Spatio-Temporal Learning for Video Segmentation and Tracking

PhD Thesis by Matthieu Paul

Diss. ETH No. 29366

DISS. ETH NO. 29366

SPATIO-TEMPORAL LEARNING FOR VIDEO SEGMENTATION AND TRACKING

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES
(Dr. sc. ETH Zurich)

presented by

MATTHIEU PAUL

Diplôme d'Ingénieur
Grenoble INP - Ensimag

born on 20.06.1990
citizen of France

accepted on the recommendation of

Prof. Dr. Luc Van Gool, examiner,
Prof. Dr. Bastian Leibe, co-examiner,
Prof. Dr. Michael Felsberg, co-examiner,
Dr. Ralf Dragon, co-examiner,

2023

ABSTRACT

Video scene understanding engulfs several fundamental and challenging computer vision tasks that complement each other. Some of them are inherently reasoning on a set of consecutive images, while others can be tackled on each frame separately. In this thesis, we focus on a subset of these tasks, starting from a global scene understanding perspective with semantic segmentation, to finish on a more local one with Visual Object Tracking (VOT) and Video Object Segmentation (VOS). Within that scope, we investigate different means to leverage and combine temporal cues to improve scene understanding algorithms when processing videos.

More specifically, we start by analyzing in the first part how the spatio-temporal correlations found in videos can be used to either increase the frame rate or the accuracy of single-frame semantic segmentation methods. First, we use optical flow as a means to propagate semantic information across frames, and build a pipeline for real-time video semantic segmentation that balances the computation load between GPU and CPU. Instead of designing a heavy neural network that infers everything on the GPU, we propose to focus the GPU task on either predicting segmentation masks from scratch or refining propagated labels. At the same time, a fast optical flow running on the CPU provides the motion vectors to warp semantic labels and features from one frame to the next. The refinement is done by a lightweight module that considers potential optical flow mistakes. We propose several operating points offering different trade-offs between speed and accuracy, and observe that our approach can lead to massive speedups at the price of a small drop in segmentation accuracy.

Then, we propose to directly exploit temporal correlations and appearance cues without an additional optical flow module. To achieve this, we aggregate semantic information from previous frames in a memory module that can be used through attention mechanisms. We design our pipeline to first access the deep features from past frames stored in memory and match them in a local neighborhood around each pixel. These spatio-temporal cues are afterward fused together with the current frame encoding to improve the final segmentation prediction. Our approach introduces a set of simple yet generic modules which can convert virtually any existing single-frame method to a video pipeline. We demonstrate the improvements of our architecture in terms of segmentation accuracy on two popular single-frame semantic segmentation networks.

In the second part, we shift our focus to the tasks of tracking and segmenting single objects in a video and hope to bridge the gap between the two. We especially study how they are related and expose the benefits of working with segmentation masks in the context of VOT. To that end, we propose a segmentation-centric approach which, in contrast with most existing approaches, internally works with segmentation masks and predicts segmentation masks without the need for an additional module. A dedicated instance localization branch inspired by existing trackers is used to bring the necessary robustness for VOT challenges and to condition the segmentation decoder to predict the correct segmentation mask. We show that our unified architecture yields state-of-the-art results compared to other trackers, both in terms of robustness and accuracy, while generating accurate segmentation masks.

RÉSUMÉ

La compréhension des scènes vidéo englobe plusieurs tâches fondamentales et difficiles de vision par ordinateur qui se complètent les unes les autres. Certaines d'entre elles raisonnent par définition sur une succession d'images, tandis que d'autres peuvent être définies pour une simple image. Dans cette thèse, nous nous concentrons sur un sous-ensemble de ces tâches, en commençant par une perspective de compréhension globale de la scène avec la segmentation sémantique, pour finir sur une perspective plus locale avec le suivi visuel d'objets (VOT) et la segmentation d'objets vidéo (VOS). Dans ce cadre, nous étudions différents moyens d'exploiter et de combiner des indices temporels susceptibles d'améliorer les algorithmes de compréhension de scènes lors du traitement de vidéos.

Plus précisément, nous commençons par analyser dans la première partie comment les corrélations spatio-temporelles présentes dans les vidéos peuvent être utilisées pour augmenter le taux de trame ou la précision des méthodes de segmentation sémantique d'images. Tout d'abord, nous utilisons le flux optique comme moyen de propagation de l'information sémantique entre les images, et nous construisons un pipeline pour la segmentation sémantique des vidéos en temps réel qui équilibre la charge de calcul entre la carte graphique et le processeur. Plutôt que de concevoir un réseau de neurones complexe qui effectue tout le travail sur la carte graphique, nous proposons de concentrer les tâches de celle-ci sur la prédiction des masques de segmentation à partir de zéro et sur l'affinage des caractéristiques propagées. Dans le même temps, un flux optique rapide fonctionnant sur le processeur fournit les vecteurs de mouvement pour déformer les étiquettes sémantiques et les caractéristiques d'une image à l'autre. L'affinage est effectué par un module léger qui prend en compte les erreurs éventuelles du flux optique. Nous proposons plusieurs points de fonctionnement offrant différents compromis entre la vitesse et la précision, et nous observons que notre approche peut conduire à des accélérations non négligeables au prix d'une légère baisse de la précision de la segmentation.

Ensuite, nous proposons d'exploiter directement les corrélations temporelles et l'apparence visuelle de celles-ci sans module supplémentaire pour calculer le flux optique. Pour ce faire, nous regroupons les informations sémantiques des images précédentes dans un module mémoire qui peut être utilisé via des mécanismes d'attention. Nous concevons notre pipeline de manière à accéder d'abord aux caractéristiques profondes des images précédentes stockées dans la

mémoire et à les faire correspondre dans un voisinage local autour de chaque pixel. Ces indices spatio-temporels sont ensuite fusionnés avec l'encodage de l'image actuelle pour améliorer la prédiction finale de la segmentation. Notre approche introduit un ensemble de modules simples, mais génériques, qui peuvent convertir pratiquement n'importe quelle méthode existante fonctionnant sur une image en un pipeline vidéo. Nous démontrons les améliorations apportées par notre architecture en termes de précision de segmentation sur deux réseaux de neurones populaires de segmentation sémantique d'image.

