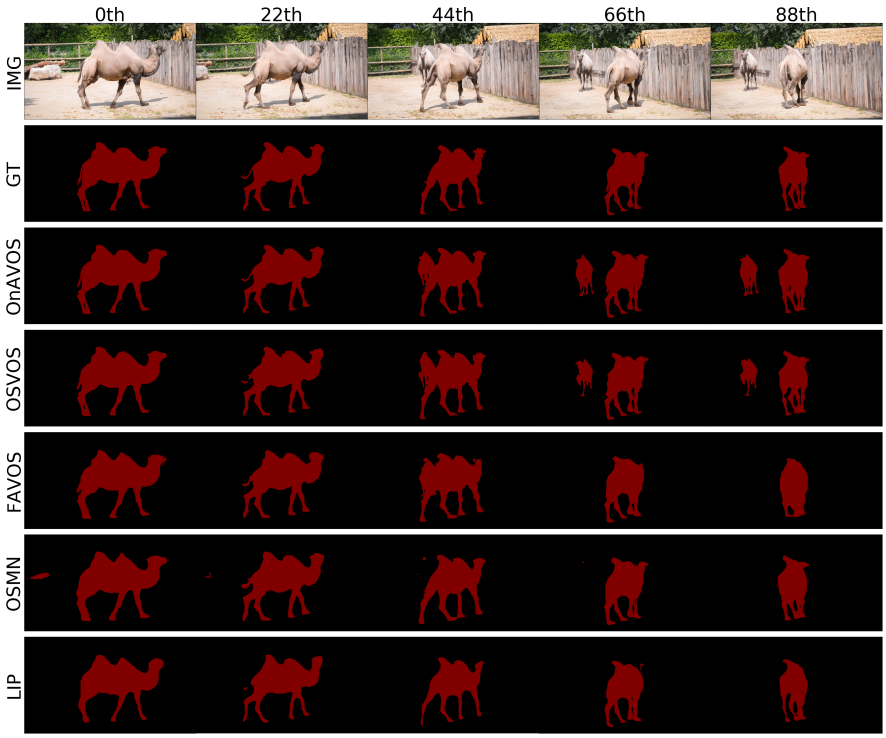

Figure 4.8 Qualitative results comparison of OnAVOS [183], OSVOS [24], FAVOS [41], OSMN [209] and LIP on DAVIS 2016 dataset. The index of each image in a sequence is shown on the top.

4.5.4 Compare with visual memory based method

In addition, we compare our method with another visual memory (Conv-GRU) based video object segmentation method [177] on DAVIS 2016 dataset. Both of the methods are trained with additional image dataset, but we achieve better result without optical flow and CRF post-processing, as shown in Table 4.3. The result shows the value of the learned concept of instances.

Method	VisualMem [177]	LIP(Ours)
Additional dataset	PASCAL VOC	Ms COCO
Additional aid	CRF&Optical Flow	-
J&F Mean↑	74.0	78.5
J Mean↑	75.9	78.0
F Mean↑	72.1	79.0

Table 4.3 Comparison with another visual memory based method [177]. Results reported on DAVIS 2016.

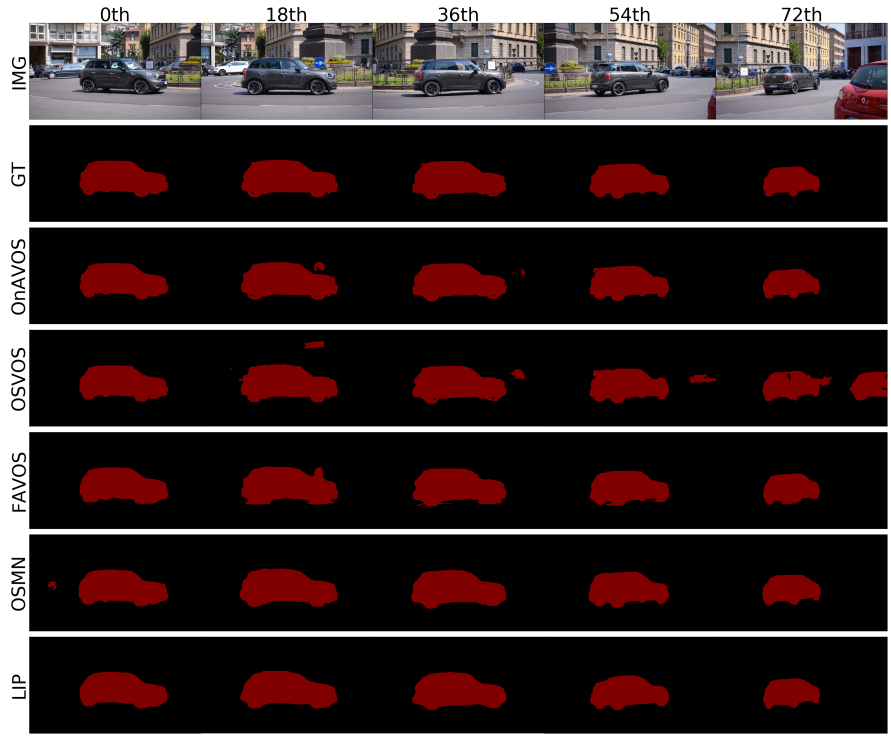

Figure 4.9 Qualitative results comparison of OnAVOS [183], OSVOS [24], FAVOS [41], OSMN [209] and LIP on DAVIS 2016 dataset. The index of each image in a sequence is shown on the top.

4.5.5 Ablation study

We perform ablation study on DAVIS 2017 dataset by comparing the model with and without dynamic visual memory as shown in Table 4.4. We first evaluate the static model by fixing input hidden states (h_{t-1}) to zeros for Conv-GRU module. This is Mask-RCNN with static Conv-GRU module and bottom up path augmentation. Fine-tuning on video dataset is done by training with static images only. The J&F mean score is 59.2%, which is 1 percent lower than the performance of OSVOS [24] with post-processing. The full version of our model is trained with dynamic video images. It reaches the best J&F mean score of 61.1%. The dynamic visual memory contributes as it learns to propagate masks. The static model lacks such property to handle large appearance change, as shown in Fig. 4.12.

Mask-RCNN	Conv-GRU	J Mean	F Mean	J&F Mean
✓	input zero hidden states	56.9	61.5	59.2
✓	✓	59.0	63.2	61.1

Table 4.4 Ablation study results on DAVIS 2017 dataset.

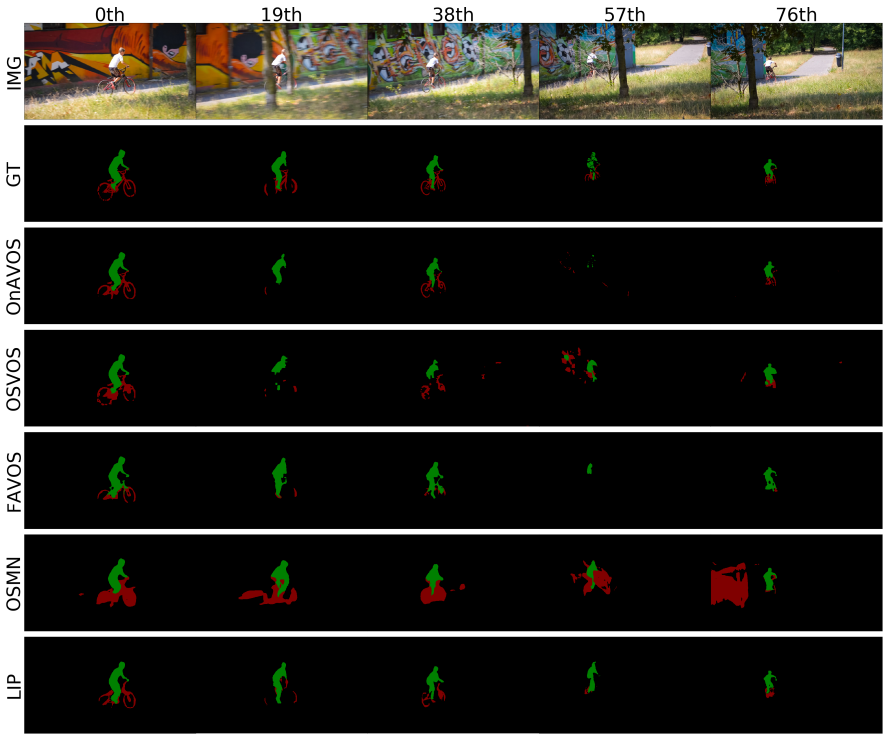

Figure 4.10 Qualitative results comparison of OnAVOS [183], OSVOS [24], FAVOS [41], OSMN [209] and LIP on DAVIS 2017 dataset. The index of each image in a sequence is shown on the top.

We further examine the prediction without *one maximum constraint* and online fine-tuning during inference to see the behavior of the learned model. Example is shown in Fig. 4.13. Most of the detected boxes are around the target objects, except a few boxes covering similar objects nearby. It shows that our model is able to track multiple target objects. It could also be seen that there is decay for probability over time even though the maximum probability is correctly matched to target objects. Such limitation is caused by the fact that the id classifier is learned from the first image, which does not generalize for all images and requires online fine-tuning to preserve probability.

4.5.6 Failed cases and limitation

During online inference, we find two failed cases that are most relevant to our method. Although long-term visual memory is included in our model, it still fails to handle some occlusions as shown in Fig. 4.14. Our model also finds it difficult to infer for multiple instances with large bounding box overlaps as it is difficult to predict the correct ids for all bounding boxes. Example is shown in Fig. 4.15.

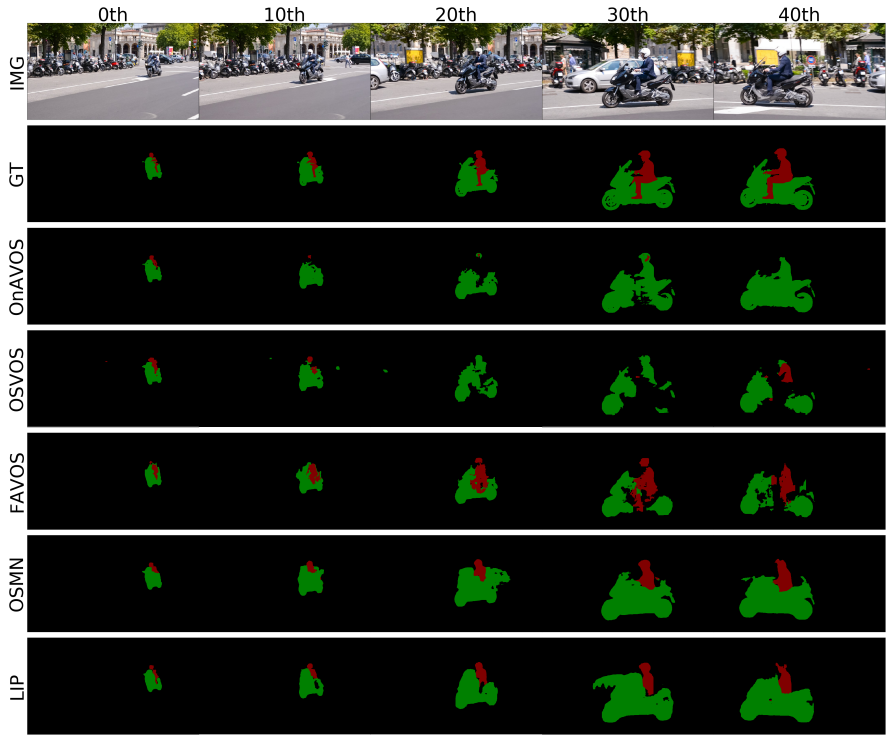

Figure 4.11 Qualitative results comparison of OnAVOS [183], OSVOS [24], FAVOS [41], OSMN [209] and LIP on DAVIS 2017 dataset. The index of each image in a sequence is shown on the top.

Figure 4.12 A qualitative example of prediction with (top row) and without (bottom row) dynamic visual memory.

Figure 4.13 A qualitative example of prediction without one maximum constraint and online fine-tuning for id head.

Figure 4.14 Example of failed cases due to occlusions.

Figure 4.15 Example of failed cases due to large overlaps between target objects.

4.6 Conclusions

We have presented a single end-to-end trainable neural network for video segmentation of multiple objects. We extend the powerful instance segmentation network with visual memory for inference ability across time. Such design serves as an instance segmentation based baseline for VOS task. The newly designed convolutional gated recurrent Mask-RCNN learns to extract and propagate information for multiple instances simultaneously and achieves the state of the art results.

Video object detection with a convolutional regression tracker

5

Abstract

Video object detection is a fundamental research task for scene understanding. Compared with object detection in images, object detection in videos has been less researched due to shortage of labelled video datasets. As frames in a video clip are highly correlated, a larger quantity of video labels are needed to have good data variation, which are not always available as the labels are much more expensive to attain. Regarding the above-mentioned problem, it is easy to train an image object detector, but not always possible to train a video object detector if there are insufficient video labels for certain classes. In order to deal with this problem and improve the performance of an image object detector for the classes without video labels, we propose to augment a well-trained image object detector with an efficient and effective class-agnostic convolutional regression tracker for the video object detection task. The tracker learns to track objects by reusing the features from the image object detector, which is a light-weighted increment to the detector, with only a slight speed drop for the video object detection task. The performance of our model is evaluated on the large-scale ImageNet VID dataset. Our strategy improves the mean average precision (mAP) score for the image object detector by around 5% and around 3% for the image object detector plus Seq-NMS post-processing.

5.1 Introduction

The last several years have witnessed the rapid development of scene understanding in the field of computer vision, especially the fundamental object detection task. The object detection task is to simultaneously localize the bounding boxes of objects and identify their categories in an image. Video object detection extends this task to video sequences, which requires detectors to utilize multiple frames in a video to detect objects over time, which is another emerging topic in computer vision. Compared with the ImageNet object detection (DET) challenge [165], designed for the static image object detection task, the ImageNet video object detection (VID)

* This chapter is based on:

Y. Lyu, M. Y. Yang, G. Vosselman, and G.-S. Xia. Video object detection with a convolutional regression tracker. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2021

challenge [165], designed for the video object detection task, brings additional challenges into focus. The appearance of objects might deteriorate significantly in some frames of a video, which could be caused by motion blur, video defocus, part occlusion, or rare poses. Examples of images in good quality and bad quality are shown in Fig. 5.1. However, rich context information in temporal domain provides clues and opportunities to improve the performance of the object detection in videos. Both of the tasks have received much attention since the introduction of the ImageNet DET challenge and the ImageNet VID challenge in ILSVRC 2015 [165].

Figure 5.1 Examples of images in good and bad quality are marked by the green box and the red box, respectively. Images with good quality are better for training the object classifier in an object detector. Images with bad quality are handled by a tracker for better object localization and object re-identification. Examples of video object detection results by our method, which unifies object detection and object tracking, are shown on the right. The class scores are consistent over long time even if image quality decays, or objects are in rare poses or partially occluded.

There are more datasets for the image object detection task than for the video object detection task. It is much more expensive to collect labels for video datasets as there are more frames to label as the frames in a video clip are highly correlated. In this chapter, we explore the possibility of augmenting a well-trained image object detector for the video object detection task. Suppose that video labels for certain classes are not available, is it still possible to boost the performance for an image object detector? Our solution is to design a class-agnostic plug&play tracker for the object detector.

Without the image quality deterioration problem on the ImageNet VID benchmark [165], a well-trained image object detector should perform well for the video data. However, the existing image quality deterioration problem undermines its performance greatly, and various methods have been brought out to best utilize the video data to handle the quality deterioration problem.

Feature aggregation has been a widely used idea for the video object detection task [200, 13, 232, 190, 196, 53, 172, 38, 74].However, feature aggregation is not applicable if there are no annotated labels for certain classes in the video datasets.

Unifying object detection and object tracking is one possible direction worth attention. As tracking searches for similarity between images, it is generally easier to track than to detect an object with deteriorated appearance between consecut-

ive frames. Tracking methods could keep track of each individual instance in a class-agnostic way and they are designed to perform robustly when image quality deteriorates. The class-agnostic property could be the key to tackle the missing label problem for the video object detection task, as it allows the tracker to be successfully trained for tracking objects in a class-agnostic way.

There are mainly two solutions for the object tracking task. The first one is to directly localize each object in consecutive images [111, 14, 110], and the second one is to compare and match object candidates between consecutive images [223, 4]. Our focus is on the first solution, and it should provide better object localization when image quality deteriorates. Tracked objects and newly detected objects could be linked based on the IoU scores of their bounding boxes.

It should be noted that the goals for object detectors and object trackers are different. Object detectors need to be trained for better object recognition ability, while object trackers should be trained for better re-identification and object localization abilities. Such difference requires different training data supply. An object detector is better to be trained with images of relatively good quality, since blurred foreground objects in deteriorate images would misguide and undermine the object classifier. In contrast, deteriorate images are preferred for an object tracker, since the tracker needs to be trained to maintain the precision for object localization even if the image quality deteriorates.

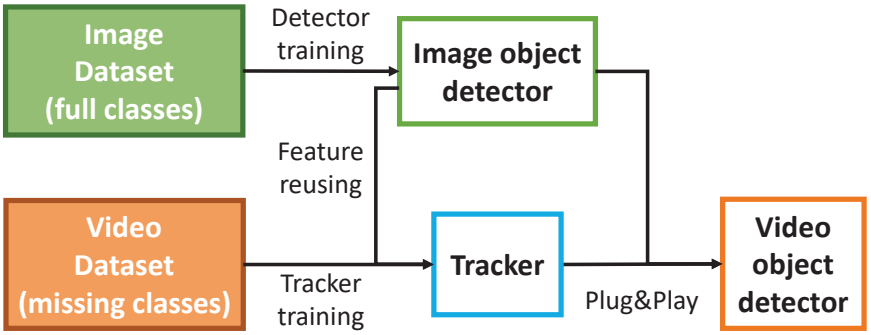

Figure 5.2 Model design for the video object detection task. Firstly, an image object detector is trained with image dataset with full classes. Then, the newly proposed tracker is trained with video dataset by reusing the features from the detector. Lastly, the learned tracker is plugged into the image object detector forming the video object detector.

The tracker design for the video object detection task is different from it is for the video object tracking task, therefore, there are several points that need to be taken into consideration. First, as object detection networks require more powerful recognition ability, the features extracted are often heavier than those in object tracking networks. It would be preferable if a tracker could re-use the features from an object detection network for better efficiency. However, such deep features may not be compatible with some state-of-the-art object trackers, e.g., siamese region proposal networks [111, 110]. The zero paddings during feature extraction in the object detector are not preferred for the siamese region proposal networks as they

affect differently to different anchors in different spatial positions, and the tracking performance would be heavily undermined. Such zero padding problem requires a different tracker design. Another common practice for the single-object tracking task is to crop and resize objects in the original images, which ensures that the template and the target inputs are in pre-defined sizes [111, 110]. Such practice makes the features extracted scale invariant to different sizes of an object in a video. However, this is also not preferred in the video object detection task as an input image needs to be resized to different scales for different objects, which is not preferred for the object detection.

In order to tackle the problem discussed above and improve the combination of object detection and object tracking, we propose a novel siamese convolutional regression tracker for the video object detection task, which takes feature sharing, efficiency, and varied object sizes into account. The design outline is shown in Fig. 5.2.

The **main contributions** of this chapter are the following:

- We have created an object tracker for the video object detection task, which can be easily inserted into a well trained image object detector. Without harming the performance of an object detector, the tracking functionality can be implanted into the model.
- Our tracker is light-weighted, memory efficient, and computationally efficient, as it re-uses the features from an object detector. Our tracker is compatible with the deep features extracted for the object detection purpose.
- Our new tracker performs in adaptive scales according to the sizes of different objects being tracked, which could cope with large object size variation in a video.
- We have designed a new video object detection pipeline to combine the advantages from both object detection and object tracking. With better bounding box proposals and linkages through time, we improve the performance with better effectiveness and efficiency.

5.2 Related work

In this section, methods for the video object detection task and methods related to our work will be introduced.

5.2.1 Video object detection by linking and re-scoring

Object detection in images is one of the fundamental tasks in computer vision, and a number of one-stage and two-stage image object detectors have been proposed recently [159, 26, 123, 121, 156, 124, 48]. One natural solution for the video object detection task is to first detect objects from individual images with image object detectors, and apply post-processing to link and re-score the detection results of all images in a video [75, 96]. Both the image detection part and the linking post-processing part can be fast and efficient. Seq-NMS [75] is a widely used post-processing method. Detected objects are linked by finding max score paths between boxes in consecutive frames under 0.5 IoU constraint. Such constraint is

not optimal as it may not hold for objects with fast movement. Linked detection results are re-scored afterwards by averaging their detection scores. [4] proposes a learning based object linking method for post-processing, which does not rely on the IoU constraint for linking and improves the linking for objects in fast movement.