Dans la deuxième partie, nous nous concentrons sur les tâches de suivi et de segmentation d'objets dans une vidéo et espérons combler le fossé entre celles-ci. Nous étudions en particulier les liens entre les deux et exposons les avantages de travailler avec des masques de segmentation dans le contexte du VOT. À cette fin, nous proposons une approche centrée sur la segmentation qui, contrairement à la plupart des approches existantes, fonctionne en interne avec les masques de segmentation et en prédit de nouveaux sans nécessiter de module supplémentaire. Une branche dédiée à la localisation inspirée des algorithmes de suivi d'objets existants est utilisée pour apporter la robustesse nécessaire aux défis VOT et pour conditionner le décodeur de segmentation à prédire le masque de segmentation pour la bonne cible. Nous montrons que notre architecture unifiée donne des résultats de pointe par rapport à d'autres algorithmes de suivi d'objet, à la fois en termes de robustesse et de précision, tout en générant des masques de segmentation précis.

PUBLICATIONS

The following publications are included in this thesis:

— Matthieu Paul et al. “Efficient video semantic segmentation with labels propagation and refinement.” In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2020, pp. 2873–2882

— Matthieu Paul et al. “Local Memory Attention for Fast Video Semantic Segmentation.” In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 1102–1109

— Matthieu Paul et al. “Robust visual tracking by segmentation.” In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*. Springer. 2022, pp. 571–588

Furthermore, the following publications were published during my PhD studies, but are not covered in this thesis:

— Christoph Mayer et al. “Adversarial feature distribution alignment for semi-supervised learning.” In: *Computer Vision and Image Understanding* 202 (2021), p. 103109

— Christoph Mayer et al. “Transforming Model Prediction for Tracking.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 8731–8740

— Matej Kristan et al. “The Tenth Visual Object Tracking VOT2022 Challenge Results.” In: *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VIII*. Springer. 2023, pp. 431–460

ACKNOWLEDGMENTS

First of all, I would like to thank both Prof. Dr. Luc Van Gool and Dr. Ralf Dragon for giving me the opportunity to pursue a PhD at the Computer Vision Lab while working at uniqFEED. This "hybrid" setup between academia and industry definitely came with its additional set of challenges compared to a more "traditional" setup. But I am grateful for that. I could deepen my knowledge of Computer Vision and Deep Learning, and directly apply new methods in a concrete product: Going from theory and reading research papers, to developing and deploying technological solutions that can reach millions of people.

A big part of my learning process came from all the interactions that I had with brilliant researchers at the lab. Among them, I was undoubtedly lucky to have Dr. Martin Danelljan as my second supervisor. I do believe that my doctoral studies would not have reached the same outcome without his support. What started as an initial succession of semi-random late lunches where we got to know each other, turned out to become a fruitful research collaboration. I learned a lot from him, both from the theoretical and practical facets of doing scientific research. From the start, he gave me ingenious advice that really helped me to move forward. He put a lot of trust in me despite my unusual work setup. His faith in me and motivating attitude, especially in the tough and doubtful moments, truly made a difference in staying focused. I honestly think that I could not have had a better supervisor than Martin, and I am happy to have him as a friend now.

From the lab, I would like to thank Dr. Radu Timofte for guiding me at the beginning of my stay at the lab and through my first publication. Also, Christoph, with whom I shared the usual struggles from the PhD student life early on: (deep) cleaning our office, climbing mountains and glaciers, and writing our first papers. Although my presence at the lab was limited due to my setup (and a global pandemic!), I got the chance to get acquainted with a couple of awesome people through the conferences, their related trips, and all the social events in between. Evan, his perpetual good mood and interest in board games; Erik, his taste in adventure and enthusiasm for sports; Patricia, her aversion to cold water and our always interesting conversations. They made the end of my PhD brighter, funnier and lighter. I am also glad for the good times I spent with all the others at the CVL.

From the company, everything transformed in a couple of years, from a start-up of less than 10 employees to more than 50 as I am writing these lines. It was quite an interesting journey that allowed me to expand my horizon to a lot of different personalities and talents, both in Computer Vision and in Broadcasting. This was extremely enriching, and I am thankful for everything I have learned with them. I want to particularly mention Dr. Angelo Martinovic, who has been there since the very beginning at uniqFEED. We shared a lot of good moments at the company, but also hiking and biking all around Switzerland. He often gave me helpful advice and insights during my PhD. His moral support throughout the years, as well as his energy and detailed proofreading skills, were of great help. Moreover, a special mention to Thomas, Louise, Tom, Martin, and Alex, the talented people with whom I have been sharing my daily work for years, also known as *The Tracking Team*. They are making it really enjoyable and stimulating to work together release after release. This environment truly helped me to push through tight deadlines both for the company and for my doctoral studies. I am sincerely grateful for that.

I would of course like to extend my gratitude to all my friends and flatmates with whom I had incredible times over the years. They all supported me in their own way and this work would have been for sure a much harder time without them. Countless hikes, bike rides, laser tag and paintball massacres, board games evenings and beach volleyball games were a great deal of fun. Flying above the Alps on a WW1 Grasshopper, jumping from a cliff with a hang-glider, or sailing in the Caribbean were incredible moments. Alberto's legendary laugh saved me during the toughest moments of lockdowns. Swimming inside a volcano and seeing the Northern Lights while exploring Iceland with Matthieu were unforgettable adventures. These were all welcome breaths of fresh air on the way to success, and reminders to not give up. Thank you everyone!

Finally, none of this would have been possible without the support of my family, all of them believed in my success and encouraged me throughout my doctorate. My brother Grégory especially believed in my potential, he was enthusiastic and supportive of this endeavor from day one. His passion and talent for scientific research were inspiring. My mother has been supporting and encouraging me since my decision to follow this path. She was always convinced that I would push through and eventually succeed. Her unwavering support helped me until the finish line.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Video footage represents a significant portion of all the data processed and consumed by everyone on the planet. This is nowadays possible and easy since cameras are essentially everywhere: smartphones, computers, drones, cars, security cameras, TV cameras, etc. Videos are used for a wide range of purposes: connecting people, sharing information and knowledge, autonomous driving and flying, surveillance, broadcasting sports, entertainment, etc. Considering only YouTube, one of the biggest online platforms for video consumption, several hundreds of hours of video are uploaded every minute and made available to billions of people across the globe. The total amount of video footage generated and used every day is probably much higher. Video data is at the core of our digital societies and is simply too big to be exploited by humans without technological means.

Video Understanding consists in automatically extracting information and meaning from videos. It is a vast umbrella under which fall many Computer Vision tasks such as *Visual Object Tracking*, *Video Object Segmentation*, *Visual Odometry*, *Simultaneous Localization And Mapping*, *Instance Semantic Segmentation*, *Object Segmentation*, *Object Detection*, *Body Pose Estimation*, etc. Each of these tasks aims at answering one or several of the following example questions: What is the video about? What is happening during the video and when? What are the different elements in the video, where are they and how do they interact with one another? What is the motion of the camera over time and how do the elements in the scene move?

Having powerful video understanding capabilities is in practice very useful when it comes to exploiting video footage beyond simply recording or watching. The tasks previously described find very concrete applications for different people and organizations. It can be used to organize, classify, index and search across a huge video database. It can be used to present content in a different way and highlight relevant information, for instance in sports with in-game augmentations to visualize strategies and actions, replays, creating best moments, highlights, etc. It can be used to bridge the gap between the physical and digital worlds, with Augmented Reality and Virtual Reality that can be used for instance to enhance remote assistance or work, from supporting factories to facilitating remote surgeries, enhancing teaching, etc. It can be used to alter the video for artistic purposes, entertainment, or even to create commercial value with virtual advertisement.

Video understanding comes with a lot of additional challenges when compared to single-frame computer vision tasks. They manifest themselves at different levels in a general and unconstrained setup. First, the scene content is dynamic, which means that elements in one image can, in the next images, move, get occluded, change in appearance and size, get out of the camera view, etc. Second, the camera itself can move and its intrinsic parameters can change, both of which can additionally induce motion blur, modify the size of objects, change the global illumination and drastically modify the appearance of the scene, change distortion, induce rolling shutter, etc. Third, a video can suffer from additional artifacts that can alter the quality of each individual image: it can be compressed with different codecs for storage or streaming, it can be interlaced for broadcasting, etc.

From a scientific and machine learning perspective, working with video data poses yet another problem that resides in the availability of annotated data for supervised training and testing. While datasets with single-frame annotations have grown larger over the years, densely annotated video data is far more scarce, especially when it comes to segmentation annotations. For instance, the Cityscapes [24] dataset has sequences with only a single full frame annotated per sequence for semantic segmentation. Other datasets [101, 130, 31] have dense annotations for video object segmentation, but the sequences remain relatively short and at best limited to a few objects annotated. The only alternative for supervised methods, while bigger datasets are being forged, is to rely on purely synthetic datasets like Sintel, GTA5, or SYNTHIA [11, 102, 104]. This however poses another set of issues when it comes to real data generalization [97, 20]. Fortunately, a growing number of self-supervised methods are emerging to deal with the lack of data available [14, 138, 147, 65, 126]. These are especially useful in the industry, where use cases are very specific and scalability is key.

One could consider a video sequence as a series of consecutive images captured by a camera and reason on a per-frame basis. From that perspective, video understanding would be an extension of image understanding to the temporal domain, where an algorithm extracts information from each frame separately. Although this “naive” approach provides a natural baseline, considering the whole video allows us to extract more information that might not be deduced from a single image, for instance, depth estimation, occlusion handling, camera motion estimation, analyzing dynamics of objects in the scene, etc.

This additional information potential is not the only advantage of considering the temporal dimension in videos. The consecutive frames are taken by the same sensor in a short period of time, therefore each image is highly correlated with its neighbors. This additional amount of redundant sensor data can be used in two major ways: to reduce the average computational intensity over several frames and to improve the quality of the information extracted.

The former consists in extracting semantics for a sparse set of frames instead of every frame. The assumption is that propagating information between similar frames comes at a lower computational cost and is easy since the relative changes between consecutive images are minimal. This can be used to optimize the overall computational cost of processing a video since the total cost will be lower than the combined sum of per-frame costs.

The latter exploits the high redundancy in the image content shared between neighboring frames. In the general case, both the camera and the scene can be dynamic, and the camera sensor itself can generate slightly different images from the same content due to noise or other factors. Performing tasks like object detection or segmentation on each frame separately can therefore easily lead to temporal inconsistencies, discontinuities, and flickering. When using temporal consistency, one can potentially hope to get better segmentation borders, disambiguate detection or segmentation parts and solve occlusions, by fusing the information from several frames.

In this thesis, we explore some of the aforementioned aspects through the scope of Semantic Segmentation, Video Object Segmentation (VOS), and Visual Object Tracking (VOT). In the first part, we analyze various ways of exploiting temporal information to boost either the performance or the accuracy of single-frame semantic segmentation methods. We particularly thrive towards lightweight and generic approaches that can be adapted to different segmentation methods easily. In the second part, we slightly shift our focus to VOS and its relation to VOT. We aim to reduce the existing gap between the two fields and try to highlight the benefits of tackling the VOT problem with segmentation masks at its core. We organize the thesis around three chapters.

Chapter 3 presents *Efficient Video Semantic Segmentation with Labels Propagation and Refinement* (EVS), a pipeline that focuses on maximizing efficiency and frame rate when performing semantic segmentation with videos. In that work, we propose to balance the computation load between the CPU and the GPU. A very fast optical flow runs on the CPU and the resulting motion vectors are used to propagate semantic labels and features across consecutive frames. At the same time, two neural networks operating on the GPU are responsible for either predicting dense semantic labels from scratch or refining existing predictions with the help of the propagated semantic features. We validate our approach on the Cityscapes [24] dataset using as a baseline a single-frame semantic segmentation network, showing comparable accuracy metrics compared to other methods. We suggest a set of operating points to use that range between 80 to 1000 Hz, depending on the desired trade-off between speed and accuracy.