Object tracking is another option to handle the linking problem. Many methods have been proposed to bring in trackers, but very few methods achieve real integration of detection and tracking in one network. Instead, external trackers [96], optical flows [233, 232, 190], or alternatives for tracking are required [17].

T-CNN [96] proposes a deep learning framework that incorporates an external independent tracker [189] to link the detection results, which makes the pipeline slow and less favored as separate pipelines require twice the time and memory for feature extraction. A smarter way is to share the feature extraction part for both the object detection networks and the object tracking networks, e.g., [223] learns an additional feature embedding to help link the detected objects to the corresponding tracklets. Our model design also adopts the feature sharing strategy.

In [95] tubelet proposal network is utilized to propose tubelet boxes for multiple frames simultaneously, boxes in the same tubelet are linked.

5.2.2 Video object detection by feature aggregation

Another direction to improve the detection results in deteriorate images for the video object detection task is through feature aggregation, which is to use features from multiple frames simultaneously to acquire more temporally coherent augmented features. The short-term feature aggregation [200, 13, 232, 190] methods pre-define a limited range of frames for the feature augmentation, while the long-term feature aggregation achieves longer consistency with better performance [196, 53, 172, 38, 74].

DFF [233], FGFA [232], MANet [190] adopt optical flows to warp the features for alignment. Modern deep learning based optical flow models, such as FlowNet [58] and LiteFlowNet [91], can process images in a very fast speed. However, optical flow models are normally trained with synthetic data and the performance is limited by the domain discrepancy. STMN [200] aggregates the spatial-temporal memory from multiple frames according to the MatchTrans module, which is guided by the feature similarity between consecutive frames. STSN [13] directly extracts the spatially aligned features by using deformable convolutions [49]. [73] adopt progressive sparse local attentions to propagate the features across frames, while [52] utilizes explicit external memory to accumulate information over time. [172, 53, 196] use more powerful attention based relation modules to distill semantic information from longer sequences for object recognition in a video. [38] improves the long-term relation modeling with memory enhanced global-local aggregation. [74] further extends the intra-video relation reasoning with the inter-video relation reasoning to achieve a higher score.

Feature aggregation costs more memory and time during inference as features from multiple frames are used. Feature aggregation does not re-identify and keep track of different object instances either. However, this problem is ignored by the standard performance evaluation method for the ImageNet VID dataset [165], which evaluates in the same way as the image object detection task.

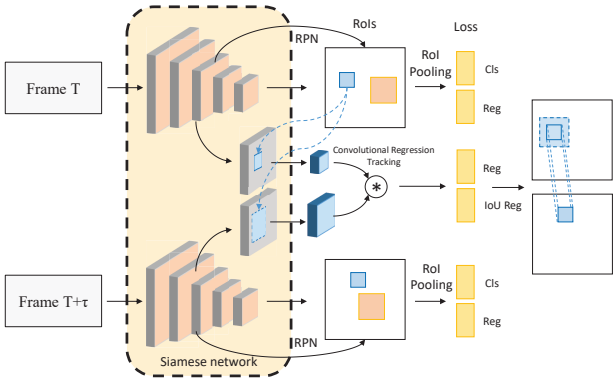

Figure 5.3 Architecture of our video object detection network. Our Plug&Play tracker reuses the features of the detection networks from both branches. The regional features within RoIs are pooled and sent to the tracker. The regional features from the two branches are convolved with each other for bounding box and IoU regression. (Details illustrated in Sec. 5.3).

5.2.3 Methods for the object tracking task

For the video object tracking task, the siamese networks have received much attention recently. GOTURN [79] adopts the fully connected layers to merge features from the siamese network for bounding box regression. [14, 179] score the locations of objects by using feature correlation through a convolution operation between the template patch and the target patch. The idea is extended by [111, 110] with region proposal networks, which infer the object scores and the box regression simultaneously for improved box localization.

5.2.4 Unify object detection and object tracking

D&T [63] brings the tracking into the detection network, which is the most relevant work to ours. By utilizing the feature map correlations between the frames under several pre-defined spatial translations, the model learns a box regression model from one frame to another. D&T is inefficient in memory and speed as it computes the feature map correlations across the whole feature maps with multiple translations. Besides, the tracked boxes are not used for improved object localization, they are used to link the detected objects only.

5.3 Architecture Overview

In this section, we will introduce an overview of our model structure. The goal of our model is to plug the tracking network into the detection network without harming the performance of the image object detector.

The architecture design is shown in Fig. 5.3. Our model takes two consecutive frames with a gap of τ (1 for testing) as inputs $I^t, I^{t+\tau} \in \mathbb{R}^{H\times W\times 3}$, followed by a siamese network for backbone feature extraction. The two branches share the same

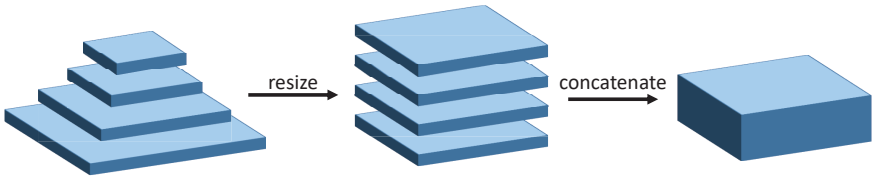

Figure 5.4 Input feature extraction for the tracker in the light model. The features from the HRNet-w32 backbone are used for tracking. The features from all 4 stages are spatially resized to the size of the features in the second stage before concatenation.

weights to keep the feature extractors identical. In order to satisfy the need for object detection in complex scenes, we exploit powerful feature extraction backbones such as HRNet [188] and ResNeXt [77, 202]. The two models correspond to a light model and a heavy model. We further enhance the heavy model with the deformable convolutional networks (DCN) to test the compatibility between the tracker and the deformed features.

The extracted backbone features from the siamese network are sent to the two detection branches and the tracking branch. In the detection branches, regions of interest (RoIs) are proposed by the Region Proposal Network (RPN) [159]. The RoI-wise features are pooled for object classification and bounding box regression, which is the same as the mask-rcnn [76]. One branch of the siamese network plus one detection branch forms a standard two-stage object detector.

In the tracking branch, the novel scale-adaptive siamese convolutional regression tracker is utilized to predict the bounding box transformation from one frame to another. The tracker utilizes regional features from the siamese network, based on the RoIs to be tracked. During training, the RoIs for tracking are generated from the perturbed ground truth bounding boxes, while in the testing phase, the RoIs are the detected objects. Besides the bounding box regression, the tracker has another tracking confidence evaluation branch, which is to evaluate the bounding box regression quality. This is achieved by predicting the IoU scores between the tracked boxes and the ground truth boxes.

The backbone features from the light model, i.e., features from all 4 stages in the last layer of HRNet-w32 [188], and features from the middle 3 stages in the feature pyramid networks(FPN) [120] from the heavy model are utilized as the input for the tracker. Features are resized and concatenated as shown in Fig. 5.4 5.5.

5.4 Scale-adaptive convolutional regression tracker

Many trackers used for the video object tracking task crop and resize the image patches according to the sizes of the objects to be tracked [111, 110, 14]. Such standard sizes for the network input makes feature extraction invariant to the sizes of objects. However, image patch cropping and resizing are not applicable to the detection network as there may be multiple objects of different sizes. Instead of regularizing the size of the network input, we aim to extract scale-adaptive features

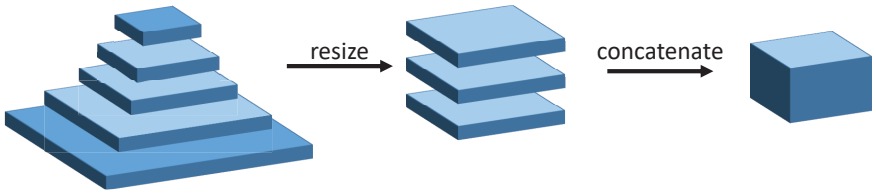

Figure 5.5 Input feature extraction for the tracker in the heavy model. The features from the FPN are used for tracking. The features from the middle 3 stages are extracted and resized to the size of the features in the third stage before concatenation.

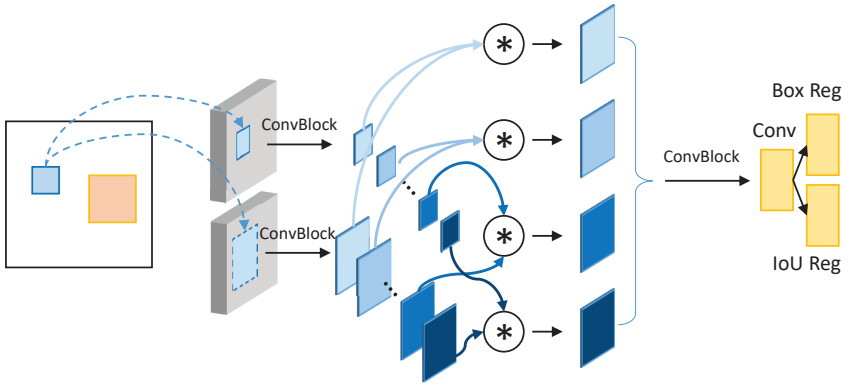

Figure 5.6 Illustration of the depth-wise feature correlation in our tracker. The bounding box of an object determines the location where the tracker acquires the local features. The features of each channel from the first branch are convolved with the features of the same channel from the second branch. Convolutional blocks are inserted to adjust the features, each of which is comprised of a convolutional layer, a batch normalization layer and a relu layer.

for tracking by reusing features from the shared backbone, which augments the detector in a Plug & Play style.

In order to infer object translation from one frame to another, we rely on the feature correlations under a set of different translations. We extract regional features from both of the two branches based on the RoIs to be tracked. The RoI for the first branch marks the bounding box extent of an object. The width and height of the RoI bounding box is expanded k times for the second branch with the center point and the aspect ratio fixed, which marks the local area of interest to search for the object in the second frame. k is set to 3, indicating one object space to each side of the center object, and the pooled feature sizes for the two branches are 21×21 and 7×7, respectively. RoIAlign [76] is adopted to pool the features from the two branches of the siamese network. In order to keep the same scale of the pooled features, the pooled feature size from the second branch is also k times the size of the first branch, as shown in Fig. 5.7. The features pooled from outside the range of the image are set to zero. We adopt the backbone features from multiple stages for the RoIAlign pooling. Features from different stages are resized to the

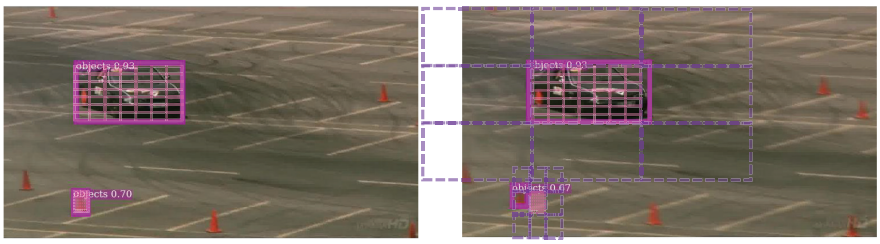

Figure 5.7 A real example of scale-adaptive tracking feature extraction. 7×7 features are pooled within the object bounding boxes from the first image, and 21×21 features are pooled in the searching area from the second image.

same intermediate size as in [144]. Instead of averaging, we concatenate the resized features. For HRNet-W32 [188] backbone in the light model, all stages are adopted. For ResNeXt101 [202] backbone in the heavy model, features from the middle 3 stages are applied.

Our scale-adaptive tracker calculates feature correlation with a depth-wise convolution operation [110] between the two feature patches from the two branches of the siamese network. Fig. 5.6 illustrates the process. For our scale-adaptive tracker, convolutional blocks are inserted before and after the correlation operation for better feature adjustment, each of which consists of a convolution layer with 256 channels, a batch normalization layer and a relu layer. The head of the tracker has a bounding box regression branch and an IoU score (confidence) regression branch, which are two fully connected layers attached to a shared 2D convolution layer with 256 filters. Sigmoid function is applied to normalize the IoU score prediction. As we adopt a class-agnostic regression tracking, the output dimensions are 4 and 1 respectively for each object. Class scores of the tracked objects are assigned by the detection sub-network. The different instances are differentiated according to the translation variant features after RoI pooling. Smooth L_1 loss is utilized for regression [69]. Our tracker learns to predict the target bounding boxes directly, and it can rectify the object localization during tracking.

The memory cost for the tracker is linear to the number of objects being tracked. Each object takes about $3.75MB$ memory, and the total memory cost can be calculated as $N_{obj} \times 3.75MB$, where the N_{obj} is the number of objects being tracked. As we track the objects proposed by the R-CNN rather than the RPN, there are limited objects to be tracked, ranging from 1 to 50. Compared to the memory consumption for the detection network (More than $1GB$), the tracker is a very light weighted increment.

An example of extra memory cost derivation for the tracker in the heavy model is shown in Table 5.1. For the features in the convolutional block before and after the correlation in Table 5.1, $\times 3$ means the number of features due to convolution, batchnorm and relu layers. We add up all the memory cost and multiply it by 4 as we use 32 bit precision for the network. The total memory cost is $3,932,180$, which equals 3.75 MB.

For our tracker design, the feature correlations of different translations are naturally encoded into different spatial positions in the output features. In contrast,

Feature source	Feature size derivation	Feature size
Pooled features in two branches	$7 \times 7 \times 768 + 21 \times 21 \times 768$	376,320
Features in the convolutional block before correlation	$(7 \times \times 7 \times 256 + 21 \times 21 \times 256) \times 3$	376,320
Features after correlation	$15 \times 15 \times 256$	57,600
Features in the convolutional block after correlation	$15 \times 15 \times 256 \times 3$	172,800
Features for bounding box regression and confidence prediction	$4 + 1$	5

Table 5.1 Example of feature size derivation for the tracker in the heavy model.

D&T [63] encodes them into different channels. For D&T [63], the translations are predefined within a fixed square window, which is invariant to the object being tracked. In our method, the translations are defined within a rectangle, which scales with the object size. Larger translations are applied for larger objects. The intuition is that if an object is closer to the camera, it would be larger in size and move faster in image.

The learning targets are defined by the bounding boxes to be tracked $b^t = (b_x^t, b_y^t, b_w^t, b_h^t)$ in time t, the predicted bounding boxes $p^{t+\tau} = (p_x^{t+\tau}, p_y^{t+\tau}, p_w^{t+\tau}, p_h^{t+\tau})$ in $t + \tau$ and the corresponding ground truth bounding boxes $g^{t+\tau} = (g_x^{t+\tau}, g_y^{t+\tau}, g_w^{t+\tau}, g_h^{t+\tau})$ in $t + \tau$. The targets for bounding box regression $\Delta^{t+\tau} = (\Delta_x^{t+\tau}, \Delta_y^{t+\tau}, \Delta_w^{t+\tau}, \Delta_h^{t+\tau})$ are:

$$\begin{aligned} \Delta_x^{t+\tau} &= \frac{g_x^{t+\tau} - b_x^t}{b_w^t} \\ \Delta_y^{t+\tau} &= \frac{g_y^{t+\tau} - b_y^t}{b_h^t} \\ \Delta_w^{t+\tau} &= \ln \frac{g_w^{t+\tau}}{b_w^t} \\ \Delta_h^{t+\tau} &= \ln \frac{g_h^{t+\tau}}{b_h^t} \end{aligned} \tag{5.1}$$

The target for predicted IoU score $p_{score}^{t+\tau}$ is calculated by $b_{score}^{t+\tau} = IoU(p^{t+\tau}, g^{t+\tau})$, $b_{score}^{t+\tau} \in [0, 1]$. The predicted bounding box $p^{t+\tau}$ are inferred through predicted bounding box regression $\hat{\Delta}^{t+\tau} = (\hat{\Delta}_x^{t+\tau}, \hat{\Delta}_y^{t+\tau}, \hat{\Delta}_w^{t+\tau}, \hat{\Delta}_h^{t+\tau})$ as following,

$$\begin{aligned} p_x^{t+\tau} &= \hat{\Delta}_x^{t+\tau} b_w^t + b_x^t \\ p_y^{t+\tau} &= \hat{\Delta}_y^{t+\tau} b_h^t + b_y^t \\ p_w^{t+\tau} &= \exp(\hat{\Delta}_w^{t+\tau}) b_w^t \\ p_h^{t+\tau} &= \exp(\hat{\Delta}_h^{t+\tau}) b_h^t \end{aligned} \tag{5.2}$$

Our feature extraction design for tracking has several advantages. First, only resized local features are utilized, which makes the tracker memory efficient. Second, features extracted from the two branches are of the same spatial resolution and the translations for feature correlation are adaptive to the scale of an object. It should be noted that the learning target $\Delta^{t+\tau}$ of the bounding box regression for tracking is scale-normalized, which aligns with the scale-adaptive feature convolution operation. Finally, the tracker is light-weighted as it reuses the features from the backbone network.

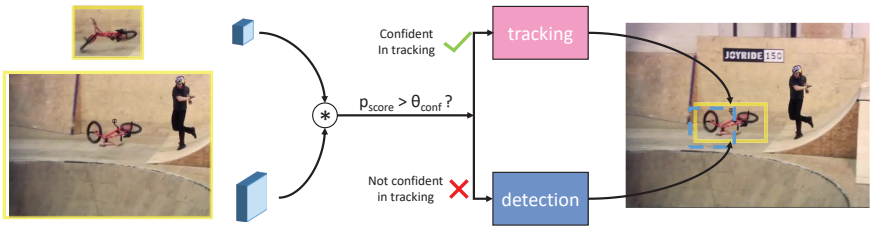

Figure 5.8 Strategy for the bounding box selection. Bounding boxes can be inferred from both the object detection and the object tracking. We choose the object according to confidence in tracking.

5.5 Unify detection and tracking

Our model has functionalities for both detection and tracking. In this section, we will introduce how detection and tracking interact with each other in our model. The aim of the detection is to find the newly appeared objects in an image, while the tracking is to better localize objects across frames if the tracking is reliable.

Detect to track. The two-stage object detection network has two object proposal stages, the RoI proposals by the RPN and the detected objects by the R-CNN. We base our tracking on top of the detected objects instead of the RoI proposals, as the R-CNN provides fewer, cleaner, and more accurate objects for tracking. The design of RPN is to provide proposals with high enough recall rate and it is not time or memory efficient to track hundreds of RoIs. Setting a score threshold for the foreground proposal selection would not be proper either, as the scores and the bounding boxes would not be accurate enough due to the anchor based design. The detection results from R-CNN is more robust. If an object is identified in an image with a high enough class score (0.03 in our case), it would be a candidate for tracking in the next image.

Track to detect. Tracking could aid the detection by offering better object localization, which could also be achieved by the detection network. We choose the object adaptively according to the confidence for the object tracking. We first filter out the tracked objects with low confidence scores. If the predicted tracking confidence score is lower than a threshold θ_{conf} (0.5 in our case), indicating bad tracking results for an object, the tracked object is discarded. As multiple objects may occlude each other during tracking, we apply a non-maximum suppression (0.7 IoU threshold in our case) to the tracked boxes based on the IoU scores. The confidence score can also help select the front objects in tracking. The selected tracked objects are combined with the newly detected objects for the video object detection in the new frame. When an object is identified by both the detection network and the tracking network, i.e., objects proposed by the detection network and the tracking network, have large enough IoU, we choose the tracked one over the detected one, as shown in Fig. 5.8. We call it the tracking-first detection (TFD) strategy as the tracked objects are preferred if they are with good enough confidence. The TFD strategy is preferred due to the fact that non-maximum suppression in the object detection process favors the objects with higher class scores, shown within the

blue dashed box, instead of the objects with more accurate bounding boxes, shown within the yellow box. We prefer better bounding boxes by the TFD strategy to help acquire improved object linking across frames. Only newly detected objects that have IoUs with the tracked ones lower than a threshold T_{merge}^{nms} are reserved. During the inference across all frames in a video, the tracked boxes are saved, and the detection results are refined by average re-scoring and non-maximum suppression as in [75].

5.6 Experiments

The experiments are designed to answer two fundamental questions in our model design. First, whether the features from image object detectors can be re-used by the tracker. Second, how does the model perform if certain classes are missing in the video dataset. We show the effectiveness of feature re-using by training on video dataset with full classes. The performance with missing classes in the video dataset are reported by training the tracker with only the first half of all the classes.

5.6.1 Dataset and evaluation

Our method is evaluated on the ImageNet [165] video object detection (VID) dataset. There are 3862 training and 555 validation videos with objects from 30 classes labelled for the task. The ground truth annotations contain the bounding box, the class ID and the track ID for each object. The performance of the algorithm is measured with mean average precision (mAP) score over the 30 classes on the validation set as it is in [232, 190, 63, 96, 95, 200, 13]. In addition to the ImageNet VID datset, the ImageNet object detection (DET) dataset has 200 classes, which include all the 30 classes in the ImageNet VID dataset as well. We follow the common practice by utilizing data from both the DET and the VID datasets [232, 190, 63, 96, 95, 200, 13]. Compared with the VID dataset, the DET dataset contains static images with better quality. The training is separated into an image training stage and a video training stage. The image training stage focuses on training the object detector with better image qualities (relatively less data from the VID dataset), while the video training stage aims at training the object tracker with more deteriorate images (more data from the VID dataset).