Chapter 4 goes in the orthogonal direction and focuses on improving the quality of semantic segmentation masks when provided a video as input. We propose *Local Memory Attention for Fast Video Semantic Segmentation* (LMANet), a method that does not use an optical flow component to model changes across frames, but matches instead directly semantic features from different frames. In this work, we aggregate a rich representation of the semantic information in past frames into a memory that is queried and matched using attention mechanisms. A dedicated module makes sure to fuse the temporal cues from prior frames matched against the current one in a local neighborhood before the segmentation decoder makes the final prediction. Our approach, used on Cityscapes with two popular semantic segmentation networks ERFNet [103] and PSPNet [143], yields an improvement in segmentation performance by 1.7% and 2.1% in mean *Intersection Over Union* (IoU) respectively, while marginally increasing inference time.

Chapter 5 introduces *Robust Tracking by Segmentation* (RTS), a segmentation-centric tracking pipeline that works internally with segmentation masks instead of bounding boxes. Its unified architecture is therefore not only able to learn a better representation of the target but also outputs accurate segmentation masks without the need for additional segmentation components or post-processing steps. To make our method robust to challenging tracking scenarios, we design a dedicated instance localization component for a double purpose. It is used to condition the segmentation decoder to the right target when making the final segmentation prediction. It also enhances the online updating routine during inference, which helps prevent wrong semantic information to be stored in memory or used to update our model. To accommodate for the lack of segmentation masks in VOT benchmarks, we validate the quality of our segmentation masks on two popular VOS datasets. RTS proves to be competitive with other methods on multiple VOT benchmarks and establishes a new state of the art on the challenging LaSOT [35] dataset, with an *Area-Under-Curve* (AUC) score of 69.7%.

2 BACKGROUND

2.1 SEMANTIC SEGMENTATION

Image Semantic Segmentation consists in partitioning an image into semantically meaningful parts. Each part is taken from a set of predetermined classes relevant to the dataset or use case. For instance, in Figure 2.1 below, we would assign to each pixel of each image its corresponding label from a predefined list. For the football game: pitch, player, referee, ad boards, stairs, public, etc. For the autonomous driving case: road, sky, vegetation, car, building, pedestrian, advertisement, road sign, etc.

This is a core computer vision task that can be performed on a single image and is especially useful to get a high-level understanding of the content of an image. Of course, when extended to videos, it can be key to video scene understanding, for instance for autonomous driving and flying, commercial applications for productivity, entertainment, sports, etc. In particular, some works integrate the additional semantic knowledge as part of their pipeline to help with other tasks such as Structure from Motion (SfM) or SLAM [110, 100, 136], where parameters can be tuned based on different parts of the image, or additional constraints can improve results.

Figure 2.1 – Semantic Segmentation visually represented with colored overlays on the original images. Each semi-transparent colored overlay represents a different class. Each situation has a different set of pre-determined classes: on the left a football game and on the right a typical autonomous driving situation.

2.2 VIDEO OBJECT SEGMENTATION

In contrast with Semantic Segmentation, Video Object Segmentation (VOS) focuses on a specific target and generates a binary segmentation mask between the object of interest (*foreground*), and the rest of the scene (*background*), see figures 2.2 and 2.3. Moreover, the task in this case consists in segmenting that target object throughout a video, starting from an initial segmentation mask. The initial segmentation is propagated across the video frames and modified over time as the target changes, see Figure 2.3. It is pretty common for methods and datasets to track and segment several targets in the scene at the same time.

As the initialization of a target for a VOS pipeline is done with a segmentation mask, it takes a substantial amount of work building non-synthetic datasets, as each target pixel has to be annotated manually. Therefore, an active research field consists in providing an easier initialization with for instance weaker annotations [6, 141, 54], or *Referring VOS* [127, 8] that works with textual descriptions. Other approaches go further and can automatically find the most probable object of interest and segment it [14, 138, 147].

Figure 2.2 – Two examples of Video Object Segmentation visually represented with the help of a different semi-transparent colored overlay per target (*foreground*). Everything else in the scene is considered as *background*.

Figure 2.3 – A simple example of Video Object Segmentation captured 10 frames apart. During that time, the shape and position of the masks representing both the tennis player and the racket changed completely. One of the targets, the tennis ball, disappeared from the camera view.

2.3 VISUAL OBJECT TRACKING

Visual Object Tracking (VOT) solves a very similar problem compared to Video Object Segmentation (VOS). Given an object of interest in the initial frame, the goal is to track that object across consecutive frames. The difference, in this case, is that a bounding box is traditionally used to represent the target instead of a segmentation mask.

This representation is unfortunately very limiting for characterizing the target precisely unless it is convex and axis-aligned. Some methods or datasets might consider rotated bounding boxes to better fit the target [134, 34, 22]. Unfortunately, in many cases, the bounding box will cover as much background as foreground because the target will have thin parts or holes. In other cases, an accurate bounding box will even include occluding objects, and fail again to represent fully the target beyond its size and location. Figure 2.4 shows examples of these limitations.

In contrast with VOS, it is however much easier to annotate ground truth with bounding boxes. This partially explains why there is such a big gap between the nature of VOT and VOS datasets: Whereas VOS datasets are rather short and focus on challenging segmentation situations (bigger or thin objects, appearance changes), VOT datasets are longer and contain more challenging scenarios in terms of tracking such as small and/or fast targets, long occlusions, changes in the camera motion, etc.

Visual Object Tracking can be further split into *Single-Object Tracking* (SOT) and *Multiple-Objects Tracking* (MOT). The number of target objects in the MOT case is in general quite high, therefore it has its own methods [79, 119], benchmarks [86, 32], and metrics [108]. In this thesis, we place ourselves in the SOT case to facilitate the reasoning with VOS.

Figure 2.4 – In Visual Object Tracking, a target is defined by a bounding box. Simple targets like the volleyball (□) are very well defined by this representation. In contrast, more complex objects like the volleyball player or the bike (□) can only be represented well in terms of location in the image. A big part of the content inside each bounding box does not belong to the target object.