5.6.2 Configurations

Image training stage. In the first stage, we train the detection parts of our video object detector in the same way as a standard object detector. The training samples are selected from both the DET and the VID dataset. To balance the classes in the DET dataset, we sample at most $2K$ images from each of the 30 categories to get our DET image set ($53K$ images). To balance the VID videos, which have large sequence length variations, we evenly sample 15 frames from each of the video sequence to get our VID image set ($57K$ images). The combined DET+VID image set is used for detector training.

The anchors in RPN have 3 aspect ratios $(0.5, 1, 2)$ spanning 5 scales, which are $(32, 64, 128, 256, 512)$. For the ResNeXt model with FPN, 5 scales are distributed to the 5 stages in the FPN pyramid.

In the detector, the RoIAlign pooling extracts features to be of size 7×7. The pooled features are the average features of the 4 nearest points, which offers higher classification accuracy. For the tracker, the pooled features are of sizes 7×7 and 21×21, which is neither too large nor too small to balance the speed and accuracy tradeoff. The pooled features are the average of the features in the corresponding bin, which makes the features for correlation less sensitive to scales.

The R-CNN has a bounding box regression branch and a logistic regression branch for classification. The two branches have 2 shared fully connected layers with 1024 filters, attached with 1 fully connected layer in each branch for their own prediction.

In ResNeXt [202], the DCN [49] with 32 groups and modulation [230] is applied for stage 3,4,5, which has been a standard configuration in ResNeXt [230].

We apply SGD optimizer with a learning rate of 10^{-3} for the first $90K$ iterations and 10^{-4} for the last $45K$ iterations. Online hard example mining (OHEM) [171] is utilized for R-CNN training to acquire better detection performance.

The training batch is set to 8 images that are distributed among 4 gpus. In both training and testing, we apply a single scale with the shorter dimension of the images to be 600 pixels, which offers a balanced scale for different objects in the dataset, and is of the same setting as in [63, 232]. During training, only random left-to-right flip is used for data augmentation.

Video training stage. In this stage, we train our tracking parts with the image pairs from the ImageNet VID dataset. We randomly select two consecutive images with a random temporal gap from 1 to 9 frames, so that the objects in the image pairs have various relative positions, while not being too far for tracking. As there is no causal reasoning involved, we randomly reverse the sequence order to gain more variety of translations. The RoIs for tracking $R = (R_x, R_y, R_w, R_h)$ are generated by resizing and shifting the ground truth bounding boxes $g = (g_x, g_y, g_w, g_h)$ as in Eq. 5.3. The noise is added as data augmentation to simulate the imperfect localization for the input objects. The coefficients $\delta = (\delta_x, \delta_y, \delta_w, \delta_h)$ are sampled from uniform distributions U. $\delta_x, \delta_y \in U[-1.0, 1.0]$ and $\delta_w, \delta_h \in U[0.5, 1.5]$. For each ground truth object, we sample 256 RoIs and randomly select 128 RoIs from those satisfying the constraint of $IoU(R, g) > 0.5$, which ensures proper quality for the inputs during training.

$$
\begin{aligned}
R_x &= \delta_x g_w + g_x \\
R_y &= \delta_y g_h + g_y \\
R_w &= \delta_w g_w \\
R_h &= \delta_h g_h
\end{aligned}
\tag{5.3}
$$

We freeze the backbone parts to train the tracking parts only in order to retain accuracy for detection. RPN and R-CNN are not included either. The model is trained with SGD optimizer with a learning rate of 10^{-3} for the first $80K$ iterations and 10^{-4} for the next $40K$ iterations. We apply a batch of 16 image pairs for

training that are distributed among 4 gpus. The images are resized to the same single scale as the image training step.

Testing. In the testing stage, we select the detected objects with the class scores higher than 0.01 for tracking, which provides object candidates with a high recall rate. An IoU threshold of 0.45 is applied for the non-maximum suppression on the final detection output, which balances the precision and recall as in [53]. Single scale testing is used with the shorter side of images resized to 600 pixels as in training.

Implementations. Our model is implemented with pytorch [146] and integrated with MMDetection [34]. The source code will be released.

5.6.3 Results

We compare several major competitive video object detection algorithms in Tab. 5.2. FGFA [232] and MANet [190] utilize optical flow to guide linking between frames, but they cannot achieve a good balance between the mAP score and the speed. The extra optical flow limits their speed to 1.15 FPS and 4.96 FPS respectively. STMN [200] and STSN [13] aggregate pixel level information from multiple frames, which limits their speed greatly (1.2 FPS for STMN and not reported for STSN). More recent PSLA [73] and EMN [52] have comparable performances, which adopt attention or memory for temporal linking, but they cannot preserve object identities in tracking. D&T [63] has strong performance, but we have better memory efficiency and speed for tracking, which is explained in the ablation study. The long-term feature aggregation methods are in the leading places of the ImageNet VID benchmark, including RDN [53], SELSA [196], MEGA [38], and HVR-Net [74], which all have mAP scores more than 83%. However, these methods cannot be used when there is no annotated labels for certain classes in the video object detection task. RDN [53] and MEGA [38] only report the speed of their lighter version models without post-processing, which are around 10 and 9 FPS evaluated on more powerful GPUs, Tesla V100 and RTX 2080ti, respectively. The speed for the top scores shown in Table 5.2 should be much slower. With the novel light-weighted tracker, our light and heavy models achieve performance of 78.6% and 81.1% respectively, which is comparable to some of the state-of-the-art methods. It should be noted that only our model handles the problem of missing classes in the video dataset, as the detector trained from the image dataset is fixed. When half of the classes are trained, the performance only drops by 0.4% compared to training all classes in the video dataset. More details are shown in the ablation study.

5.6.4 Ablation study

To test the effectiveness of bringing the tracker into the model, we perform ablation study by gradually adding the components. We start with the per-image detection model. The faster-RCNN with HRNet backbone is adopted as the basic model. We first test the standard detection result without any aid from tracking. The performance score is shown in Tab. 5.3. The per-image detection achieves a decent mAP score of 73.2%. We further add Seq-NMS [75] to see the performance of linking and re-scoring based solely on the detection results. As in [75], we set the IoU threshold for boxes linking constraint to be 0.5 and IoU threshold for detection

Methods	Temporal linking	mAP (%)	FPS
FGFA [232]	optical flow	78.4	1.15
FGFA+ [232]	optical flow	80.1	1.05
MANet [190]	optical flow	78.1	4.96
MANet+ [190]	optical flow	80.3	-
STMN [200]	STMM	80.5	1.2
STSN [13]	DCN	78.9	-
STSN+ [13]	DCN	80.4	-
D&T [63]	box regression	79.8	7.09
PSLA [73]	attention	77.1	**18.7**
PSLA+ [73]	attention	81.4	5.13
EMN [52]	memory	79.3	8.9
EMN+ [52]	memory	81.6	-
RDN [53]	attention	83.2	-
RDN+ [53]	attention	83.9	-
SELSA [196]	attention	84.3	-
SELSA+ [196]	attention	83.7	-
MEGA+BLR [38]	attention+memory	85.4	-
HVR-Net [74]	attention	84.8	-
HVR-Net+ [74]	attention	**85.5**	-
Ours(HRNet-w32)[full]	box regression	78.6	11
Ours(ResNeXt101*)[full]	box regression	81.1	6
Ours(ResNeXt101*)[half]	box regression	80.7	6

Table 5.2 Comparisons among different video object detection methods. + stands for Seq-NMS [75]. * means with FPN [120] and DCN [49]. [full] means full classes training on the video dataset. [half] means first half classes (15 classes) training on the video dataset.

NMS to be 0.45. The average precision scores improve for all categories and the mAP score has increased by 2.3%. We further add tracking into our model by adopting our TFD video object detection strategy. During the training of the tracking modules, we freeze the parameters of the feature extraction backbone, the RPN and the R-CNN in order to control the performance of the object detector. We compare two values for merging NMS IoU threshold T^{nms}_{merge}, 0.3 and 0.7 (marked as TFD(0.3) and TFD(0.7) in Tab. 5.3). The mAP scores have increased another 2.3% and 3.1% respectively. The mAP score is higher with $T^{nms}_{merge} = 0.7$, showing that the video object detector still benefits more from the denser object proposals. The TFD strategy is very effective in helping to improve the quality of linking and re-scoring. By now, we have improved the performance of the object detector by a large margin (+5.4% mAP) without modifying any parameters of an object detector. With heavier ResNeXt101 [202] backbone plus FPN [120] and DCN [49], the direct object linking with Seq-NMS from the detection results only achieve 77.2% mAP, improving by only 1.0%. With the tracker, TFD+Seq-NMS improves the detector by a large margin (+4.2% mAP) from 76.2% to 80.4%.

Seq-NMS links boxes across frames under constraint $IoU(b^t, b^{t+1}) > 0.5$, which fails if there is large position translation between consecutive frames. We improve the constraint to be $IoU(p^{t+1}, b^{t+1}) > 0.5$, where p^{t+1} is the predicted

Methods	airplane	antelope	bear	bicycle	bird	bus	car	cattle	dog	d_cat	elephant	fox	g_panda	hamster	horse	lion
HRNet-w32	83.7	82.8	79.6	73.4	72.0	68.3	58.0	68.5	65.5	73.4	75.8	86.1	81.4	91.9	69.2	48.2
HRNet-w32+Seq-NMS	83.8	85.1	83.8	73.8	72.5	70.0	59.0	70.8	67.9	78.5	76.7	88.7	81.8	96.2	70.6	58.3
HRNet-w32+TFD(0.3)+Seq-NMS	80.9	84.6	**86.9**	73.6	**74.0**	74.0	56.4	82.6	72.0	87.8	76.0	96.1	82.3	98.0	74.3	62.3
HRNet-w32+TFD(0.7)+Seq-NMS	87.0	**86.0**	85.6	**76.6**	71.7	75.0	56.6	80.8	72.5	89.8	**80.4**	95.4	82.4	98.8	79.3	63.8
ResNeXt101*	87.9	78.7	77.5	73.3	71.8	82.5	**60.5**	73.6	70.7	82.0	75.6	90.7	86.0	91.6	74.4	57.3
ResNeXt101*+Seq-NMS	86.2	81.3	80.9	73.3	71.3	84.5	57.8	74.3	73.9	86.6	73.8	91.6	82.3	97.4	74.8	64.5
ResNeXt101*+TFD(0.7)+Seq-NMS	**88.5**	84.3	81.8	74.6	72.5	**89.0**	58.7	85.2	79.7	**92.3**	78.6	**98.7**	**86.6**	98.8	80.1	66.7
ResNeXt101*+TFD(0.7)+Seq-Track-NMS	88.3	84.7	83.2	74.6	72.8	88.8	58.5	**85.3**	**80.5**	**92.3**	78.7	**98.7**	86.2	**98.9**	80.4	**71.2**
ResNeXt101*+TFD(0.7)+Seq-Track-NMS+[half]	87.6	85.0	82.4	73.9	72.4	88.7	59.1	84.6	80.1	91.8	78.6	98.6	85.4	98.2	**81.7**	71.1

Methods	lizard	monkey	motorcycle	rabbit	red_panda	sheep	snake	squirrel	tiger	train	turtle	watercraft	whale	zebra	mAP (%)
HRNet-w32	82.4	47.1	81.2	70.8	81.2	55.1	73.5	56.2	89.7	78.0	79.2	63.8	70.1	91.0	73.2
HRNet-w32+Seq-NMS	84.0	49.4	83.3	75.0	87.9	57.3	74.2	57.2	90.2	78.3	80.7	65.2	72.5	91.0	75.5
HRNet-w32+TFD(0.3)+Seq-NMS	86.8	48.6	83.9	80.5	**94.8**	55.3	74.9	63.1	90.0	80.3	82.7	70.3	67.6	93.6	77.8
HRNet-w32+TFD(0.7)+Seq-NMS	**88.3**	51.7	**87.8**	80.2	93.2	58.3	73.7	57.3	89.9	82.3	83.0	**72.2**	67.1	91.0	78.6
ResNeXt101*	79.2	52.2	82.2	74.9	71.5	61.8	**79.7**	58.2	**91.9**	86.0	81.2	67.4	**73.6**	91.5	76.2
ResNeXt101*+Seq-NMS	80.9	51.5	84.9	79.7	75.2	58.6	77.7	61.4	90.4	85.8	82.0	67.8	72.7	91.5	77.2
ResNeXt101*+TFD(0.7)+Seq-NMS	83.0	55.1	87.7	82.0	76.4	**64.4**	77.1	**69.2**	91.7	**86.4**	85.2	70.5	72.9	94.0	80.4
ResNeXt101*+TFD(0.7)+Seq-Track-NMS	82.8	**55.2**	87.7	82.1	90.8	**64.4**	76.9	69.1	91.7	**86.4**	85.3	70.7	72.4	**94.6**	**81.1**
ResNeXt101*+TFD(0.7)+Seq-Track-NMS+[half]	83.3	54.7	86.8	**82.8**	90.1	64.3	76.4	66.1	91.6	85.7	**85.4**	69.7	71.9	93.4	80.7

Table 5.3 Ablation study of our method. TFD stands for tracking-first detection. * stands for with FPN [120] and DCN [49]. [half] means first half classes (15 classes) training on the video dataset.

box from b^t by the tracker. The improved re-scoring method (Seq-Track-NMS) provides another 0.7% mAP score boost. Seq-Track-NMS for the light model does not improve or degrade further for the mAP score. The reason is that the tracking performance for the light model is not as strong as the heavy model.

When the tracker is trained with only the first 15 classes, there is only a 0.4% performance drop resulting in a mAP score of 80.7%. If we only consider the second half of the 15 classes for evaluation, the full classes training has a mAP score of 78.75%, while the half classes training has a mAP score of 78.22%, which is only 0.53% lower but still much higher than the mAP score of 74.97% by the ResNeXt101* plus Seq-NMS model. This shows that our class-agnostic tracker, which reuses the features from the detector, could generalize for different classes.

5.6.5 Visualization of the correlation features

Since the template branch has informative features for object localization, it is possible that only one branch is needed for tracking. To ensure that the tracker utilizes the features from both branches rather than being dominated by one branch, we randomly sample 20 channels from the features before and after the correlation operation for visualization, as shown in Fig. 5.9. The examples are air-plane, elephant and motorcycles. The two motorcycle examples are from two consecutive images, where the second one shows an inverse sequence order for tracking. For each image, the upper two rows are the correlation features from the two branches while the bottom row shows the correlation output. The randomly selected 20 channels are ordered in 20 columns. The center of a feature map is marked with a red dot for spatial reference. It should be noted that the template and the target patches are encoded in the same scale, but of different sizes. It is interesting to notice that the center of mass of the template features are shifted to different positions. The features of different objects are distinctive and the correlation output is greatly affected by both branches. By examining the examples of the two motorcycles, which have opposite moving patterns, it could be seen that the features from the target branch determine the tracking prediction, while the features from the template

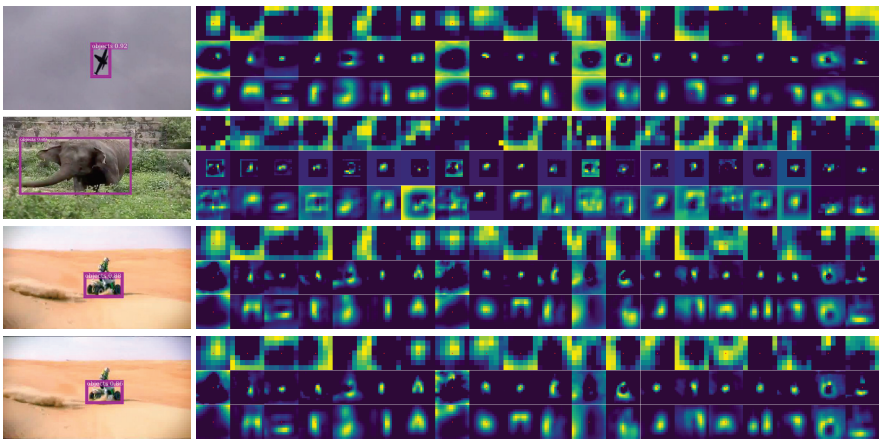

Figure 5.9 Correlation feature visualization. Images on the left show the objects being tracked. Feature maps on the right are the correlation features and the output features. For feature maps of each image (3 rows in total), the upper two rows are the correlation features from the two branches, while the bottom row shows the convolution output. The randomly selected 20 channels are ordered in 20 columns. The center of a feature map is marked with a red dot for spatial reference.

branch are more similar. In conclusion, the tracking prediction is mainly affected by the relative position of an object. The template features of different objects would be encoded differently. And the features from both branches affect the tracking prediction at the same time.

5.6.6 Computational efficiency

The aim of our tracker design is to be light-weighted in both time and memory. In this section, the computational efficiency for the tracker is examined. We analyze for both the light model and the heavy model. All the experiments are conducted with a single Titan X (Pascal) GPU during testing stage with a supported driver version of CUDA8.0. The CPU is of type Intel(R) Xeon(R) CPU E5-1650 v4.

We test the running time of different components in our models with the same image resolution (1000×600) as in [63]. The approximate time costs are reported in Tab. 5.4 with ratios of different components marked. Our tracker is very efficient and only takes an extra time of 3 ms for the light model, and 4 ms for the heavy model, which is lighter than the RPN or the R-CNN. It is also much faster than the tracker in D&T [63], which takes 14 ms to run.

For the video object detection, we examine the running speed of our pipeline and analyze the effect of different components. The speed is measured in frames per second (FPS). The detector with HRNet-w32 backbone runs at 15 FPS and our tracker embedded pipeline runs at 12 FPS. The additional Seq-NMS slows down our pipeline slightly to 11 FPS. As our detector with ResNeXt101 backbone is combined with FPN [120] and DCN [49], it runs relatively slower but still much faster than methods like FGFA+ [232], STMN [200] and MANet+ [190], which run at speed below 5 FPS.

Model	HRNet-w32	ResNeXt101*
Backbone	43 (66.2%)	54 (52.4%)
RPN	12 (18.5%)	32 (31.1%)
R-CNN	7 (10.8%)	13 (12.6%)
Tracker	3 (4.6%)	4 (3.9%)

Table 5.4 Time cost comparison of different components measured in milliseconds (ms). * stands for with FPN [120] and DCN [49].

Backbone	Detector	TFD	Seq-NMS	FPS
HRNet-w32	✓	✗	✗	15
HRNet-w32	✓	✓	✗	12
HRNet-w32	✓	✓	✓	11
ResNeXt101*	✓	✗	✗	7
ResNeXt101*	✓	✓	✓	6

Table 5.5 Time efficiency comparison of the pipelines. * stands for with FPN [120] and DCN [49].

Figure 5.10 Tracking examples by our Plug & Play tracker. Our tracker can track single or multiple objects and can handle problems like motion blur, partial occlusion and rare object poses. The objects are labelled with IoU regression scores.

5.6.7 Qualitative results

Our tracker learns to track single or multiple objects and can handle problems such as motion blur, partial occlusion and rare object poses, as shown in Fig. 5.10. By incorporating tracking networks into the detection networks, our video object detector can achieve very long term detection consistency across frames. Fig. 5.11 5.12 5.13 are examples that show how the detector and the tracker collaborate. The detector finds new objects, while the tracker follows the objects and provides better boxes for linking and re-scoring. Without re-scoring, the detection results could be incorrect or weak. After re-scoring, the long term consistency can be achieved. In the third row of Fig. 5.11 5.12, the dog is wrongly classified as a bird or a fox before re-scoring, and the lizard is wrongly classified as an elephant or a squirrel. After re-scoring, the wrong classifications are corrected with long term score consistency preserved. In the third row of Fig. 5.13, the tree in the background could be wrongly classified as monkey, which is rectified to be background after re-scoring. Thanks to the tracking, a long term object linking with good quality can be achieved, which benefits both background and foreground objects.

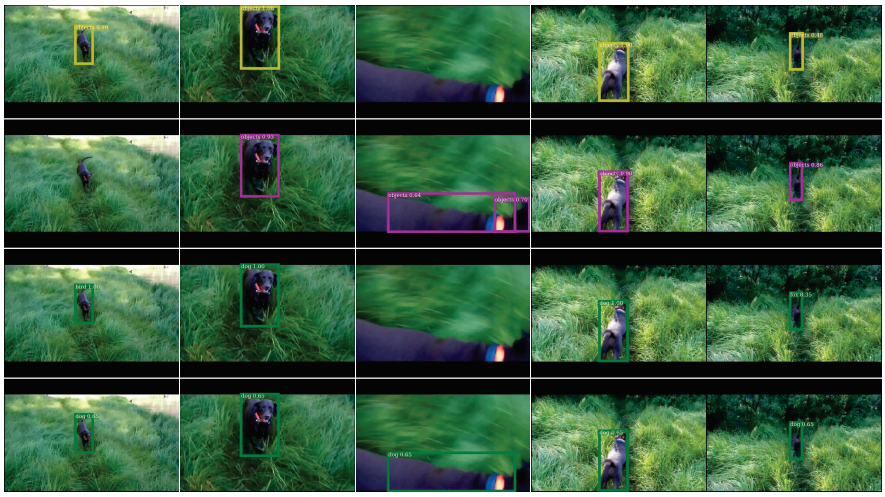

Figure 5.11 Example of dog. The first row shows the proposed objects from the detector. The second row shows the proposed objects from the tracker. The third row shows the tracking-first detection (TFD) results based on the proposed objects from both the detector and the tracker. The fourth row shows the re-scored detection results. Only the objects with scores above 0.2 are shown. The false bird and fox detection results, and the missing dog detection result shown in the third row are rectified as shown in the fourth row.

5.7 Conclusion

We have proposed a Plug & Play convolutional regression tracker that augments image object detectors for the video object detection task. The tracker utilizes the deep features from image object detectors for tracking with very little extra memory

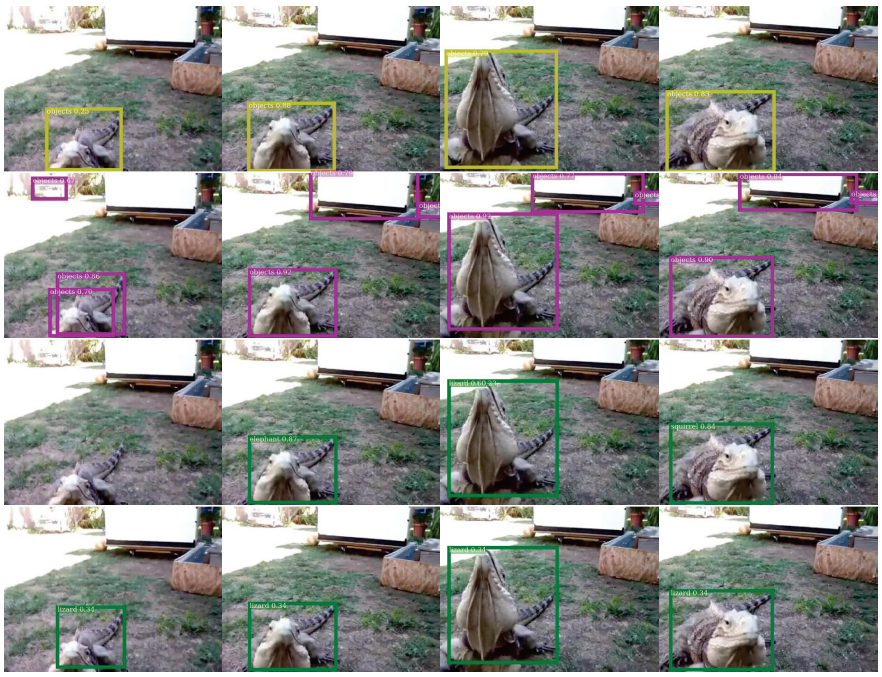

Figure 5.12 Example of lizard. The four row image layout is the same as in Fig. 5.11. The false elephant and squirrel detection results are rectified to correct lizard.

Figure 5.13 Example of red panda. The four row image layout is the same as in Fig. 5.11. The false monkey detection result in the background is rectified.

and time cost. The light-weighted tracker can track a single object or multiple objects, and handle the problem of image deterioration. With our tracking-first detection strategy for better object localization and linking, the performance of the detector improves by a large margin. 5% mAP boost for the image object detector or around 3% mAP boost for the image object detector plus the Seq-NMS post-processing. Our model design can also effectively improve the performance of an image object detector for the video object detection task even if some classes are not available in the video dataset.

Joint Inference for Multi-Object Tracking and Segmentation

6

Abstract

In this chapter, we propose a new deep neural networks (UMotsNet) for the multi-object tracking and segmentation (MOTS) task. The UMotsNet combines the cues from both of the bottom-up and the top-down branches for the instance segmentation and the object tracking. The top-down and the bottom-up cues are mutually complementary and both offer useful features for the MOTS task. The top-down branch provides instance level information, while the bottom-up branch aggregates local information for instance segmentation and tracking. The unified inference allows us to take advantage of the mask refinement module and the relation guided unified instance association for better performance. Our model operates in an online fashion with a speed of 7 fps on a Titan X(Pascal) GPU and achieves a high score of 76.5% (car) and 57.1% (pedestrian) for the ***sMOTSA*** metric on the KITTI MOTS benchmark testset, which exceeds the baseline (Track-RCNN) by a large margin without any auxiliary data such as depth maps or optical flows.

6.1 Introduction

The focus of this chapter is on multi-object tracking and segmentation (MOTS) [182], which has been one of the fundamental research field for video scene understanding in computer vision. Previous results [137] have shown that performance of bounding box level multi-object tracking (MOT) [68] has been saturating. In addition to MOT task, MOTS task further brings the pixel level labels in order to improve the overall performance. The MOTS task incorporates three sub-tasks: detection, segmentation and tracking. The challenges of the MOTS task are that objects enter or leave the scene at any time, while the appearances change over time and often with long-time occlusion among objects. Preserving the consistency of the instance identities and the masks across frames is the major objective.

The combination of the first two sub-tasks, i.e., detection and segmentation, is also known as the instance segmentation task. There are two major approaches for the instance segmentation task, the bottom-up [5, 7, 141] and the top-down [76, 18] approaches, both of which have been actively researched. Recent research [33] also explores a joint approach for instance segmentation. The top-down approaches gather the instance level information and predict object masks based on the instance

Figure 6.1 Example predictions of UMotsNet for the KITTI MOTS dataset. The goal of the MOTS task is to provide accurate and consistent instance identities and masks over time.

level features. The bottom-up approaches predict semantic class for each pixel and group them into meaningful objects based on certain heuristics. For example, instances are separated by edge [104] or grouped by distance of pixel embedding [141] or by the distance of predicted object centers for each pixel [40]. Segmentation quality can be improved by taking advantage of both of the bottom-up approach and the top-down approach [221, 33]. Our idea is inspired by this observation, and we take advantage of the complementary top-down cues and the bottom-up cues for object recognition.

One strategy for the MOTS task is tracking-by-detection. There are two steps involved, the detection step and the association step. Objects are detected and segmented in each frame in the detection step and linked across frames according to the features of objects in the association step [152, 182]. This strategy has been dominating in the MOT [137] task. Features of the detected objects are mapped to instance embedding and certain measures of distance, e.g., inner product [193] or euclidean distance [152], are used to guide the association of the objects.

Previous work has either tackle the problem in the tracking-by-detection paradigm from either the bottom-up approach [206, 207] or the top-down approach [152, 182, 130]. We argue that combining both of the top-down and the bottom-up cues are beneficial for both of the instance segmentation and the tracking performance in the MOTS task. In this chapter, we therefore introduce UMotsNet, which unifies the cues from both of the bounding box level and the pixel level, to simultaneously detect, segment and track multiple objects. We explore the usability of the newly designed mask refinement module for better instance segmentation and the relation

guided top-down and bottom-up unified instance association. In summary, the **contributions** of this chapter are following:

- We present a novel network for the multi-object tracking and segmentation task, which utilizes both of the bottom-up and the top-down cues for better instance segmentation and association.
- We propose the novel mask refinement module for segmentation improvement and the relation guided unified instance association method for better tracking.
- We have achieved a high score for the KITTI MOTS benchmark [182] without using any additional auxiliary data such as optical flow or depth map.

6.2 Related Works

In this section, we review tasks and methods that are related to the MOTS task and our method.

Multi-Object Tracking and Segmentation. The Track-RCNN [182] is proposed with the KITTI MOTS dataset, serving as a baseline for the MOTS task [182]. It extends the Mask R-CNN [76] with additional 3D convolutions for temporal context integration and an extra association embedding head for object re-identification. MOTSNet [152] is extended from the Mask R-CNN with mask-pooling in the association embedding head for better association features, which is trained on automatically harvested training data [152]. Mask R-CNN has also been extended with variational autoencoder in [119], which learns embedded spatial interdependence and motion continuity in video data, to reduce false negative in detection.

In addition to the top-down approaches above, which detect, segment and track objects in instance level. There are methods tackling the MOTS task with bottom-up approaches. STEm-Seg [6] and STE [84] directly group pixels in spatial-temporal domain into instances with coherent identities across frames according to their embedding. Such approaches simplify the pipeline as the instance segmentation and the object identities are jointly inferred. PointTrack [206, 207] relies on the state-of-the-art bottom-up instance segmentation [141], which effectively detects and segments objects in crowded scenes. With its tracking-by-point paradigm, it has won the CVPR2020 MOTS Challenge.

Multi-modality is also explored for the MOTS task. Including 3D information is beneficial to tracking. MOTSFusion [130] utilizes optical flow and depth maps to complement the image input of videos. Bepix [168] utilizes 3D position and orientation information for more accurate tracking. The 3D information helps to recognize objects under occlusions, which improves the overall performance for the MOTS task. However, methods with 3D modalities are often slower than methods with 2D modalities only.

Video Instance Segmentation. Video instance segmentation (VIS) task [208] is similar to the MOTS task, where objects need to be detected in pixel level with coherent identity across frames. It focuses on general objects and is evaluated with average precision (AP) and average recall (AR) metrics in sequence level instead. Tracking-by-detection paradigm is also used for the VIS task, e.g., MaskTrack R-CNN [208] and SipMask [28] re-identify objects with learned object association embedding. Mask propagation is utilized in [12] to propagate instance masks across

frames. The propagated masks are adopted to improve the quality of the instance segmentation and the object association.

Multi-Object Tracking. Multi-Object Tracking (MOT) task is the predecessor of the MOTS task. It only focuses on bounding box level object detection and association. DeepSORT [195, 194] brings deep appearance descriptor for the SORT [15] tracking algorithm, which applies Hungarian algorithm [105] for object matching in an online fashion. Graph networks are used globally to handle the difficult cases with occlusions [112, 19]. Although methods with separate object detector and tracker provide good performance [227, 217, 62], recent work has shown that a joint detector and tracker optimization is beneficial [193] to the performance and the running speed.

Panoptic segmentation. Panoptic segmentation [103] is introduced to unify the instance segmentation and the semantic segmentation task, which has been actively researched [102, 221, 40, 154, 186]. The UPSNet [221] is the most relevant method to ours, which unifies the segmentation from both of the semantic segmentation branch and the instance segmentation branch with the panoptic head.

Relation networks. Relations among objects offer context information for object recognition. [85] applies relation networks for the image object detection. Relation networks are further extended into spatial-temporal domain for video object detection [53] and MOT task [204]. Our network also utilizes the relation networks to acquire better features for object association in tracking.

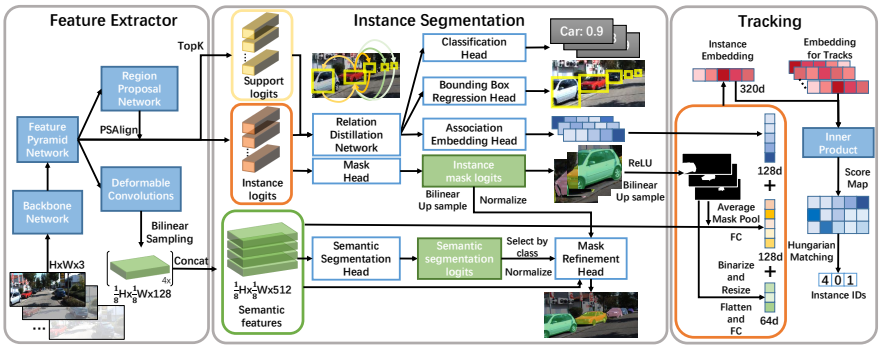

Figure 6.2 Architecture overview. Our UMotsNet is mainly comprised of three parts, i.e., feature extractor, instance segmentation and tracking. The bottom-up semantic segmentation and the top-down instance segmentation jointly refine the final segmentation results. The top-down instance embedding, the bottom-up semantic embedding and the mask embedding are jointly used for instance association.

6.3 Method

In this section, we present the proposed method UMotsNet in detail. The MOTS task performance is mainly affected by two major factors. The first is the performance of instance segmentation and the second is the performance of object tracking. Our UMotsNet is built upon the Mask R-CNN [76] by considering the above two aspects.

6.3.1 Architecture Overview

The architecture of our UMotsNet is illustrated in Fig. 6.2. The whole pipeline can be divided into three parts, i.e., feature extractor, instance segmentation, and tracking. We first illustrate the architecture design overview, and then the details of each major component.

Feature Extractor. For a video $\{I_t|t = 1...T\}$ with T frames, the feature extractor F extracts features for each frame I_t. The feature extractor F is comprised of the backbone feature extractor F_b, the instance feature extractor F_{inst} and the semantic feature extractor F_{sem}. The backbone feature extractor F_b consists of ResNet101 [77] and feature pyramid networks [120] (FPN). The extracted backbone features f_b from FPN are adopted by the instance feature extractor F_{inst} and the semantic feature extractorF_{sem}. Instance feature extractor F_{inst} first proposes object candidates $\{C_i|i = 1...N_c\}$ with region proposal networks [159]. Each candidate C_i consists of a bounding box C_i^{box} and an objectness score C_i^{score}. The position sensitive roialign [48, 76] operator is then adopted to pool the features f_{C_i} from f_b for each candidate C_i. Similar to UPSNet [221], we acquire bottom-up semantic features f_{sem} with the semantic feature extractor F_{sem}. F_{sem} transforms the backbone features f_b into f_{sem} with consecutive operations, i.e., deformable convolutions [49], bilinear sampling (to $\frac{1}{8}$ image height and width) and concatenation in the channel dimension.

Instance segmentation. Instance segmentation is the unity of object detection and segmentation. Bounding boxes $\{f_i^{box}|i = 1...N_c\}$, class scores $\{f_i^{cls}|i = 1...N_c\}$ and instance masks $\{f_i^{mask}|i = 1...N_c\}$ for the object candidates $\{C_i|i = 1...N_c\}$ are inferred from the instance logits $\{f_{C_i}|i = 1...N_c\}$ by the bounding box regression head H_{box}, the classification head H_{cls} and the mask head H_{mask} respectively. We additionally extract instance association embedding $\{emb_i^{top}|i = 1...N_c\}$ with the association embedding head H_{emb} for the tracking part. In the bottom-up branch, the semantic category (car, pedestrian and background) for each pixel is inferred by the semantic segmentation head H_{sem}. The semantic segmentation logits $logits^{sem}$ from the semantic segmentation head H_{sem} will be used jointly with the instance mask logits $logits^{inst}$ and the semantic features f_{sem}, which are acquired from the mask head H_{mask} and the feature extractor F respectively, to refine the final instance segmentation in the mask refinement head H_{mrf}.

The objects in the scene are often occluded, as shown in Fig. 6.3. The car in the green box is occluded by the car in the orange box, resulting in poor appearance condition. However, the cars in the front and nearby provide cues to identify the occluded car. We therefore exploit the relation networks [85] to augment the instance logits f_{C_i} before they are sent to different heads. Considering the stability of the networks, we augment the pooled instance features f_{C_i} in the relation distillation networks [53] with a fixed number of relations for each object.

We select top N_p candidates from the object candidates $\{C_i|i = 1...N_c\}$ as the support candidates $\{C_i^{sup}|i = 1...N_p\}$ based on their objectness scores C_i^{score}. The corresponding pooled logits f_{C_i} form the support logits $f_{C_i}^{sup}$, which are used to augment the instance logits f_{C_i}. The intuition behind the candidates selection with high objectness scores is that those candidates are normally in better appearance

Figure 6.3 An example of object occlusion in the scene. Car in the green box has a large appearance change between two different time, which is more difficult to re-identify. The foreground object in the orange box has a relatively stable appearance, which is easier to re-identify. The relations between the two objects can be exploited for better recognition.

Figure 6.4 Illustration of the segmentation error caused by inaccurate bounding boxes in UPSNet [221]. Left image shows an example of segmentation of a car. Mask outside the bounding box is missing (FN). Missing parts are marked with red circles. Right image shows the refined mask output.

quality, which are more distinguishable references. N_p is determined through the ablation study in Sec. 6.4.5. The augmentation process can be expressed as follows,

$$f_{C_i}^{aug} = F_{rdn}(f_{C_i}, \{f_{C_j}^{sup} | j = 1...N_p\}) \tag{6.1}$$

Tracking. We follow the tracking-by-detection paradigm. The association embedding emb_i^{inst} of each detected object C_i is extracted from both of the top-down branch and the bottom-up branch as shown in Fig. 6.2. We jointly utilize the top-down association embedding emb_i^{top} (blue color) and the bottom-up association embedding emb_i^{bot} (orange color), which characterize the appearance of the instances. We further utilize position and shape information by transforming the binary mask of each instance $mask_i^b$ into mask embedding emb_i^{mask}. We associate the objects through the score map, calculated by the inner product of the association embedding between the instances emb_i^{inst} and the tracks emb_j^{track} [208], and a larger value implies a higher probability for matching. Zero values are added to the score map for no-match cases [208]. As there is chance that multiple objects are

matched to one track or that multiple tracks are matched to one object, the hungarian algorithm [105] is used to resolve the matching confliction.

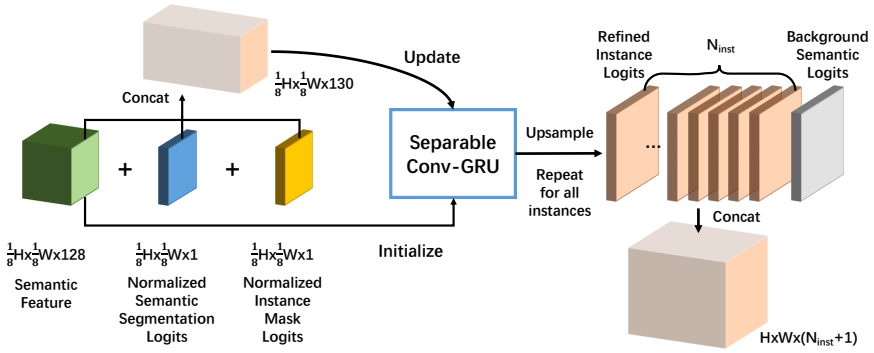

Figure 6.5 Illustrator of the mask refinement head. The mask refinement head utilizes the semantic features, the normalized semantic segmentation logits and the normalized instance mask logits.

6.3.2 Mask Refinement

The top-down instance segmentation and the bottom-up semantic segmentation are mutually complementary. We observe that instance segmentation has better quality for small objects, while semantic segmentation is more accurate for large objects. The main reason is that the top-down instance segmentation has a fixed sampling grid (28×28 in our case), which limits the spatial resolution for large objects. Semantic segmentation has higher spatial resolution for large objects, but the receptive field is too large for small objects, causing lower accuracy. Combining the cues from both of the instance segmentation and the semantic segmentation is beneficial. We acquire the instance mask logits $logits_i^{inst}$ and the semantic segementation logits $logits_i^{sem}$ from the mask head H_{mask} and the sementic segmentation head H_{sem} respectively based on the instance class prediction f_i^{cls} as in [221]. UPSNet [221] directly fuses the instance mask logits $logits_i^{inst}$ and the semantic segmentation logits $logits_i^{sem}$ through addition, but this limits the accuracy if either branch is not accurate. Another source of inaccuracy is caused by the inaccurate bounding box prediction f_i^{box}. Since logits are cropped by the bounding boxes, the instance mask and the semantic segmentation mask outside the bounding box will be missed, resulting in false negative (FN) prediction as shown in Fig. 6.4. We instead adopt $logits_i^{sem}$ and $logits_i^{inst}$ as cues for the mask refinement as shown in Fig. 6.5.

A separable Conv-GRU (SCG) module is adopted for the mask refinement process, which has also been used for optical flow estimation [176]. The inputs for the SCG module are the semantic features f_{sem}, the normalized semantic segmentation logits $logits_i^{sem_{norm}}$ and the normalized instance mask logits $logits_i^{inst_{norm}}$. Semantic features f_{sem} are first reduced to 128 channels with 1×1 convolution, and then concatenated to $logits_i^{sem_{norm}}$ and $logits_i^{inst_{norm}}$ forming the input features $f_i^{mrf_{in}}$ for the mask refinement head H_{mrf}. $logits_i^{sem_{norm}}$ and $logits_i^{inst_{norm}}$

are acquired through the following normalization,

$$\begin{aligned} logits_i^{inst_{norm}} &= Sigmoid(logits_i^{inst}) \times 2 - 1 \\ logits_i^{sem_{norm}} &= Softmax(logits_i^{sem}) \end{aligned} \tag{6.2}$$

$logits^{inst}$ are normalized to $[-1, 1]$, and pixels with value above 0 are foreground. Pixels outside the bounding box are assigned -1. $logits^{sem}$ are normalized to $[0, 1]$ with softmax operator, which is the class probability for the pixels. Pixels outside the bounding box are assigned 0. We initialize the SCG module with the semantic feature maps f_{sem} and zero hidden states as input. The refinement process can be expressed as,

$$mask_i^{refine} = H_{mrf}(f_i^{mrf_{in}}) \tag{6.3}$$

The effectiveness of such design is shown in the ablation study (Sec. 6.4.5).