PART I

FROM IMAGE TO VIDEO SEMANTIC SEGMENTATION

3 EFFICIENT VIDEO SEMANTIC SEGMENTATION WITH LABELS PROPAGATION AND REFINEMENT

This chapter tackles the problem of real-time semantic segmentation of high-definition videos using a hybrid GPU-CPU approach. We propose an Efficient Video Segmentation (EVS) pipeline that combines:

(i) On the CPU, a very fast optical flow method, that is used to exploit the temporal aspect of the video and propagate semantic information from one frame to the next. It runs in parallel with the GPU.

(ii) On the GPU, two Convolutional Neural Networks: A main segmentation network that is used to predict dense semantic labels from scratch, and a Refiner that is designed to improve predictions from previous frames with the help of a fast Inconsistencies Attention Module (IAM). The latter can identify regions that cannot be propagated accurately.

We suggest several operating points depending on the desired frame rate and accuracy. Our pipeline achieves accuracy levels competitive to the existing real-time methods for semantic image segmentation (mIoU above 60%) while achieving much higher frame rates. On the popular Cityscapes dataset with high-resolution frames (2048×1024), the proposed operating points range from 80 to 1000 Hz on a single GPU and CPU.

3.1 INTRODUCTION

A lot of efforts have been made in semantic segmentation over the past years. Yet, while segmentation accuracy reached astonishing levels, little focus has been put on making it usable in real-time scenarios. Achieving very fast semantic segmentation would have many advantages, especially when used as an additional building block for other computer vision tasks related to real-time scene understanding. Particularly in the context of real-world scenarios for industrial or commercial cases such as augmented reality, autonomous driving, autonomous flying, etc.

Video scene understanding is already a wide and active research topic, especially in accurate object instances tracking and segmentation. However, in the context of real-time video semantic segmentation, fewer efforts have been put into exploiting the temporal information as a means to decrease inference time.

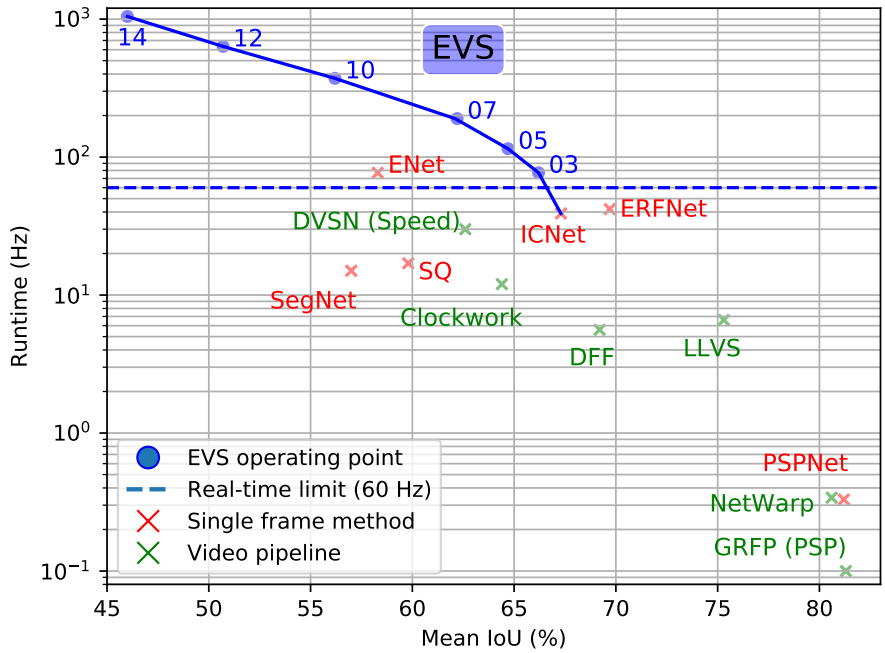


Figure 3.1 – Comparison between our EVS pipeline and state-of-the-art methods on the Cityscapes [24] dataset with input resolution 2048×1024. Table 3.2 provides more operating points and comparisons, from 1 Hz to 1000 Hz.

When used, this temporal aspect is in most methods used as additional information to improve either the accuracy of the predictions or their consistency over time, at the cost of additional runtime.

On the contrary, the focus of this chapter is to use temporal information as a way to minimize the inference time for each frame as much as possible, while limiting the drop in accuracy resulting from the reduced computations. The baseline we use runs at around 40 Hz on a frame resolution of 2048×1024. Our EVS pipeline defines several operating points among which the speedup factor varies from $\times 2$ to $\times 27$ on the same resolution.

The proposed pipeline uses ICNet [142] as the main prediction network since it is the current state of the art in terms of trade-off between accuracy and performance for single frame processing. To compute the dense optical flow, we use Dense Inverse Search (DIS) [61] as it is the current state of the art in terms of computational efficiency on the CPU. Dense optical flow plays a key role in our pipeline, as it can run on the CPU in parallel with the GPU at a much higher frame rate than the prediction network. This information is then used to:

- Warp the semantic information from one frame to the next, both high-level predictions and low-level contextual features. This warped semantics is used as input for the Refiner which will improve the prediction of the labels for the current frame.
- Feed the IAM to focus the refinement on regions where the optical flow is unreliable (typically thin and/or moving objects boundaries), by computing the forward-backward consistency of the propagated labels.

Contributions: Since semantic segmentation is crucial for video scene understanding, we aim to push the limits of this field through the following contributions, with a focus on efficiency and frame rate. Our contributions are the following:

First, our hybrid EVS pipeline balances the workload between GPU and CPU. They work in parallel, either computing semantic predictions or propagating them from frame to frame using optical flow, instead of having one large pipeline running fully on the GPU. Running the optical flow directly on the CPU decreases the workload on the GPU and leads to a massive reduction in computation time. Our goal is to establish new standards in terms of speed for real-time video semantic segmentation while preserving sound segmentation quality.