Multi-instances refinement. The mask refinement module process multiple instances individually and in parallel. Instances having overlapped bounding boxes share features within the overlapped area. Pixel assignment in the overlapped area can be resolved with joint learning [221]. We concatenate the refined instance logits $\{mask_i^{refine} | i = 1...N_c\}$ in the channel dimension with an additional background semantic logits $logits_{bg}^{sem}$ from the semantic segmentation. The instance identity for each pixel can be determined with argmax operation along the channel dimension.

6.3.3 Relation Distillation

Relations among objects provide useful cues for object recognition, and we adopt relation distillation networks [53] for the instance feature augmentation. The detailed structure is shown in Fig. 6.6. The support logits $f_{C_i}^{sup}$ and the instance logits f_{C_i} are the two inputs for the relation distillation networks F_{rdn}. The cascades of two relation modules [85] interleaved with the fully connected layers form the relation distillation networks F_{rdn}, which effectively augment the instance logits f_{C_i} with the support logits $f_{C_i}^{sup}$ according to their relations that are guided by the relative positions and the instance features.

6.3.4 Embedding for Association

The design of the association embedding also takes advantage of the bottom-up cues and the top-down cues. The top-down cues come from the association embedding head H_{emb}, which provides instance level embedding emb_i^{top}, while the bottom-up cues emb_i^{bot} come from the pooled semantic features guided by the instance segmentation. The association embedding head H_{emb} is comprised of 2 fully connected layers, which outputs top-down embedding emb_i^{top} of 128 dimensions. For the bottom-up embedding emb_i^{bot}, we use the average of the semantic features f_{sem} weighted by the positive instance mask logits, which can be acquired with ReLU operator as $ReLU(logits_i^{inst})$. The pooled features are converted to 128 dimensions by another fully connected layer. In order to incorporate the global position and the shape of the instance, the binary mask $mask_i^b$ of each

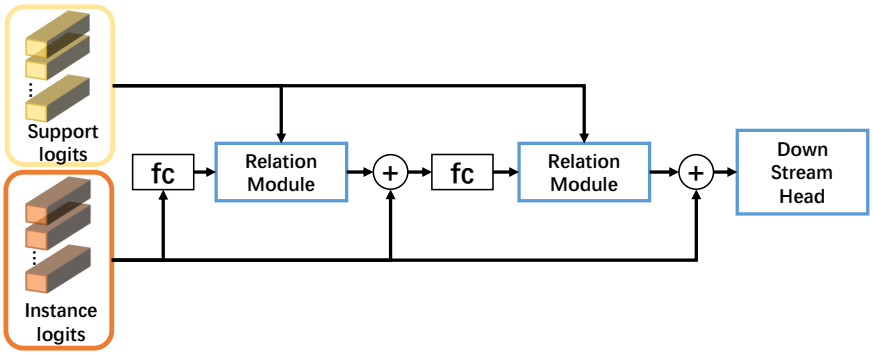

Figure 6.6 Illustrator of the relation distillation networks. The instance logits are recursively refined through two relation modules, and the augmented instance logits are used for down stream tasks.

instance is encoded into mask embedding. We binarize the instance mask logits $logits_i^{inst}$ into the hard mask $mask_i^b$ with a threshold of 0, which separates the foreground from the background area. As the whole binary mask has a large spatial dimensions, we resize the mask into a square shape of 32×32 by the adaptive average pooling operator. The resized binary mask is flattened and converted to an embedding emb_i^{mask} of 64 dimensions with a fully connected layer. The above top-down embedding emb_i^{top}, bottom-up embedding emb_i^{bot} and mask embedding emb_i^{mask} are concatenated to form the final instance embedding emb_i^{inst} for tracking.

6.3.5 Tracking Pipeline

Our model runs instance segmentation and tracking in an online fashion. New tracks will be created if the detected objects have no matched tracks, otherwise the embedding of the matched tracks are updated. The embedding of the tracks emb_i^{track} are updated to the corresponding latest instance embedding emb_i^{inst}. If an active track is not updated for α frames, it is ended and removed from the matching process. We set a similarity threshold β to the score map, and only the scores higher than β can be considered as potential valid matches. In the experiments, α and β are set to 10 and -10, respectively.

6.3.6 Losses

Our model is trained end-to-end with multi-task losses, including losses for detectionL_{det}, segmentationL_{segm} and trackingL_{track}, as follows,

$$L = L_{det} + L_{segm} + L_{track} \tag{6.4}$$

Detection losses. L_{det} includes losses from the RPN and the R-CNN, which is the same as the Mask R-CNN [76], shown in the following,

$$L_{det} = L_{rpn_{box}} + L_{rpn_{cls}} + L_{rcnn_{box}} + L_{rcnn_{cls}} \tag{6.5}$$

Segmentation losses. L_{segm} contains multiple losses, including top-down instance segmentation loss $L_{mask_{inst}}$, bottom-up semantic segmentation loss $L_{mask_{sem}}$ and

unified mask refinement loss $L_{mask_{refine}}$ (cross entropy loss) as follows,

$$L_{segm} = L_{mask_{inst}} + L_{mask_{sem}} + L_{mask_{refine}} \tag{6.6}$$

Online hard example mining [171] is applied for semantic segmentation and mask refinement during training.

Tracking loss. L_{track} is the categorical cross entropy loss on the score map with object IDs as the target [208].

6.4 Experiments

Our experiments have two parts. Firstly, our UMotsNet is evaluated on the KITTI MOTS dataset [182]. Secondly, we conduct thorough ablation study for the model.

Dataset. KITTI MOTS dataset [182] is designed for the urban driving scenes, where there are complex street scenes and heavy occlusions. There are two object categories for evaluation, i.e., the cars and the pedestrians. KITTI MOTS dataset consists of $8,008$ frames in 21 sequences, 12 for training and 9 for validation. Another 29 sequences consisting of $11,095$ images are collected as the test set for the MOTS benchmark.

Metric. The metrics for evaluation are adapted from the well-established CLEAR MOT metrics, which are used for multi-object tracking [11]. By additionally considering the segmentation, the mask version metrics are introduced [182], including **sMOTSA**, **MOTSA**, **MOTSP** and **IDS**, which form the mostly used metrics for performance comparisons.

6.4.1 Implementation Details

Association Embedding Head. The top-down association embedding head consists of 2 fully connected layers with 256 channels. The number of channels of the output is 128.

Relation Modules. The settings of the attention module [180] in the relation modules, which are used by the relation distillation networks, are as follows. Number of channels for the keys and queries are set to 256 and 1024 respectively. The number of attention group is set to 16.

Average Feature Pooling within Mask Area. The average feature pooling for the bottom-up tracking embedding can be very easily acquired with inner product. We first flatten the last 2 dimensions of f_{sem} of size $c \times h \times w$ into $f_{sem}^{c \times hw}$. We also flatten the last 2 dimensions of the positive instance mask logits (instance mask logits after ReLU operator) of all instances into $f_{inst}^{n \times hw}$, where n is the number of instances. We normalize the logits with element-wise division (with broadcasting [146]) by the number of foreground pixels $f_{norm}^{n \times hw} = f_{inst}^{n \times hw} / N_{fg}^{n \times 1}$. $N_{fg}^{n \times 1}$ is the number of foreground pixels for the n instances. The average pooling within mask area can be calculated with matrix multiplication as $f_{sem}^{ave} = f_{norm}^{n \times hw} \times (f_{sem}^{c \times hw})^T$, where f_{sem}^{ave} is of shape $n \times c$.

Region Proposal Networks (RPN). The anchors for the RPN have 4 scales and 3 aspect ratios $(1:2, 1:1, 2:1)$ as in [221]. The RPN outputs $2,000$ rois during training and $1,000$ rois during testing.

Position Sensitive ROI Pooling (PSROIPooling). The PSROIPooling requires a different number of input feature channels compared with Mask R-CNN [76]. The features from FPN (128 channels) are first converted to 490 channels before the pooling operation. The pooling size for the detection branch and the mask branch are 7×7 and 14×14 respectively and both with groups of 7.
Non-maximum Suppression (NMS). The IoU threshold for bounding box non-maximum suppression is 0.7 and 0.6 for the RPN proposals and the detection boxes respectively.
Online Hard Example Mining (OHEM). We apply OHEM [171] for the semantic segmentation loss and the mask refinement loss. We select pixels set $P_{sel} = \{p_i | i = 1, ..., N_{sel}\}$ with the smallest $\frac{1}{16}$ losses of all pixels. If the maximum loss in P_{sel} is below a threshold γ (0.7 in our case), only pixels in P_{sel} are used for training. Otherwise, standard cross entropy loss is applied.

6.4.2 Experimental Setup

We implement the model with PyTorch. The training and testing are performed with 1 Titan X(Pascal) GPU only.
Training. Following previous work [182, 152, 206, 119, 84], we apply a pre-training step first as the number of training images in the KITTI MOTS dataset is limited. Inspired by [182, 206, 207, 152, 119, 130], we firstly pre-train our model with instance segmentation datasets that focus on driving scenes, including Cityscapes [47], Mapillary Vistas [140] and KINS [153] datasets. The detection, segmentation and tracking parts are jointly trained. For the pre-training stage, the focus is more on improving the detection results, we therefore set a loss weight of 0.1 to the segmentation loss L_{segm} and the tracking loss L_{track} except the detection loss L_{det}. SGD [163] optimizer is adopted with a learning rate of $1e^{-3}$ for $240K$ iterations. For the video dataset training, pairs of images with a random frame gap ranging from 1 to 10 are used for training. The model is trained with a learning rate of $1e^{-4}$ for $40K$ iterations and reduced to $1e^{-5}$ for another $10K$ iterations. Random scaling is utilized for data augmentation, with the image short side ranging from 416 to 608.

As there are many objects in the scene, it is not always possible to apply mask refinement for all instances during training due to the limited GPU memory (12GB in our case). We therefore randomly select 16 instances for joint refinement. The pixels within the unselected instances are ignored for training.
Testing. For testing, we keep track of all the objects in the scene. Only the detected objects with scores higher than 0.7 are considered as valid detection for the instance segmentation and tracking.

6.4.3 Results

In this part, we compare the performance of our model with other methods. The results of different methods on the KITTI MOTS dataset are listed in Table 6.1. Our model utilizes 2D data only, and methods with 3D modality are also included for reference. Our method outperforms all other 2D methods except the PointTrack [206] regarding the sMOTSA metric. PointTrack takes advantage of the state-of-the-art bottom-up instance segmentation methods [141], which is more suitable for scenes

with heavy occlusions. Compared with PointTrack without optical flow support, there is a smaller gap between our methods. Among all the anchor based object detection methods, i.e., TRCNN [182], MOTSNet [152] and VAE Mask R-CNN [119], our UMotsNet performs the best. Methods with 3D modality have stronger performance on average. The BePix [168] and MOTSFusion [130] adopt heavy network design by using RRC [158] for detection and BB2SegNet [131] for segmentation. The overall model speed for these 3D methods are much slower compared to the 2D methods.

Table 6.1 Results on the KITTI MOTS validation set. Both 2D methods and 3D methods are listed for comparisons with sMOTSA, MOTSA and IDS metrics. ↑ stands for the higher the better. ↓ stands for the lower the better. w. stands for with and w/o stands for without.

Type	Method	Base Detector	Speed	Cars			Pedestrians		
				sMOTSA(%) ↑	MOTSA(%) ↑	IDS ↓	sMOTSA(%) ↑	MOTSA(%) ↑	IDS ↓
2D	TRCNN [182]	TRCNN [182]	0.5	76.2	87.8	93	46.8	65.1	78
2D	MOTSNet [152]	MOTSNet [152]	-	78.1	87.2	-	54.6	69.3	-
2D	VAE Mask R-CNN [119]	VAE Mask R-CNN [119]	-	77.6	88.8	-	49.7	67.6	-
2D	STEm-Seg [6]	STEm-Seg [6]	-	76.2	87.8	93	48.9	64.8	15
2D	PointTrack w. optical flow [206]	SpatialEmbedding [141]	0.045	85.5	94.9	22	62.4	77.3	19
2D	PointTrack w/o optical flow [206]	SpatialEmbedding [141]	0.045	82.9	92.7	25	61.4	76.8	21
3D	BePix [168]	RRC [158]+TRCNN [182]	3.96	76.9	89.7	88	-	-	-
3D	BePix [168]	RRC [158]+BB2SegNet [131]	3.96	84.9	93.8	97	-	-	-
3D	MOTSFusion [130]	TRCNN [182]+BB2SegNet [131]	0.84	82.6	90.2	51	58.9	71.9	36
3D	MOTSFusion [130]	RRC [158]+BB2SegNet [131]	4.04	85.5	94.9	22	62.4	77.3	19
2D	UMotsNet(Ours)	UMotsNet	0.14	81.8	93.2	77	55.7	74.3	46

We also report our results on the KITTI MOTS test set, the results are shown in Table 6.2. It should be noted that the performance gap between our method and the state-of-the-art methods is narrowed. Our UMotsNet is comparable to MOTSFusion [130], behind PointTrack [206] and significantly better than MOTSNet [152] and TRCNN [182]. The comparison is fairer due to the very limited number of times for test set evaluation.

Table 6.2 Results on the KITTI MOTS test set. ↑ means the higher the better.

Method	Cars		Pedestrians	
	sMOTSA(%) ↑	MOTSA(%) ↑	sMOTSA(%) ↑	MOTSA(%) ↑
TRCNN [182]	67.0	79.6	47.3	66.1
MOTSNet [152]	71.0	81.7	48.7	62.0
MOTSFusion [130]	75.0	84.1	58.7	72.9
PointTrack [206]	78.5	90.9	61.5	76.5
UMotsNet(Ours)	76.5	88.5	57.1	75.7

We also compare the segmentation quality with STEm-Seg [6] and VAE Mask R-CNN [119], which have strong segmentation design. MOTSP metric [182] is reported in Table 6.3 for comparisons. Our method performs the best for car and second best for pedestrians. On average, our model outperforms STEm-Seg [6] and VAE Mask R-CNN [119].

6.4.4 Qualitative Results

More qualitative examples for cars and pedestrians of the KITTI MOTS benchmark are shown in Fig. 6.7, Fig. 6.8, Fig. 6.9, and Fig. 6.10.

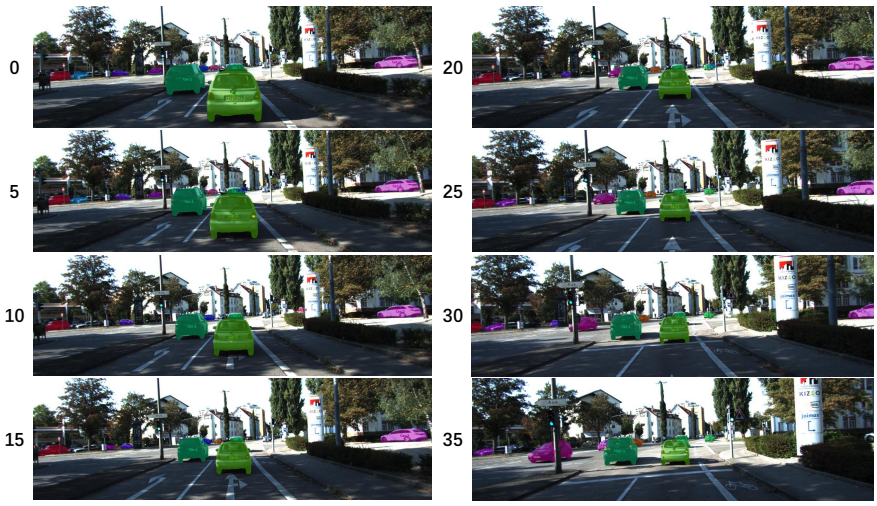

Figure 6.7 Qualitative example of KITTI MOTS test sequence 0012. Frame index is shown on the left side of each image.

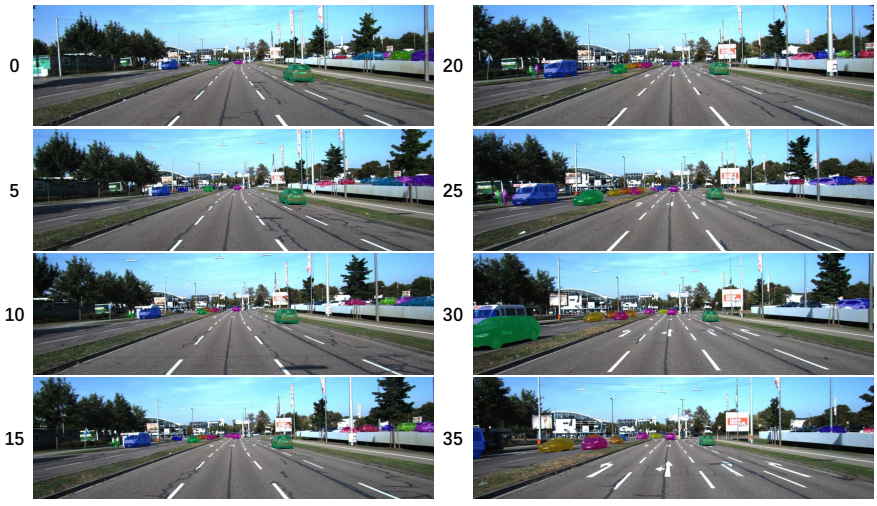

Figure 6.8 Qualitative example of KITTI MOTS test sequence 0013. Frame index is shown on the left side of each image.

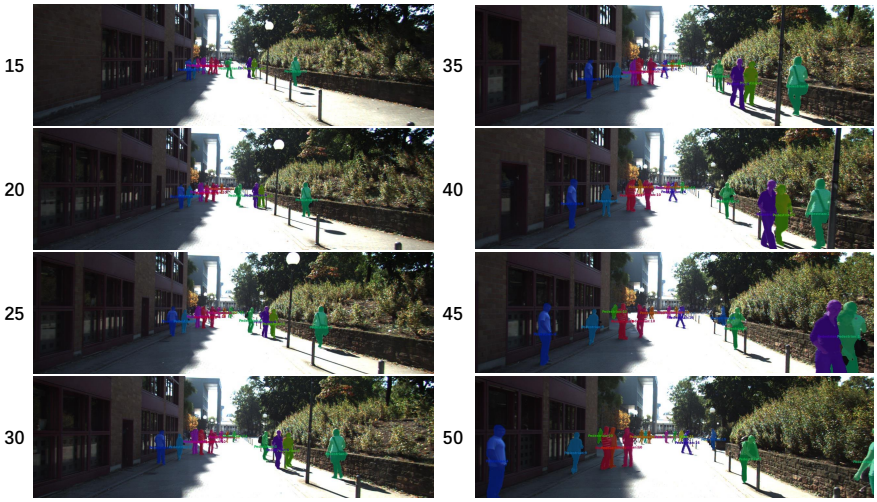

Figure 6.9 Qualitative example of KITTI MOTS test sequence 0026. Frame index is shown on the left side of each image.

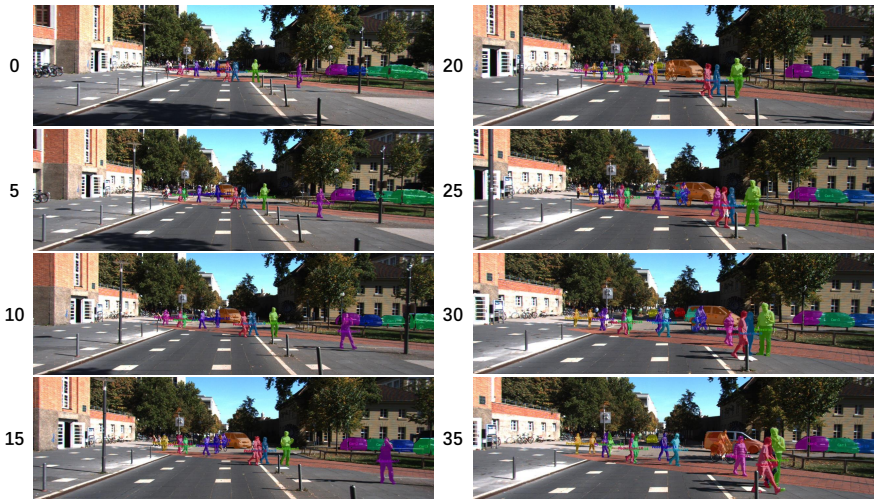

Figure 6.10 Qualitative example of KITTI MOTS test sequence 0028. Frame index is shown on the left side of each image.

Table 6.3 Mask Accuracy Comparisons on the KITTI MOTS validation set. ↑ means the higher the better.

Method	MOTSP(car)(%) ↑	MOTSP(ped.)(%) ↑
TRCNN [182]	87.2	75.7
STEm-Seg [6]	87.2	77.7
VAE Mask R-CNN [119]	87.7	77.0
UMotsNet(ours)	88.0	77.2

6.4.5 Ablation Study

In the ablation study, we investigate the usefulness of the bottom-up and the top-down cues for the mask refinement and the tracking, as well as the best number of support proposals used for relation distillation networks.

Different Cues for Mask Refinement Module Our mask refinement module takes advantage of different cues, which include bottom-up semantic segmentation logits and top-down instance mask logits. We investigate the usefulness of different cues by removing each individual one. The results for different settings are shown in Table 6.4. The first column shows the performance of the full model, which performs the best in all metrics. By removing the top-down logits, the sMOTSA score for car drops by 4.3 percent, which is much larger than the 0.7 percent by removing the bottom-up logits. This makes sense as top-down logits help to distinguish the overlapping objects. The 77.5% sMOTSA score is still higher than TRCNN [182]. Removing the bottom-up cues improve the sMOTSA score for pedestrian. It is caused by the fact that the pedestrians are often small, the bottom-up logits have large receptive field causing slight drop in accuracy. On average, the bottom-up logits are beneficial to the mask refinement.

Qualitative examples are shown in Fig. 6.11. It could be noted that the bottom-up cues and the top-down cues are mutually complementary, and joining both cues give the best performance. The top-down instance mask logits help to differentiate different objects in the overlapped bounding box area, which is marked by the red arrow in the first row, where part of the car is assigned wrong identity. The bottom-up logits contribute to a more complete instance segmentation. As shown in the bottom left image, the pedestrian is divided into two objects, which is resulted by the incomplete instance segmentation from the top-down branch. The bottom-up semantic segmentation complete the segmentation as shown in the bottom right image.

Different Cues for ReID In order to show the effectiveness of different cues for the object re-identification in tracking, we remove each individual component and compare models of different settings with the full model. The results are shown in the Table 6.5. The full model gives the best performance in all metrics. Removing the top-down embedding causes significant drop for the sMOTSA metric, which shows that the top-down cue is important for object re-identification. Removing the bottom-up embedding or the mask embedding would both cause smaller performance drop, which shows that the bottom-up cues and the shape and location cues are complementary to the top-down cues for the object re-identification.

The qualitative examples are shown in Fig. 6.12. It could be noted that the full model gives the best performance, and the top-down embedding is the most critical

Table 6.4 Ablation study on cues for mask refinement.

Top-down logits	✓	-	✓
Bottom-up logits	✓	✓	-
sMOTSA(%)(car)	81.8	77.5	81.1
MOTSA(%)(car)	93.2	89.2	92.7
MOTSP(%)(car)	88.0	87.7	87.9
IDS(car)	77	327	92
sMOTSA(%)(ped.)	55.7	42.1	56.2
MOTSA(%)(ped.)	74.3	60.7	74.8
MOTSP(%)(ped.)	77.2	76.8	77.0
IDS(ped.)	46	513	52

one for the object re-identification.

Relation Distillation Network The top-down instance logits are augmented by considering the relations among the detected objects. We experiment with different numbers of object proposals to determine the best number (N_p) of support logits.

Figure 6.11 Qualitative examples of removing different cues for the mask refinement module. The two images on the right column show the predictions of the full model. The top left image shows the prediction without top-down cues (instance mask logits). The bottom left image shows the prediction without bottom-up cues (semantic segmentation logits). The red arrows mark the places for comparisons.

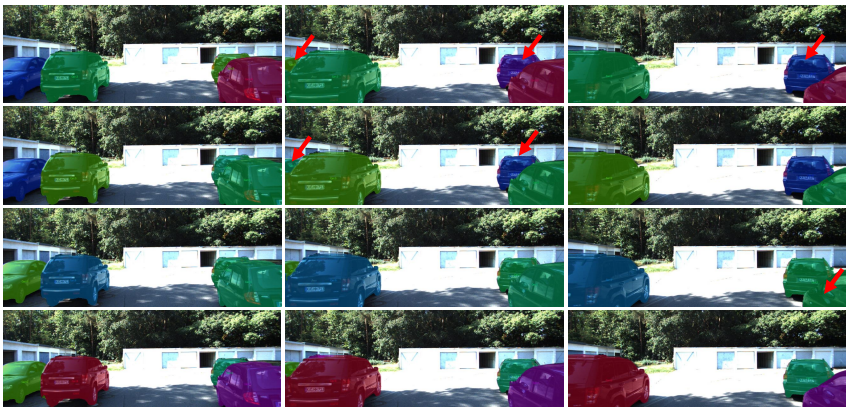

Figure 6.12 Qualitative examples of removing different cues for the association embedding. Consecutive images are shown horizontally. Each row corresponds to one model setting. The first row shows the prediction without top-down embedding. The second row shows the prediction without bottom-up embedding. The third row shows the prediction without mask embedding. The last row shows the prediction of the full model. The identity switches are marked with red arrows.

Table 6.5 Ablation study on cues for object ReID.

Top-down embedding	✓	-	✓	✓
Bottom-up embedding	✓	✓	-	✓
Mask feat.	✓	✓	✓	-
sMOTSA(%)(car)	81.8	74.3	81.4	81.4
MOTSA(%)(car)	93.2	85.8	92.8	92.8
IDS(car)	77	614	85	90
sMOTSA(%)(ped.)	55.7	32.8	54.5	54.9
MOTSA(%)(ped.)	74.3	51.7	72.9	73.5
IDS(ped.)	46	830	48	61

We also compare with the model without the relation support. The performance of different model settings are listed in Table 6.6. We notice that an improper number of object relations would not be beneficial to all classes. Without the relation support, the sMOTSA scores for cars and pedestrians are 81.1 and 55.4 respectively. With the relation support, the performances for car and pedestrian reach peaks with different N_p, 75 and 125 respectively. Too many or too few support objects will result in incomplete or over-complete relations causing performance drop. 125 proposals provide a good performance balance for the two classes, and the score is better than model without relation support.

Table 6.6 Ablation study on number of proposals for relation distillation network. - means without relation support.

No. Prop.(N_p)	-	25	50	75	100	125	150
sMOTSA (%)(car)	81.1	81.4	81.2	82.1	81.4	81.8	81.0
MOTSA (%)(car)	93.0	93.0	92.8	93.8	93.0	93.2	92.4
sMOTSA (%)(ped.)	55.4	54.0	54.4	54.6	55.3	55.7	54.7
MOTSA (%)(ped.)	73.4	72.4	73.1	73.0	73.8	74.3	73.5

6.5 Conclusion

We have introduced the UMotsNet, which combines the top-down and bottom-up features for joint object detection, segmentation and tracking. By incorporating the novel mask refinement module and the relation guided joint association embedding design, the performance is significantly boosted. The experiments are conducted on the KITTI MOTS dataset and the results have shown the effectiveness of our model for the multi-object tracking and segmentation task. In the future work, we would boost the system with better detector as the anchor based detection has limited the performance for the MOTS task.

Synthesis

7

In this synthesis chapter, the conclusions, reflections and outlook are discussed by reviewing the previous chapters and contemporaneous or more recent related publications.

7.1 Conclusions per Objective

Semantic segmentation of UAV images.
This objective is addressed in chapter 2 and chapter 3. We have created a new semantic segmentation benchmark for oblique view UAV imagery. We have also brought out the multi-scale dilation net and bidirectional multi-scale attention net to better handle the multi-scale problem introduced by the UAVid dataset. The multi-scale dilation net circumvents the need for an image pyramid by applying multiple dilated convolutions, which allows intermediate feature fusion instead of class probability fusion. The bidirectional multi-scale attention net achieves better performance than hierarchical multi-scale attention net [175] by inferring features and weights from both higher level and lower level branches.

The multi-scale dilation net transforms the spatial dimensions to the batch dimension or vice versa. A similar strategy has been used to transform the spatial dimensions to the channel dimension [187] for upsampling, or the channel dimension to the spatial dimension for positional sensitive pooling [48] for object recognition. Such similar information routing could be used for different purposes.

The multi-scale attention design has significantly improved the recognition of small objects, such as humans. But it is still far from perfect as the mIoU score for human class is just over 30%, which means that small object recognition still remains a challenge. For differentiating static cars and moving cars, we have mostly leveraged on the context of objects. More accurate recognition could be achieved by using sequence input.

Video object segmentation.
This objective is addressed in chapter 4. We have proposed a model that combines the Mask R-CNN [76] and the Conv-GRU [8] module for the video object segmentation task. Our method learns to propagate the target object labels without auxiliary data, such as optical flow, which simplifies the model. By utilizing instance level segmentation, our model can restrict the mask prediction to the range of an object, which is better than semantic segmentation based methods that easily propagate the masks to other objects.

In order to speed up the overall performance, online learning is not an effective solution. Instead, better design choices could be used, such as pixel matching [181, 213, 142] and meta learning [16]. However, current state-of-the-art methods still run at low speed. E.g., 6 FPS for LWL [16], 6 FPS for STM [142], and 5 FPS for [213].

Video object detection.

This objective is addressed in chapter 5. We have proposed a model that unifies the object detection networks and the object tracking networks by sharing their feature extraction component. The proposed siamese convolutional regression tracker is a very light-weighted increment to the object detection network, but effective to track objects when their appearances deteriorate. Its class-agnostic property and the design to re-use the features from the object detector enable it to handle the problem of missing video labels for some classes. The limitation of the current design is that the proposed model would not handle long-term tracking. If an object is occluded for a long time, it may be out of the target patch and fail to be tracked. The design could be improved with methods for the task of multi-object tracking [68, 137].

Contemporary work SiamFC++ [205] has designed a similar convolution plus regression tracking model to boost the tracking performance for the task of single object tracking. The difference is that our model re-uses features from an object detector. Another design choice for object tracking in siamese style is introduced by Siam-RCNN [184] for video object segmentation task, which uses a similar strategy. Instead of using convolution between pooled template and target patches, Siam-RCNN concatenates them as input features for regression branches.

Multi-object tracking and segmentation.

This objective is addressed in chapter 6. In this objective, object detection, segmentation, and tracking are unified. The whole pipeline follows the multi-task design, i.e., a single feature extraction backbone with multiple heads for different sub-tasks. In order to improve the performance for object detection, segmentation, and tracking, the two branches design, i.e., bottom-up branch and top-down branch, is utilized. By combining the bottom-up and the top-down inference for all sub-tasks, the overall performance improves. The task of multi-object tracking and segmentation is just introduced in [182], there is still a lot of room for performance improvement in tracking. From the most recent research, multi-modality seems to be the best solution for robust tracking. ViP-DeepLab [155] jointly uses images and point clouds to achieve the state-of-the-art result. EagerMot [99] has also applied 3D information to achieve good results. Our future work could also leverage on multi-modality to improve the overall performance.

7.2 Reflections and Outlook

In this section, reflections and outlook regarding the related tasks in this thesis will be discussed.

7.2.1 Range of Object Context Matters

In chapter 2 and chapter 3, we have designed novel deep neural networks to handle object recognition of different scales. Objects of different scales are better recog-

nized in different ranges of context. The context provides important information for object recognition. Such contextual information is even critical in certain circumstances, for example, when the distant object is too small, the recognition would rely heavily on the context. Taking the oblique view UAV images as an example, where there is large scale variance for different objects, recognition of distant cars greatly relies on the road context. This makes sense as the objects lack sufficient details for recognition. From the semantic segmentation experiments with bidirectional multi-scale attention in the model, it could be noted that an adaptive context scale selection for different objects is preferred. With respect to the deep neural networks, the object context relates to the receptive field of the deep neural networks. Larger receptive fields result in larger context ranges. Besides the solutions used in chapter 2 and chapter 3, there are additional methods. Deformable convolution [49, 230] could also adaptively adjust the receptive field for different objects to a certain degree, but the extra computation makes it relatively slow. Self-attention [180] has been more popular recently, as it could enlarge the receptive field of the deep neural networks to the entire image [66]. However, such design undermines the performance if a large amount of unrelated context is included, or a very long training time is required for the deep neural networks to learn to reject unnecessary context noise. In order to achieve proper scale of context, self-attention with a full image receptive field may not be a good option if the objects are small compared with the size of the input images.

7.2.2 Quality of Video Data Matters

Chapter 4, chapter 5, and chapter 6 all deal with tasks related to videos, but the video qualities are quite different resulting in different research problems. For real-world applications, the quality of videos would affect the model design. Another example is as follows, video instance segmentation (VIS) and multi-object tracking and segmentation (MOTS) both require object detection, segmentation and tracking. From this perspective, there is not much difference. However, the quality of the datasets designed for these two tasks have different features as shown in Table 7.1. Compared with MOTS datasets (e.g., KITTI MOTS dataset), VIS datasets (e.g., Youtube VIS dataset) have more class types, making it more difficult for object recognition. But there are also much fewer objects in one image, making tracking easier than it is in MOTS datasets. The challenge for the VIS dataset is that the objects are often in rare pose and heavy occlusion, resulting in difficulty for correct detection. The MOTS datasets have more emphasis on the object re-identification. Object detection is relatively easy as there are normally only one or two classes, but there are many more objects in one image. The challenge is to differentiate objects of the same class type that are similar in appearance, and to preserve long-term tracking after occlusion. The Quality of the dataset determines how the task should be solved. The best approach for the VIS task is to combine instance segmentation with mask propagation, which focuses on more accurate detection and more complete mask segmentation, while the best practice for the MOTS task is in the tracking-by-detection paradigm, where detection serves as a preliminary, and more research tries to improve the tracking part only.

Table 7.1 Comparisons between datasets designed for VIS task and MOTS task.

Dataset type	VIS dataset	MOTS dataset
Number of objects in a scene	Small	Large
Number of categories in the dataset	Many	Few
Object detection difficulty	High	Low
Object tracking difficulty	Low	High

7.2.3 Relation Between Detection and Tracking

As discussed in chapter 5, there is quite a close relation between detection and tracking. These two tasks could improve each other in certain circumstances. Tracking has been used in video object detection, and detection has also been used for tracking purposes. For example, for the multi-object tracking task, the tracking-by-detection paradigm is still the dominant strategy for multi-object tracking. The tracking is based on the detection results, where the detected objects are linked based on similarity comparisons. For single object tracking, the detection paradigm has also been used for tracking purposes. Several works [184, 185, 89] have achieved competitive performance for multiple tracking benchmarks, showing that the detection paradigm is also effective for tracking. Tracking has also been useful for object recognition in videos as illustrated in chapter 5. One of the weakness for the Imagenet video object detection benchmark is that its evaluation does not consider the identity consistency of the objects detected. The evaluation metric only takes the mean average precision into account for the object detection in images. For the Youtube video instance segmentation benchmark, the evaluation takes object identities into account, but compared to other multi-object tracking benchmark datasets, the cases for tracking are too simple. For the tracking datasets, e.g., MOT16, MOT17, there is no need for strong detection models compared with those models used for detection datasets. A more powerful system would require the model to detect and track objects simultaneously and robustly in different circumstances, however, exploration in such direction is still lacking due to the missing benchmark dataset that requires the model to perform competitively in both detection and tracking. Fusing the paradigm of detection and tracking is still an interesting direction, and improved feature sharing methodology for detection and tracking could be explored in the future.

7.2.4 Unified Dynamic Scene Understanding

In chapter 2 and chapter 3, we have developed models for semantic segmentation, which is bottom-up scene understanding as object recognition is based on pixel level. In chapter 4 and chapter 5, we have designed models for top-down scene understanding as objects are recognized at instance level. In chapter 6, we have designed a model that unifies bottom-up and top-down inference, which is inspired by the UPSNet [221] designed for panoptic segmentation [103]. Semantic segmentation (bottom-up) and instance segmentation (top-down) complement each other. Instance segmentation requires detection and segmentation of different objects, whilst distinguishing them even if they are of the same category. Semantic segmentation could handle uncountable objects, such as sky, road and so on, but would fail to differentiate overlapped objects of same classes. The panoptic segmentation task has

been developed to unify instance segmentation and semantic segmentation, whereby both of their advantages can be incorporated. The research in this field has been developing rapidly. The model design for panoptic segmentation has transited from the heuristic combination of semantic segmentation and instance segmentation [102], which are independently inferred from different heads and merged, to unified inference [40, 221]. The unified approach is favored as it enables the networks to resolve conflicts between outputs of the semantic segmentation branch and the instance segmentation branch.

Panoptic segmentation has also been extended from static images to dynamic videos, which is known as the video panoptic segmentation task [101]. The model needs to segment the scene, whilst being able to differentiate different objects and track them. The newly proposed VIPER dataset [101] has a great number of labeled video to support this task, and lays a good dataset foundation. It propels the development of spatial-temporal unified full scene dynamic scene understanding, which combines detection, segmentation, and tracking.

7.2.5 Is Attention All You Need for Dynamic Scene Understanding?

In chapter 5 and chapter 6, we have developed models that combine detection and tracking through multi-task learning. However, multi-task learning through multiple heads may not be the best solution as they might compete for shared features. One possible solution is to leverage on transformer [180], which has been used for processing data of multiple modalities [113, 212, 228]. The transformer is a more general computation unit in deep neural networks compared with convolutions [180]. In the design of a transformer, self-attention plays an important role in extracting semantic features. BERT [54] and GPT [21], which are built upon the transformer, have attained world striking results for a number of natural language processing (NLP) tasks, such as question answering and language inference. The huge success of the NLP has inspired the researchers to apply the modules in vision tasks such as classification, detection, segmentation and tracking. There has been great success in importing the self-attention mechanism, many new state-of-the-art results have been achieved. One of the advantages is that the self-attention mechanism enlarges the receptive field of the neural networks selectively according to the correlation of key-query pairs. The dual attention network [66] has brought in a position attention module and a channel attention module for better semantic segmentation performance. Axial deeplab [186] has obtained the fully attentional networks for the semantic segmentation task by restricting the attention to local regions. Convolutional neural networks have been combined with transformers in DETR [29, 231] to achieve state-of-the-art results for object detection. More recent research has been exploring whether the whole convolutional neural networks (CNN) can be replaced by transformers only. The vision transformer [57] is one such pioneering work that replaces CNNs completely with transformers, and achieves state-of-the-art results for the classification task.

The most valuable point for the transformer is that it is a general computation unit, it could fit to many different tasks, and ease the interaction between different tasks. For example, NLP and vision tasks, e.g., visual question answering [212, 113].The transformer makes it possible to formulate power models purely based on it for different tasks. Not only for the object detection, but also in the field of tracking,

TransReID [78] has been proposed for object re-identification. All these innovations have shown the promising future of unified model design for different tasks related to dynamic scene understanding.

7.2.6 Learning with Self-Supervision

In chapter 5, we have introduced one possible way to relieve the problem of lacking video labels, which could happen since dense labels are expensive to acquire. In contrast, video data collection is much easier and cheaper, especially when compared with dense labeling for segmentation related tasks, e.g., video semantic segmentation, video object segmentation, and video instance segmentation. A better solution is required to relieve this problem for dynamic scene understanding. Self-supervised learning should be one of the promising research directions.

Self-supervised learning has been a hot research trend recently. It has been used as a pretext task for scene understanding. In the self-supervised learning, the neural networks are trained with supervisions from augmented data itself, e.g., predicting how each input image is rotated. Through such made-up tasks, the networks could learn useful intermediate semantic representation. The actual scores for the made-up tasks are not important, but the quality of the learned latent information matters. Deep neural networks could be fine-tuned in the down stream tasks to achieve better results with less data. As a result, if good intermediate features could be learned properly, it could potentially greatly reduce the cost of human labor. For static images, data augmentation is normally needed for the made-up supervision, e.g., image resizing, random coloring, and so on, but the specialty of video data is that the data itself already contains supervision signals, and there is less need for image data augmentation.

One important feature of the video data is the high correlation in consecutive images, there is large scene overlap between consecutive images, which makes self-supervised learning for correspondences between video frames possible. For video data, self-supervised learning does not even require any human labeling involvement as the supervision comes from the data itself, e.g., pixel color for image reconstruction. Recently, there have been more and more deep learning based works [109, 191, 115, 108] applying self-supervision learning for the video object segmentation tasks, which have received much attention since they have continuously improved the results and the performance is on par with some of the supervised learning methods. This direction is quite exciting and promising, and self-supervised learning should be the key to very large scale learning in order to tackle real-world video related applications. The combination of deep learning and self-supervised excel as deep neural networks are data hungry, and there is unlimited free video data online.