Furthermore, we introduce a fast IAM and a Refiner that work together to refine the propagated predictions of the main segmentation network to better match the current frame. Our versatile design allows running our pipeline in various operating modes, trading-off speed versus segmentation quality.

3.2 RELATED WORK

The most straightforward way to perform video semantic segmentation is to simply run image semantic segmentation on each frame. Although this approach is rather slow, it leads to a natural baseline to assess the quality of video segmentation methods. Furthermore, we review recent trends and ideas in video segmentation. As our proposed method combines semantic image segmentation with optical flow, we review different methods extracting optical flow between consecutive frames using traditional or deep learning-based methods.

3.2.1 *Image Semantic Segmentation*

Semantic image segmentation aims at assigning a class label to each pixel of a given image. The recent advances in deep learning [60, 128] lead to fast progress in semantic image segmentation [75, 74, 16]. Most of the state-of-the-

art methods [17, 143] are based on Fully Convolutional Networks (FCNs) [75]. Among these methods are: DeepLabV3+ [17], PSPNet [143] or more recently Panoptic FPN [57]. These methods concentrate mainly on high-quality segmentation masks that require a large number of parameters and are computationally intensive, *i.e.*inference time of around one second for a high-resolution frame (2048×1024).

Other methods that focus on reducing computing time and memory footprint obtain more and more attention: SegNet [1], SQ [114], ENet [93] and ESPNet [85].

Combining the best of both worlds, some methods aim at finding good trade-offs between frame rate and accuracy, either from their model (ERFNet [103] and ICNet [142]) or by treating differently *complex* and *simple* parts of the image (LC [71]). These methods achieve faster inference times while preserving a decent segmentation quality.

3.2.2 *Video Semantic Segmentation*

Compared to semantic image segmentation, developing dedicated video segmentation pipelines is a less explored research track. Applying image segmentation algorithms that operate on each video frame individually is possible. However, specialized methods for videos can exploit temporal information between consecutive frames to enable more reliable predictions or increase the frame rate.

Early methods tackling video segmentation were extending classical single image segmentation methods with temporally-aware components: normalized cuts [107], tracking [66] or motion segmentation [91].

Recent methods leverage dense optical flow in a more direct way by combining it with Gated Recurrent Units (GRUs) to refine the predictions and add temporal consistency [113, 90].

In particular, some methods aim at reducing inference times by embedding the temporal aspect in their structure by using LSTM [80], or by selecting keyframes to fully segment. Clockwork [106] authors observe that intermediate representations within a network change only slowly in most videos. Therefore, they propose to schedule features computation for keyframes only and share features in between. LLVS [72] and DVSN [131] try to further optimize scheduling depending on frame content.

Another family of methods uses the geometrical structure of the 3D scene to improve the segmentation quality. There, 3D point clouds obtained from visual odometry or stereo-vision approaches add additional information that allows more reliable predictions [10, 39, 105, 64].

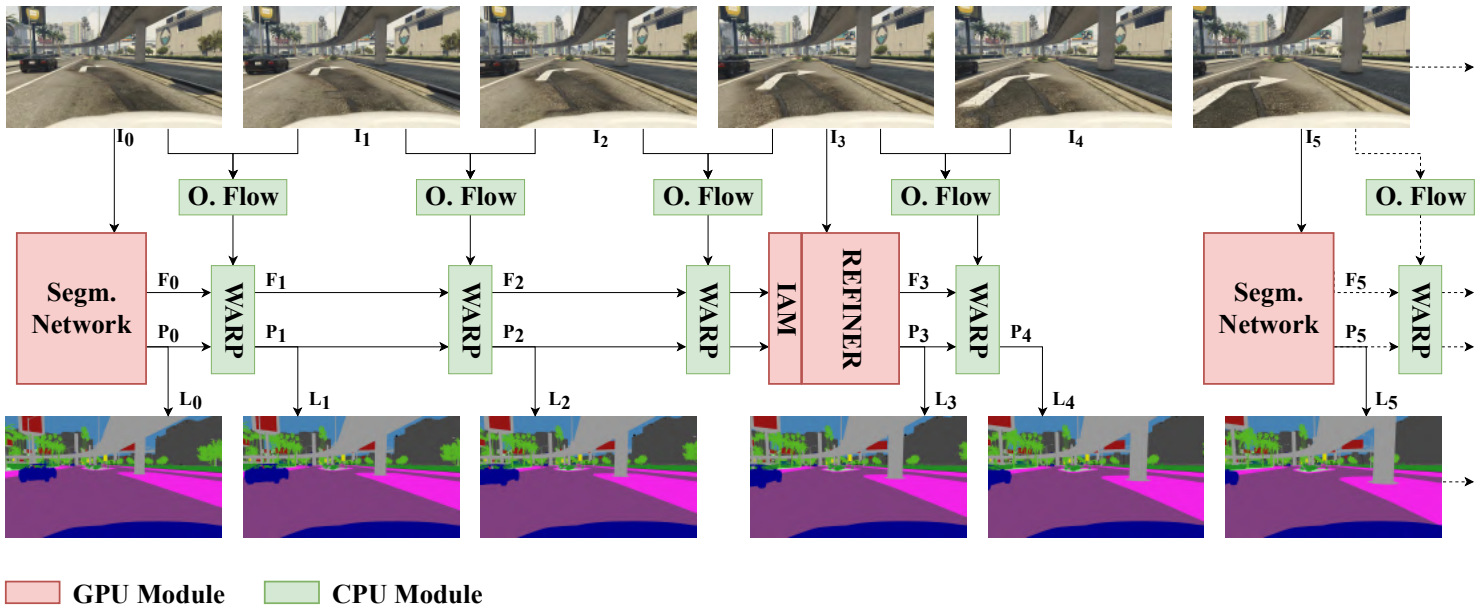


Figure 3.2 – Full pipeline overview with an input video stream $(\boldsymbol{I_0}, \boldsymbol{I_1}, \ldots)$ and the corresponding output labels $(\boldsymbol{L_0}, \boldsymbol{L_1}, \ldots)$. The predicted probabilities $\boldsymbol{P_i}$, labels $\boldsymbol{L_i}$ and deep features $\boldsymbol{F_i}$ are propagated with the corresponding dense optical flow.