7.2.7 Need for Speed

For dynamic scene understanding, speed is one of the important factors that needs to be taken into consideration, especially for driving scenes since they change drastically in a few seconds. In real-time applications, the response time of deep neural networks is even more critical. Example research tasks are the semantic segmentation task on the cityscapes dataset [47] and the multi-object tracking and

segmentation task on the KITTI MOTS dataset [182] (used in chapter 6). For the cityscapes benchmark, several models are proposed for real-time semantic segmentation. The methods that have the most competitive performance are listed in Table 7.2, including U-HarDNet-70 [31], SwiftNetRN-18 [143], ShelfNet18 [234], and BiseNet V2 [214]. The speed of the models are measured on powerful GPUs, such as Titan Xp. There should be even more delay when on-board computing hardware is utilized for real-world applications. For example, a model running on the Jetson Xavier NX is $10\times$ slower than the same model running on a NVIDIA RTX 2080Ti GPU. In addition, there is still a large performance gap compared with top ranked models that are not real-time, e.g., [175] has a mIoU score of 85.4%, defeating the state-of-the-art real-time model by around 9%, which is a huge difference.

Table 7.2 Performance of the top ranked models for real-time semantic segmentation on Cityscapes benchmark.

Method	U-HarDNet-70	SwiftNetRN-18	ShelfNet18	BiseNet V2-Large	BiseNet V2
mIoU(%)	75.9	75.5	74.8	75.3	72.6
FPS	53	39.9	59.2	47.3	156

For the task of multi-object tracking and segmentation, the models designed [206, 207, 152, 182] normally require more time to run compared with models for real-time semantic segmentation. This is due to the fact that models for MOTS task are normally not end-to-end, and sequential CPU time is required in the detection and tracking parts. The speed of the state-of-the-art methods are reported in Table 7.3, which is far from real-time.

Table 7.3 Performance of the top ranked models for KITTI MOTS benchmark of car class.

Method	PointTrack	UMotsNet	MOTSFusion	TRCNN
sMOTSA(%)	78.5	76.5	75.0	67.0
FPS	22	7	2.27	2

It is clear that there is still a large gap for real-world, real-time applications and more research could be conducted in this direction.

In conclusion, for future works, unified dynamic scene understanding, i.e., a combination of detection, segmentation, tracking, and other sub-tasks, could be a trend. A transformer plus self-supervised learning is a promising research direction. Real-time processing for dynamic scene understanding requires further research in order to put the methods into usage for real-world applications.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Süsstrunk, et al. Slic superpixels compared to state-of-the-art superpixel methods. *PAMI*, 34(11):2274–2282, 2012.

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

[4] A. C. M. Alberto Sabater, Luis Montesano. Robust and efficient post-processing for video object detection. In *IROS*, 2020.

[5] A. Arnab and P. Torr. Bottom-up instance segmentation using deep higher-order crfs. In *BMVC*, 2016.

[6] A. Athar, S. Mahadevan, A. Ošep, L. Leal-Taixé, and B. Leibe. Stem-seg: Spatio-temporal embeddings for instance segmentation in videos. In *ECCV*, 2020.

[7] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. In *CVPR*, 2017.

[8] N. Ballas, L. Yao, C. Pal, and A. C. Courville. Delving deeper into convolutional networks for learning video representations. In *ICLR*, 2016.

[9] L. Bao, B. Wu, and W. Liu. Cnn in mrf: Video object segmentation via inference in a cnn-based higher-order spatio-temporal mrf. In *CVPR*, June 2018.

[10] B. Benfold and I. Reid. Stable multi-target tracking in real-time surveillance video. In *CVPR*, pages 3457–3464, 2011.

[11] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *Journal on Image Video Process.*, 2008.

[12] G. Bertasius and L. Torresani. Classifying, segmenting, and tracking object instances in video with mask propagation. In *CVPR*, 2020.

[13] G. Bertasius, L. Torresani, and J. Shi. Object detection in video with spatiotemporal sampling networks. In *ECCV*, 2018.

[14] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In *ECCVw*, 2016.

[15] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. In *ICIP*, 2016.

[16] G. Bhat, F. J. Lawin, M. Danelljan, A. Robinson, M. Felsberg, L. Van Gool, and R. Timofte. Learning what to learn for video object segmentation. In *ECCV*, pages 777–794. Springer, 2020.

[17] A. F. Bobick and J. W. Davis. The recognition of human movement using temporal templates. *TPAMI*, pages 257–267, 2001.

[18] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee. Yolact: Real-time instance segmentation. In *ICCV*, 2019.

[19] G. Braso and L. Leal-Taixe. Learning a neural solver for multiple object tracking. In *CVPR*, 2020.

[20] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV*, pages 44–57, 2008.

[21] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

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

[23] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *PAMI*, 33(3):500–513, 2011.

[24] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video object segmentation. In *CVPR*, 2017.

[25] H. Caesar, J. Uijlings, and V. Ferrari. Coco-stuff: Thing and stuff classes in context. In *CVPR*, June 2018.

[26] Z. Cai and N. Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018.

[27] M. Campos-Taberner, A. Romero-Soriano, C. Gatta, G. Camps-Valls, A. Lagrange, B. L. Saux, A. Beaupère, A. Boulch, A. Chan-Hon-Tong, S. Herbin, H. Randrianarivo, M. Ferecatu, M. Shimoni, G. Moser, and D. Tuia. Processing of extremely high-resolution lidar and rgb data: Outcome of the 2015 ieee grss data fusion contest–part a: 2-d contest. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2016.

[28] J. Cao, R. M. Anwer, H. Cholakkal, F. S. Khan, Y. Pang, and L. Shao. Sipmask: Spatial information preservation for fast instance segmentation. In *ECCV*, 2020.

[29] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. *CoRR*, 2020.

[30] E. Casella, A. Rovere, A. Pedroncini, L. Mucerino, M. Casella, L. A. Cusati, M. Vacchi, M. Ferrari, and M. Firpo. Study of wave runup using numerical models and low-altitude aerial photogrammetry: A tool for coastal management. *Estuarine, Coastal and Shelf Science*, 149:160–167, 2014.

[31] P. Chao, C.-Y. Kao, Y.-S. Ruan, C.-H. Huang, and Y.-L. Lin. Hardnet: A low memory traffic network. In *ICCV*, pages 3552–3561, 2019.

[32] N. Chebrolu, T. Läbe, and C. Stachniss. Robust long-term registration of uav images of crop fields for precision agriculture. *IEEE Robotics and Automation Letters*, 3(4), 2018.

[33] H. Chen, K. Sun, Z. Tian, C. Shen, Y. Huang, and Y. Yan. Blendmask: Top-down meets bottom-up for instance segmentation. In *CVPR*, 2020.

[34] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin. Mmdetection: Open mmlab detection toolbox and benchmark. *CoRR*, 2019.

[35] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017.

[36] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *CVPR*, June 2016.

[37] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, September 2018.

[38] Y. Chen, Y. Cao, H. Hu, and L. Wang. Memory enhanced global-local aggregation for video object detection. In *CVPR*, 2020.

[39] Y. Chen, J. Pont-Tuset, A. Montes, and L. Van Gool. Blazingly fast video object segmentation with pixel-wise metric learning. In *CVPR*, 2018.

[40] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *CVPR*, 2020.

[41] J. Cheng, Y.-H. Tsai, W.-C. Hung, S. Wang, and M.-H. Yang. Fast and accurate online video object segmentation via tracking parts. In *CVPR*, pages 7415–7424, 2018.

[42] J. Cheng, Y.-H. Tsai, S. Wang, and M.-H. Yang. Segflow: Joint learning for video object segmentation and optical flow. In *ICCV*, 2017.

[43] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S.-M. Hu. Global contrast based salient region detection. *TPAMI*, 37(3):569–582, 2015.

[44] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734. Association for Computational Linguistics, 2014.

[45] H. Ci, C. Wang, and Y. Wang. Video object segmentation by learning location-sensitive embeddings. In *ECCV*, September 2018.

[46] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C: Emerging Technologies*, 6(4):271–288, 1998.

[47] 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 *CVPR*, 2016.

[48] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *NeurIPS*, 2016.

[49] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *ICCV*, 2017.

[50] C. Debes, A. Merentitis, R. Heremans, J. Hahn, N. Frangiadakis, T. van Kasteren, W. Liao, R. Bellens, A. Pizurica, S. Gautama, W. Philips, S. Prasad, Q. Du, and F. Pacifici. Hyperspectral and lidar data fusion: Outcome of the 2013 grss data fusion contest. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7, 05 2014.

[51] I. Demir, K. Koperski, D. Lindenbaum, G. Pang, J. Huang, S. Basu, F. Hughes, D. Tuia, and R. Raska. Deepglobe 2018: A challenge to parse the earth through satellite images. In *CVPRW*, 2018.

[52] H. Deng, Y. Hua, T. Song, Z. Zhang, Z. Xue, R. Ma, N. Robertson, and H. Guan. Object guided external memory network for video object detection. In *ICCV*, 2019.

[53] J. Deng, Y. Pan, T. Yao, W. Zhou, H. Li, and T. Mei. Relation distillation networks for video object detection. In *ICCV*, 2019.

[54] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bi-directional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[55] S. S. Y. J. D. L. D. C.-O. H. H. Di Lin, Dingguo Shen. Zigzagnet: Fusing top-down and bottom-up context for object segmentation. In *CVPR*, 2019.

[56] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *PAMI*, 37(8):1558–1570, 2015.

[57] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, 2020.

[58] 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 *ICCV*, 2015.

[59] D. Du, Y. Qi, H. Yu, Y. Yang, K. Duan, G. Li, W. Zhang, Q. Huang, and Q. Tian. The unmanned aerial vehicle benchmark: object detection and tracking. In *ECCV*, 2018.

[60] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 111(1):98–136, Jan. 2015.

[61] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, June 2010.

[62] K. Fang, Y. Xiang, X. Li, and S. Savarese. Recurrent autoregressive networks for online multi-object tracking. In *WACV*, 2018.

[63] C. Feichtenhofer, A. Pinz, and A. Zisserman. Detect to track and track to detect. In *ICCV*, 2017.

[64] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *NeurIPS*, pages 64–72, 2016.

[65] C. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. DSSD : Deconvolutional single shot detector. *CoRR*, abs/1701.06659, 2017.

[66] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu. Dual attention network for scene segmentation. In *CVPR*, 2019.

[67] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[68] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[69] R. Girshick. Fast r-cnn. In *ICCV*, 2015.

[70] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[71] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *ICML*, pages 1764–1772, 2014.

[72] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra. Draw: A recurrent neural network for image generation. In F. Bach and D. Blei, editors, *ICML*, volume 37 of *Proceedings of Machine Learning Research*, pages 1462–1471, Lille, France, 07–09 Jul 2015. PMLR.

[73] C. Guo, B. Fan, J. Gu, Q. Zhang, S. Xiang, V. Prinet, and C. Pan. Progressive sparse local attention for video object detection. In *ICCV*, 2019.

[74] M. Han, Y. Wang, X. Chang, and Y. Qiao. Mining inter-video proposal relations for video object detection. In *ECCV*, 2020.

[75] W. Han, P. Khorrami, T. L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang. Seq-nms for video object detection. *CoRR*, abs/1602.08465, 2016.

[76] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. In *ICCV*, 2017.

[77] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[78] S. He, H. Luo, P. Wang, F. Wang, H. Li, and W. Jiang. Transreid: Transformer-based object re-identification, 2021.

[79] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *ECCV*, 2016.

[80] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[81] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[82] D. Hoiem, J. Hays, J. Xiao, and A. Khosla. Guest editorial: Scene understanding. *IJCV*, 112(2):131–132, 2015.

[83] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[84] A. Hu, A. Kendall, and R. Cipolla. Learning a spatio-temporal embedding for video instance segmentation. *CoRR*, abs/1912.08969, 2019.

[85] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei. Relation networks for object detection. In *CVPR*, 2018.

[86] P. Hu, G. Wang, X. Kong, J. Kuen, and Y.-P. Tan. Motion-guided cascaded refinement network for video object segmentation. In *CVPR*, June 2018.

[87] Y.-T. Hu, J.-B. Huang, and A. Schwing. Maskrnn: Instance level video object segmentation. In *NeurIPS*, pages 325–334, 2017.

[88] Y.-T. Hu, J.-B. Huang, and A. G. Schwing. Videomatch: Matching based video object segmentation. In *ECCV*, September 2018.

[89] L. Huang, X. Zhao, and K. Huang. Bridging the gap between detection and tracking: A unified approach. In *ICCV*, pages 3999–4009, 2019.

[90] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu. Ccnet: Criss-cross attention for semantic segmentation. In *ICCV*, 2019.

[91] T.-W. Hui, X. Tang, and C. Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *CVPR*, 2018.

[92] D. R. Ishan Nigam, Chen Huang. Ensemble knowledge transfer for semantic segmentation. In *WACV*, 2018.

[93] V. Jampani, R. Gadde, and P. V. Gehler. Video propagation networks. In *CVPR*, 2017.

[94] W.-D. Jang and C.-S. Kim. Online video object segmentation via convolutional trident network. In *CVPR*, pages 5849–5858, 2017.

[95] K. Kang, H. Li, T. Xiao, W. Ouyang, J. Yan, X. Liu, and X. Wang. Object detection in videos with tubelet proposal networks. In *CVPR*, 2017.

[96] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, and W. Ouyang. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[97] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, pages 3128–3137, 2015.

[98] A. Khoreva, R. Benenson, E. Ilg, T. Brox, and B. Schiele. Lucid data dreaming for object tracking. In *The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*, 2017.

[99] A. Kim, A. Ošep, and L. Leal-Taixé. Eagermot: Real-time 3d multi-object tracking and segmentation via sensor fusion. In *ICRA*, 2021.

[100] B. Kim, J. Yim, and J. Kim. Highway driving dataset for semantic video segmentation. In *BMVC*, 2018.

[101] D. Kim, S. Woo, J.-Y. Lee, and I. S. Kweon. Video panoptic segmentation. In *CVPR*, pages 9859–9868, 2020.

[102] A. Kirillov, R. Girshick, K. He, and P. Dollar. Panoptic feature pyramid networks. In *CVPR*, 2019.

[103] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollar. Panoptic segmentation. In *CVPR*, 2019.

[104] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother. Instancecut: From edges to instances with multicut. In *CVPR*, 2017.

[105] H. W. Kuhn and B. Yaw. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 1955.

[106] S. Kumaar, Y. Lyu, F. Nex, and M. Y. Yang. Cabinet: Efficient context aggregation network for low-latency semantic segmentation. In *ICRA*, 2021.

[107] A. Kundu, V. Vineet, and V. Koltun. Feature space optimization for semantic video segmentation. In *CVPR*, pages 3168–3175, 2016.

[108] Z. Lai, E. Lu, and W. Xie. Mast: A memory-augmented self-supervised tracker. In *CVPR*, pages 6479–6488, 2020.

[109] Z. Lai and W. Xie. Self-supervised learning for video correspondence flow. In *BMVC*, 2019.

[110] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, 2019.

[111] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, 2018.

[112] J. Li, X. Gao, and T. Jiang. Graph networks for multiple object tracking. In *WACV*, 2020.

[113] L. H. Li, M. Yatskar, D. Yin, C. Hsieh, and K. Chang. Visualbert: A simple and performant baseline for vision and language. *CoRR*, abs/1908.03557, 2019.

[114] X. Li and C. Change Loy. Video object segmentation with joint re-identification and attention-aware mask propagation. In *ECCV*, September 2018.

[115] X. Li, S. Liu, S. De Mello, X. Wang, J. Kautz, and M.-H. Yang. Joint-task self-supervised learning for temporal correspondence. In *NeurIPS*, 2019.

[116] X. Li, Y. Qi, Z. Wang, K. Chen, Z. Liu, J. Shi, P. Luo, X. Tang, and C. C. Loy. Video object segmentation with re-identification. In *The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*, 2017.

[117] Y. Li and A. Gupta. Beyond grids: Learning graph representations for visual recognition. In *NeurIPS*, 2018.

[118] X. Liang, Z. Hu, H. Zhang, L. Lin, and E. P. Xing. Symbolic graph reasoning meets convolutions. In *NeurIP*, 2018.

[119] C.-C. Lin, Y. Hung, R. Feris, and L. He. Video instance segmentation tracking with a modified vae architecture. In *CVPR*, 2020.

[120] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[121] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[122] 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 D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *ECCV*, 2014.

[123] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018.

[124] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[125] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *CoRR*, abs/1506.04579, 2015.

[126] Y. Liu, B. Fan, L. Wang, J. Bai, S. Xiang, and C. Pan. Semantic labeling in very high resolution images via a self-cascaded convolutional neural network. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2018.

[127] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *ICCV*, pages 3730–3738, 2015.

[128] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, June 2015.

[129] P. Lottes, R. Khanna, J. Pfeifer, R. Siegwart, and C. Stachniss. Uav-based crop and weed classification for smart farming. In *ICRA*, pages 3024–3031, 2017.

[130] J. Luiten, T. Fischer, and B. Leibe. Track to reconstruct and reconstruct to track. *RA-L*, 2020.

[131] J. Luiten, P. Voigtlaender, and B. Leibe. Premvos: Proposal-generation, refinement and merging for video object segmentation. In *ACCV*, 2018.

[132] Y. Lyu, G. Vosselman, G.-S. Xia, and M. Y. Yang. Bidirectional multi-scale attention networks for semantic segmentation of oblique uav imagery. In *ISPRS Congress*, 2021.

[133] Y. Lyu, G. Vosselman, G.-S. Xia, A. Yilmaz, and M. Y. Yang. Uavid: A semantic segmentation dataset for uav imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 165:108 – 119, 2020.

[134] Y. Lyu, G. Vosselman, G.-S. Xia, and M. Ying Yang. Lip: Learning instance propagation for video object segmentation. In *ICCVw*, Oct 2019.

[135] Y. Lyu, M. Y. Yang, G. Vosselman, and G.-S. Xia. Video object detection with a convolutional regression tracker. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2021.

[136] K.-K. Maninis, S. Caelles, Y. Chen, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. Video object segmentation without temporal information. *TPAMI*, 2018.

[137] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler. Mot16: A benchmark for multi-object tracking. *CoRR*, abs/1603.00831, 2016.

[138] A. Milioto, P. Lottes, and C. Stachniss. Real-time blob-wise sugar beets vs weeds classification for monitoring fields using convolutional neural networks. *ISPRS Annals*, 4:41, 2017.

[139] M. Mueller, N. Smith, and B. Ghanem. A benchmark and simulator for uav tracking. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *ECCV*, 2016.

[140] G. Neuhold, T. Ollmann, S. Rota Bulò, and P. Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *ICCV*, 2017.

[141] D. Neven, B. D. Brabandere, M. Proesmans, and L. V. Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *CVPR*, 2019.

[142] S. W. Oh, J.-Y. Lee, N. Xu, and S. J. Kim. Video object segmentation using space-time memory networks. In *ICCV*, October 2019.

[143] M. Orsic, I. Kreso, P. Bevandic, and S. Segvic. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. In *CVPR*, pages 12607–12616, 2019.

[144] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin. Libra r-cnn: Towards balanced learning for object detection. In *CVPR*, 2019.

[145] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *ICML*, pages 1310–1318, 2013.

[146] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito,

M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[147] S. Penmetsa, F. Minhuj, A. Singh, and S. Omkar. Autonomous uav for suspicious action detection using pictorial human pose estimation and classification. *ELCVIA: electronic letters on computer vision and image analysis*, 13(1):0018–32, 2014.

[148] F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A.Sorkine-Hornung. Learning video object segmentation from static images. In *CVPR*, 2017.

[149] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016.

[150] D. Perez, I. Maza, F. Caballero, D. Scarlatti, E. Casado, and A. Ollero. A ground control station for a multi-uav surveillance system. *Journal of Intelligent&Robotic Systems*, 69(1-4):119–130, 2013.

[151] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool. The 2017 davis challenge on video object segmentation. *CoRR*, 2017.

[152] L. Porzi, M. Hofinger, I. Ruiz, J. Serrat, S. R. Bulo, and P. Kontschieder. Learning multi-object tracking and segmentation from automatic annotations. In *CVPR*, 2020.

[153] L. Qi, L. Jiang, S. Liu, X. Shen, and J. Jia. Amodal instance segmentation with kins dataset. In *CVPR*, 2019.

[154] S. Qiao, L.-C. Chen, and A. Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. *CoRR*, abs/2006.02334, 2020.

[155] S. Qiao, Y. Zhu, H. Adam, A. Yuille, and L.-C. Chen. Vip-deeplab: Learning visual perception with depth-aware video panoptic segmentation. In *CVPR*, 2021.

[156] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[157] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *CVPR*, 2017.

[158] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu. Accurate single stage detector using recurrent rolling convolution. In *CVPR*, 2017.

[159] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *PAMI*, 39(6):1137–1149, 2017.

[160] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *ECCV*, pages 549–565, 2016.

[161] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[162] F. Rottensteiner, G. Sohn, M. Gerke, J. Wegner, U. Breitkopf, and J. Jung. Results of the isprs benchmark on urban object detection and 3d building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 93:256–271, 2014.