One of the major challenges in video segmentation remains the massive amount of data that deep Convolutional Neural Networks (CNNs) require for training. Already, producing annotations for semantic image segmentation is costly. In this case, ensuring diversity in a video segmentation data set demands many different video sequences each consisting of hundreds of frames even for very short movies, leading to thousands of frames that should be annotated. Thus, existing data sets for video segmentation are either sparsely annotated [9, 43, 24], *i.e.*not every frame is labeled, or the segmentation task is simplified such that annotation is cheaper, *i.e.*single object segmentation [98]. In our case, we avoid this pitfall by relying on a network that is trained on single images. Only synthetic data sets such as GTA5 [102] or Sintel [11] overcome that annotation limitation.

3.2.3 *Optical Flow*

Traditional optical flows such as Lucas-Kanade or Gunnar-Farneback [76, 37], recently started to compete with new deep learning approaches: FlowNet [33, 53], MPNet [112] and SegFlow [23] produce very accurate flow estimates, but are rather expensive and slow and run on the GPU. As a result, deep video semantic segmentation pipelines using optical flow usually improve marginally their accuracy or temporal consistency, while increasing substantially their inference time: NetWarp [41], GRFP [90] or DFF [145].

In contrast, when aiming at fast and efficient video segmentation, DIS [61] is among the most suitable candidates. DIS achieves much higher frame rates than deep optical flow methods and runs on CPU, which gives more flexibility to choose between speed and accuracy by selecting different operating points.

3.3 EFFICIENT VIDEO SEGMENTATION PIPELINE

3.3.1 *Full Pipeline Overview*

Our pipeline consists of five main components that process the video stream jointly, see Figure 3.2. The GPU holds a segmentation network and a Refiner with IAM, while the CPU is responsible for computing in parallel the optical flow and for warping the CNN features and predictions.

The dense optical flow is computed for each pair of consecutive frames. It enables the forward and backward remapping of semantic information extracted by the deep networks. The IAM is responsible for computing the inconsistencies that remapping reveals. It provides this information to the Refiner, which then

corrects mistakes caused by warping around inconsistent areas, i.e. where the optical flow is not reliable.

In the best case, the flow will be consistent and the prediction of the next frame will simply be the previous prediction warped by using optical flow. In most cases, the lack of flow consistency in some regions of the image (sudden changes in brightness, occlusions, multiple fast motions, etc.) will be recovered by the Refiner, while the other prediction of other regions will still be derived from the previous prediction to increase temporal consistency.

3.3.2 *Semantic Segmentation Network*

The segmentation network in our pipeline is responsible for providing a full-frame semantic segmentation. We want to emphasize that any deep framework can be used within our framework, leaving space for improvements when better networks are developed. For this work, we choose to use ICNet [142] because of its excellent trade-off between accuracy and speed: 67% mIoU at ~ 40 Hz on the popular Cityscapes [24] dataset.

3.3.3 *Optical Flow and Semantics Propagation*

The advent of deep learning brought many optical flow methods to impressive quality levels while focusing less on computational efficiency. However, we require a fast but still accurate dense optical flow. DIS Flow [61] matches perfectly this requirement and has the advantage of producing a reasonable dense flow at a very high frame rate while running on the CPU. Thus, it allows to save the GPU resources for other tasks.

Dense optical flow provides for each pixel (x, y) of the image a flow in each dimension $F_{xy}^{1\to2}(x, y)$, between two consecutive frames I_1 and I_2. The mapping between I_1 and I_2 can be written for each dimension as follows:

$$\begin{aligned} M_x(x, y) &= I_2(x, y) - F_x^{1\to2}(x, y) \\ M_y(x, y) &= I_2(x, y) - F_y^{1\to2}(x, y) \end{aligned} \tag{3.1}$$

Using Eq. (3.1) then allows to produce image I_2 solely by remapping the pixels from image I_1.

For non-integer valued coordinates, using the nearest neighbors interpolation results in a valid remapped image:

$$I_2(x, y) = I_1(M_x(x, y), M_y(x, y)) \tag{3.2}$$

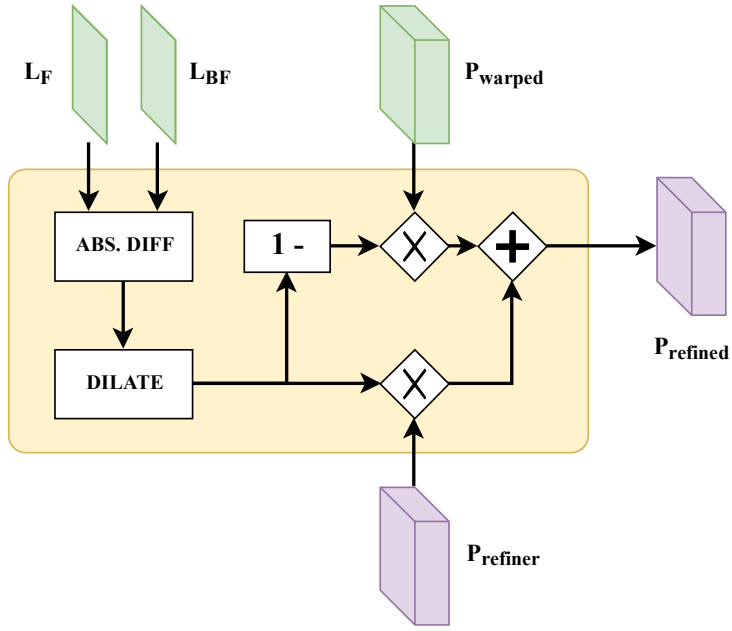


Figure 3.3 – Inconsistencies Attention Module.

We want to emphasize that such a mapping is fast to perform because it is highly parallelizable on the CPU. In our pipeline, it is used to quickly remap both the predictions and the low-level CNN features from one frame to the next. These features represent the slow-changing contextual information of the scene. The predictions can also be remapped backward such that the IAM is able to compute the inconsistencies (Figure 3.3).