[163] S. Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[164] D. E. Rumelhart, G. E. Hinton, R. J. Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[165] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[166] T. Scharwächter, M. Enzweiler, U. Franke, and S. Roth. Efficient multi-cue scene segmentation. In *GCPR*, pages 435–445, 2013.

[167] E. Semsch, M. Jakob, D. Pavlicek, and M. Pechoucek. Autonomous uav surveillance in complex urban environments. In *WI-IAT*, pages 82–85, 2009.

[168] S. Sharma, J. A. Ansari, J. K. Murthy, and K. M. Krishna. Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking. In *ICRA*, 2018.

[169] J. Shi, Q. Yan, L. Xu, and J. Jia. Hierarchical image saliency detection on extended cssd. *TPAMI*, 38(4):717–729, 2016.

[170] J. Shin Yoon, F. Rameau, J. Kim, S. Lee, S. Shin, and I. So Kweon. Pixel-level matching for video object segmentation using convolutional neural networks. In *ICCV*, pages 2167–2176, 2017.

[171] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016.

[172] M. Shvets, W. Liu, and A. C. Berg. Leveraging long-range temporal relationships between proposals for video object detection. In *ICCV*, 2019.

[173] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *ECCV*, pages 438–451, 2010.

[174] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[175] A. Tao, K. Sapra, and B. Catanzaro. Hierarchical multi-scale attention for semantic segmentation. *CoRR*, abs/1910.12037, 2020.

[176] Z. Teed and J. Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020.

[177] P. Tokmakov, K. Alahari, and C. Schmid. Learning video object segmentation with visual memory. In *ICCV*, Oct 2017.

[178] X.-Y. Tong, G.-S. Xia, Q. Lu, H. Shen, S. Li, S. You, and L. Zhang. Learning transferable deep models for land-use classification with high-resolution remote sensing images. *CoRR*, 2018.

[179] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. S. Torr. End-to-end representation learning for correlation filter based tracking. In *CVPR*, 2017.

[180] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[181] P. Voigtlaender, Y. Chai, F. Schroff, H. Adam, B. Leibe, and L.-C. Chen. Feelvos: Fast end-to-end embedding learning for video object segmentation. In *CVPR*, June 2019.

[182] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe. Mots: Multi-object tracking and segmentation. In *CVPR*, 2019.

[183] P. Voigtlaender and B. Leibe. Online adaptation of convolutional neural networks for video object segmentation. In *BMVC*, 2017.

[184] P. Voigtlaender, J. Luiten, P. H. Torr, and B. Leibe. Siam r-cnn: Visual tracking by re-detection. In *CVPR*, June 2020.

[185] G. Wang, C. Luo, X. Sun, Z. Xiong, and W. Zeng. Tracking by instance detection: A meta-learning approach. In *CVPR*, pages 6288–6297, 2020.

[186] H. Wang, Y. Zhu, B. Green, H. Adam, A. Yuille, and L.-C. Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *ECCV*, 2020.

[187] J. Wang, K. Chen, R. Xu, Z. Liu, C. C. Loy, and D. Lin. Carafe: Content-aware reassembly of features. In *ICCV*, 2019.

[188] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao. Deep high-resolution representation learning for visual recognition. *TPAMI*, 2019.

[189] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *ICCV*, 2015.

[190] S. Wang, Y. Zhou, J. Yan, and Z. Deng. Fully motion-aware network for video object detection. In *ECCV*, 2018.

[191] X. Wang, A. Jabri, and A. A. Efros. Learning correspondence from the cycle-consistency of time. In *CVPR*, 2019.

[192] Y. Wang, Y. Lyu, Y. Cao, and M. Y. Yang. Deep learning for semantic segmentation of uav videos. In *IGARSS*, 2019.

[193] Z. Wang, L. Zheng, Y. Liu, and S. Wang. Towards real-time multi-object tracking. In *ECCV*, 2020.

[194] N. Wojke and A. Bewley. Deep cosine metric learning for person re-identification. In *WACV*, 2018.

[195] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *ICIP*, 2017.

[196] H. Wu, Y. Chen, N. Wang, and Z. Zhang. Sequence level semantics aggregation for video object detection. In *ICCV*, 2019.

[197] Y. Wu and K. He. Group normalization. In *ECCV*, September 2018.

[198] S. Wug Oh, J.-Y. Lee, K. Sunkavalli, and S. Joo Kim. Fast video object segmentation by reference-guided mask propagation. In *CVPR*, June 2018.

[199] H. Xiang and L. Tian. Development of a low-cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (uav). *Biosystems Engineering*, 108(2):174–190, 2011.

[200] F. Xiao and Y. Jae Lee. Video object detection with an aligned spatial-temporal memory. In *ECCV*, 2018.

[201] H. Xiao, J. Feng, G. Lin, Y. Liu, and M. Zhang. Monet: Deep motion exploitation for video object segmentation. In *CVPR*, June 2018.

[202] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[203] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NeurIPS*, pages 802–810, 2015.

[204] J. Xu, Y. Cao, Z. Zhang, and H. Hu. Spatial-temporal relation networks for multi-object tracking. In *ICCV*, 2019.

[205] Y. Xu, Z. Wang, Z. Li, Y. Yuan, and G. Yu. Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. In *AAAI*, 2020.

[206] Z. Xu, W. Zhang, X. Tan, W. Yang, H. Huang, S. Wen, E. Ding, and L. Huang. Segment as points for efficient online multi-object tracking and segmentation. In *ECCV*, 2020.

[207] Z. Xu, W. Zhang, X. Tan, W. Yang, X. Su, Y. Yuan, H. Zhang, S. Wen, E. Ding, and L. Huang. Pointtrack++ for effective online multi-object tracking and segmentation. In *CVPRw*, 2020.

[208] L. Yang, Y. Fan, and N. Xu. Video instance segmentation. In *ICCV*, 2019.

[209] L. Yang, Y. Wang, X. Xiong, J. Yang, and A. K. Katsaggelos. Efficient video object segmentation via network modulation. In *CVPR*, pages 6499–6507, 2018.

[210] M. Y. Yang, S. Kumaar, Y. Lyu, and F. Nex. Real-time semantic segmentation with context aggregation network. *ISPRS Journal of Photogrammetry and Remote Sensing*, 178:124–134, 2021.

[211] M. Y. Yang, W. Liao, H. Ackermann, and B. Rosenhahn. On support relations and semantic scene graphs. *ISPRS Journal of Photogrammetry and Remote Sensing*, 131:15–25, 2017.

[212] Z. Yang, N. Garcia, C. Chu, M. Otani, Y. Nakashima, and H. Takemura. Bert representations for video question answering. In *WACV*, pages 1556–1565, 2020.

[213] Z. Yang, Y. Wei, and Y. Yang. Collaborative video object segmentation by foreground-background integration. In *ECCV*, pages 332–348. Springer, 2020.

[214] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *CoRR*, 2020.

[215] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *ECCV*, 2018.

[216] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[217] F. Yu, W. Li, Q. Li, Y. Liu, X. Shi, and J. Yan. Poi: Multiple object tracking with high performance detection and appearance feature. In *ECCVw*, 2016.

[218] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell. Bdd100k: A diverse driving video database with scalable annotation tooling. *CoRR*, 2018.

[219] Y. Yuan, X. Chen, and J. Wang. Object-contextual representations for semantic segmentation. In *ECCV*, 2020.

[220] Y. Yuan and J. Wang. Ocnet: Object context network for scene parsing. *CoRR*, abs/1809.00916, 2018.

[221] H. Z. R. H. M. B. E. Y.-R. U. Yuwen Xiong, Renjie Liao. Upsnet: A unified panoptic segmentation network. In *CVPR*, 2019.

[222] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016.

[223] Z. Zhang, D. Cheng, X. Zhu, S. Lin, and J. Dai. Integrated object detection and tracking with tracklet-conditioned detection. *CoRR*, abs/1811.11167, 2018.

[224] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[225] S. Zhao, Y. Wang, Z. Yang, and D. Cai. Region mutual information loss for semantic segmentation. In *NeurIPS*, 2019.

[226] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *CVPR*, volume 1, page 4, 2017.

[227] Z. Zhou, J. Xing, M. Zhang, and W. Hu. Online multi-target tracking with tensor-based high-order graph matching. In *ICPR*, 2018.

[228] L. Zhu and Y. Yang. Actbert: Learning global-local video-text representations. In *CVPR*, June 2020.

[229] P. Zhu, L. Wen, X. Bian, L. Haibin, and Q. Hu. Vision meets drones: A challenge. *CoRR*, 2018.

[230] X. Zhu, H. Hu, S. Lin, and J. Dai. Deformable convnets v2: More deformable, better results. *CoRR*, abs/1811.11168, 2018.

[231] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai. Deformable detr: Deformable transformers for end-to-end object detection. *CoRR*, 2020.

[232] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei. Flow-guided feature aggregation for video object detection. In *ICCV*, 2017.

[233] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. Deep feature flow for video recognition. In *CVPR*, 2017.

[234] J. Zhuang, J. Yang, L. Gu, and N. Dvornek. Shelfnet for fast semantic segmentation. In *ICCVw*, 2019.

Summary

Scene understanding is an important and fundamental research field in computer vision, which is quite useful for many applications in photogrammetry and remote sensing. It focuses on locating and classifying objects in images, understanding the relationships between them. The higher goal is to interpret what event happens in the scene, when it happens and why it happens, and what should we do based on the information. Dynamic scene understanding is to use information from different time to interpret scenes and answer the above related questions.

For modern scene understanding technology, deep learning has shown great potential for such task. "Deep" in deep learning refers to the use of multiple layers in the neural networks. Deep neural networks are powerful as they are highly non-linear function that possess the ability to map from one domain to another quite different domain after proper training. It is the best solution for many fundamental research tasks regarding scene understanding. This ph.D. research also takes advantage of deep learning for dynamic scene understanding.

Temporal information plays an important role for dynamic scene understanding. Compared with static scene understanding from images, information distilled from the time dimension provides values in many different ways. Images across consecutive frames have very high correlation, i.e., objects observed in one frame have very high chance to be observed and identified in nearby frames as well. Such redundancy in observation could potentially reduce the uncertainty for object recognition with deep learning based methods, resulting in more consistent inference. High correlation across frames could also improve the chance for recognizing objects correctly. If the camera or the object moves, the object could be observed in multiple different views with different poses and appearance. The information captured for object recognition would be more diverse and complementary, which could be aggregated to jointly inference the categories and the properties of objects.

This ph.D. research involves several tasks related to the dynamic scene understanding in computer vision, including semantic segmentation for aerial platform images (chapter 2, 3), video object segmentation and video object detection for common objects in natural scenes (chapter 4, 5), and multi-object tracking and segmentation for cars and pedestrians in driving scenes(chapter 6).

Chapter2 investigates how to establish the semantic segmentation benchmark for the UAV images, which includes data collection, data labeling, dataset construction, and performance evaluation with baseline deep neural networks and the proposed multi-scale dilation net. Conditional random field with feature space optimization is used to achieve consistent semantic segmentation prediction in videos.

Chapter3 investigates how to better extract the scene context information for better object recognition performance by proposing the novel bidirectional multi-scale attention networks. It achieves better performance by inferring features and attention weights for feature fusing from both higher level and lower level branches.

Chapter4 investigates how to simultaneously segment multiple objects across multiple frames by combining memory modules with instance segmentation networks. Our method learns to propagate the target object labels without auxiliary data, such as optical flow, which simplifies the model.

Chapter5 investigates how to improve the performance of well-trained object detectors with a light weighted and efficient plug&play tracker for object detection in video. This chapter also investigates how the proposed model performs when lacking video training data.

Chapter6 investigates how to improve the performance of detection, segmentation, and tracking by jointly considering top-down and bottom-up inference. The whole pipeline follows the multi-task design, i.e., a single feature extraction backbone with multiple heads for different sub-tasks.

Overall, this manuscript has delved into several different computer vision tasks, which share fundamental research problems, including detection, segmentation, and tracking. Based on the research experiments and knowledge from literature review, several reflections regarding dynamic scene understanding have been discussed: The range of object context influence the quality for object recognition; The quality of video data affect the method choice for specific computer vision task; Detection and tracking are complementary for each other. For future work, unified dynamic scene understanding task could be a trend, and transformer plus self-supervised learning is one promising research direction. Real-time processing for dynamic scene understanding requires further researches in order to put the methods into usage for real-world applications.

Samenvatting

Scene understanding is een belangrijk en fundamenteel onderzoeksgebied in computer vision, dat zeer nuttig is voor vele toepassingen in fotogrammetrie en remote sensing. Het richt zich op het lokaliseren en classificeren van objecten in beelden en het begrijpen van de relaties tussen objecten. Het hogere doel is te interpreteren welke gebeurtenis in de scène plaatsvindt, wanneer en waarom dit gebeurt en wat we op basis van deze informatie moeten doen. Dynamische scene understanding is het gebruiken van informatie van verschillende tijden om scènes te interpreteren en de bovenstaande vragen te beantwoorden.

Voor moderne technologie voor het begrijpen van scènes heeft deep learning aangetoond over een groot potentieel voor dergelijke taken te beschikken. "Deep" in deep learning verwijst naar het gebruik van meerdere lagen in de neurale netwerken. Diepe neurale netwerken zijn krachtig omdat het sterk niet-lineaire functies zijn die het vermogen bezitten om na de juiste training van het ene domein naar een heel ander domein te gaan. Het is de beste oplossing voor veel fundamentele onderzoekstaken met betrekking tot het begrijpen van scènes. Dit promotieonderzoek maakt ook gebruik van deep learning voor het begrijpen van dynamische scènes.

Temporele informatie speelt een belangrijke rol in het begrijpen van dynamische scènes. Vergeleken met statisch begrip van scènes op basis van beelden, is informatie die op verschillende momenten is waargenomen op veel verschillende manieren waardevol. Beelden in opeenvolgende frames hebben een zeer hoge correlatie, d.w.z. dat objecten die in één frame worden waargenomen een zeer hoge kans hebben om ook in nabije frames te worden waargenomen en geïdentificeerd. Dergelijke redundantie in de waarnemingen kan mogelijk de onzekerheid voor objectherkenning met deep learning gebaseerde methoden verminderen, wat resulteert in meer consistente gevolgtrekking. Hoge correlatie tussen frames kan ook de kans op het correct herkennen van objecten verbeteren. Als de camera of het object beweegt, kan het object worden waargenomen in meerdere beelden met verschillende poses en verschijningsvormen. De informatie die wordt verzameld voor objectherkenning is dan meer divers en complementair en kan worden samengevoegd om de categorieën en de eigenschappen van meerdere objecten in samenhang af te leiden.

Dit promotieonderzoek omvat verschillende taken gerelateerd aan het dynamische scene understanding in computer vision, waaronder semantische segmentatie voor luchtfoto's (hoofdstuk 2, 3), video objectsegmentatie en video-objectdetectie voor objecten in natuurlijke scènes (hoofdstuk 4, 5), en het volgen en segmenteren van meerdere objecten zoals auto's en voetgangers in verkeersscènes (hoofdstuk 6).

Hoofdstuk2 onderzoekt hoe een semantische segmentatie benchmark voor

UAV beelden tot stand kan worden gebracht. Dit omvat dataverzameling, data labelling, datasetconstructie, en de evaluatie van twee netwerken die als referentie (baseline) dienen, waaronder het voorgestelde multi-scale dilatatienetwerk. Conditional random fields met optimalisatie van kenmerken wordt gebruikt om consistente semantische segmentatie in video's te bereiken.

Hoofdstuk3 onderzoekt hoe de contextinformatie van de scène beter kan worden geëxtraheerd voor betere objectherkenning door een nieuw tweezijdig meerschalig attention netwerk voor te stellen. Het bereikt betere prestaties door het afleiden en fuseren van kenmerken gericht op verschillende beeldschalen.

Hoofdstuk4 onderzoekt hoe je tegelijkertijd meerdere objecten kunt segmenteren over meerdere frames door geheugenmodules te combineren met instantie-segmentatienetwerken. Onze methode leert om de labels te propageren zonder hulpdata, zoals optical flow, wat het model vereenvoudigt.

Hoofdstuk5 onderzoekt hoe de prestaties van goed getrainde objectdetectoren verbeterd kunnen worden met een efficiënte plug & play volger voor objectdetectie in video. Dit hoofdstuk onderzoekt ook hoe het voorgestelde model presteert bij gebrek aan videotrainingsdata.

Hoofdstuk6 onderzoekt hoe de prestaties van detectie, segmentatie en tracking verbeterd kunnen worden door gezamenlijk top-down en bottom-up inferentie te beschouwen. De gehele pijplijn volgt het multi-task ontwerp, d.w.z. een basis voor kenmerkextractie met meerdere deelnetwerken voor verschillende taken.

In het algemeen heeft dit manuscript zich verdiept in verschillende computer vision taken, die fundamentele onderzoekstaken delen, waaronder detectie, segmentatie en tracking. Gebaseerd op de experimenten en kennis uit literatuuronderzoek, zijn verschillende beschouwingen met betrekking tot dynamische scene understanding besproken: Het bereik van objectcontext beïnvloedt de kwaliteit van objectherkenning; De kwaliteit van videogegevens beïnvloedt de methodekeuze voor specifieke computer vision taken; Detectie en tracking zijn complementair aan elkaar. Voor toekomstig werk zou integrale dynamische scene understanding, d.w.z. een combinatie van detectie, segmentatie, volgen en andere taken, een trend kunnen zijn. Een transformator plus zelf-ondersteund leren is een veelbelovende onderzoeksrichting. Real-time verwerking voor dynamische scene understanding vereist verder onderzoek om de methoden in gebruik te kunnen nemen voor grootschalige toepassingen.

Author's Biography

Ye Lyu was born on July 21st, 1991 in Wuhan, China. A son of Chinese parents, he has Chinese nationality. Ye grew up in Wuhan, China, where he graduated from the No.3 Middle School of Wuhan in 2010 and started his college life at Wuhan University. After finishing his B.Sc in 2014, he got postgraduate recommendation without entrance examination and started his M.Sc. at the school of remote sensing and information engineering, Wuhan University.

In September 1st, he started his ph.D. research at the Faculty of Geo-Information Science and Earth Observation (ITC) of the University of Twente in Enschede, Netherlands. The present manuscript is the result of this research. His first journal paper was published in ISPRS Journal of Photogrammetry and Remote Sensing, and was selected as the featured article in July, 2020. It was also nominated into the final round for the ITC publication award. As part of his Ph.D., he attended the following events to present and discuss his research:

Data	Event	Place
1st, April - 30th,June 2019	Academic Visiting	Wuhan University, China
27th, October - 2nd,November 2019	ICCV2019 workshop, oral presentation	Seoul, Korea
21st, November 2019	NCG Symposium, oral presentation	University of Twente, Netherlands
16th - 17th, December 2019	NCCV2019, poster presentation	Wageningen University, Netherlands
22nd, January 2020	DSI Robotics symposium, lab tour	University of Twente, Netherlands
30th, May - 5th June 2021	ICRA2021, paper presentation	Xi'an, China (virtual event)
5th - 9th, July 2021	ISPRS Congress, oral presentation	Nice, France (virtual event)

Towards the end of his ph.D. research, he was invited as independent reviewer for the manuscripts submitted to ISPRS Journal of Photogrammetry and Remote Sensing, Journal of Information fusion, and International Conference on Robotics and Automation.

During his Ph.D., he was involved in teaching courses for 2D&3D Scene Understanding, Scene Understanding with UAVs, and Scientific Geo-computing. He authored and co-authored the following publications:

Y. Lyu, G. Vosselman, G.-S. Xia, A. Yilmaz, and M. Y. Yang. Uavid: A semantic segmentation dataset for uav imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 165:108 – 119, 2020

Y. Lyu, G. Vosselman, G.-S. Xia, and M. Ying Yang. Lip: Learning instance propagation for video object segmentation. In *ICCVw*, Oct 2019

Y. Lyu, M. Y. Yang, G. Vosselman, and G.-S. Xia. Video object detection with a convolutional regression tracker. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2021

Y. Lyu, G. Vosselman, G.-S. Xia, and M. Y. Yang. Bidirectional multi-scale attention networks for semantic segmentation of oblique uav imagery. In *ISPRS Congress*, 2021

S. Kumaar, Y. Lyu, F. Nex, and M. Y. Yang. Cabinet: Efficient context aggregation network for low-latency semantic segmentation. In *ICRA*, 2021

Y. Wang, Y. Lyu, Y. Cao, and M. Y. Yang. Deep learning for semantic segmentation of uav videos. In *IGARSS*, 2019

M. Y. Yang, S. Kumaar, Y. Lyu, and F. Nex. Real-time semantic segmentation with context aggregation network. *ISPRS Journal of Photogrammetry and Remote Sensing*, 178:124–134, 2021

ITC dissertations

A complete list of ITC dissertations is online on the ITC website:
www.itc.nl/research/phd/phd_graduates.aspx.

This dissertation has number 400.