3.3.4 *Inconsistencies Attention Module*

The IAM is working together with the Refiner. It is designed such that it is lightweight and able to focus the attention of the refinement on regions where the optical flow is inconsistent. The inputs are:

- L_{F}: the labels predicted for the current frame, obtained by warping the previous frame labels forward.
- L_{BF}: the labels predicted for the current frame, obtained by warping the labels backward and then forward L_F.
- P_{refiner}: the predicted probabilities for each class by the Refiner.
- P_{warped}: the predicted probabilities warped using the optical flow.

As a first step, the module computes a probability map M_i that represents the forward-backward inconsistencies of the optical flow for the given input frames. For every pixel (m, n) where L_{F} and L_{BF} are different, the probability is considered to be maximal because the flow is unreliable. All other pixels are considered to be reliable:

$$M_i(m,n) = \begin{cases} 1.0 & \text{if } L_{\text{F}}(m,n) \neq L_{\text{BF}}(m,n) \\ 0.0 & \text{otherwise} \end{cases} \tag{3.3}$$

As a second step, this binary mask is dilated and smoothed to engulf the surrounding areas of the inconsistencies and to let the Refiner act on them, as the predictions in these regions are more likely to be wrongly propagated by the optical flow.

Finally, the predicted probabilities for each pixel are weighted differently between the warped prediction and the prediction of the Refiner. If P_{refiner} is the prediction of the Refiner and P_{warped} is the previous prediction warped to the current frame using the optical flow, the final refined prediction P_{refined} is defined as the sum of the Hadamard products:

$$P_{\text{refined}} = M_i \circ P_{\text{refiner}} + (1 - M_i) \circ P_{\text{warped}} \tag{3.4}$$

As shown in Figure 3.3, the module only consists of lightweight operations for a GPU, especially since the inputs and outputs are processed at a resolution of 512×256.

3.3.5 *Refiner*

A carefully performed benchmark of the branches in the ICNet architecture shows that even though the network is designed to limit the heavy computations on the lowest resolution to limit the inference time per frame, almost half of that time is spent only on building low level features (see Figure 3.5). The Refiner is built on the idea that these low level features do not need to be recomputed every frame in the context of a continuous video stream: due to their resolution, they are changing at the slowest rate over time.

Its task is different from the segmentation network: given pre-aligned low-level features from past frames, the Refiner should only compute the higher-level features for the new frame, making it shallower. This allows us to spare half of the computations that would then otherwise be carried out to extract low-level features.

Besides, with the help of the IAM, this refinement is focused only on some areas of the image (see Figure 3.4). Using the dense optical flow is reliable

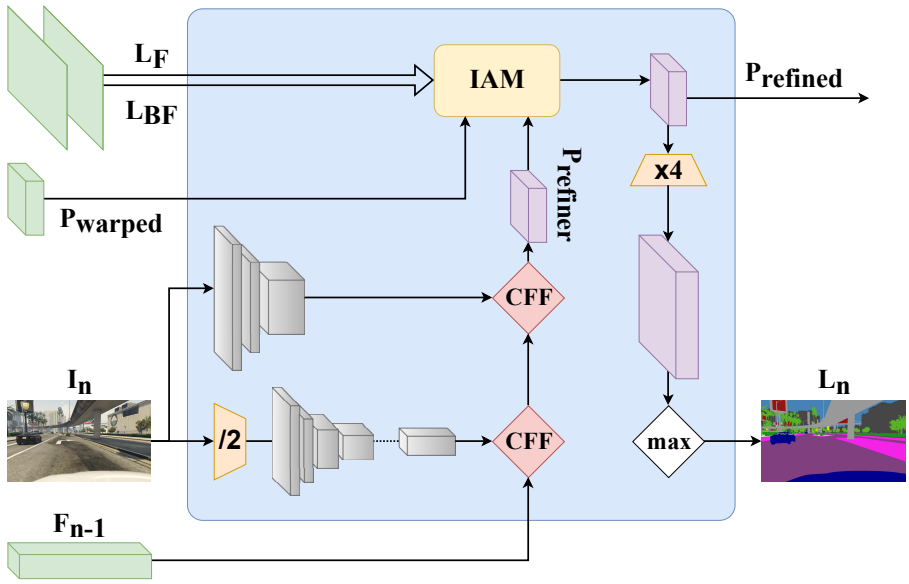


Figure 3.4 – Refiner architecture. For a given input image $\boldsymbol{I_n}$ and warped semantics from the previous frames (green), it generates refined probabilities $\boldsymbol{P}_{\text{refined}}$ and labels $\boldsymbol{L_n}$. CFF stands for "Cascade Feature Fusion", as in ICNet [142].

for large portions of the image but it causes errors next to object boundaries, especially when these objects are thin, moving, or new. Thus, using the IAM leads to a better temporal consistency overall, as most of the predicted labels were propagated from one frame to the next.

3.4 EXPERIMENTAL EVALUATION

3.4.1 *Setup and Benchmarking Method*

All the benchmarks are done using a single Nvidia Titan Xp GPU, and an Intel Core i7-5930K CPU @ 3.50GHz. The implementation is different from the original ICNet implementation which is written in Caffe and uses a proprietary version of ResNet50. Instead, we use an equivalent implementation in Tensorflow 1.8 and CUDNN 7.1 as the baseline for this chapter. The Tensorflow implementation yields almost the same performance and accuracy (67.3% vs. 67.7%). All the benchmarks and comparisons in this chapter use this Tensorflow implementation. It is worth noting that this is not problematic because the segmentation network of our pipeline can be replaced by any other implementation.

All the following benchmarks and results are produced on Cityscapes [24], which contains short video snippets of 30 frames at a high resolution (2048×1024) among which the 20^{th} frame contains a fully annotated ground truth mask. All the experiments and results presented are evaluated on the 20^{th} frame with different starting points before it depending on the operating points.

For us, it is important to measure the computation times on the GPU as accurately as possible. Thus, we build a specific probe class based on the publicly available Tensorflow Profiler, which provides the detailed timestamps for each operation on the GPU in JSON format. This data allows us to establish very accurate timings for each part of the network.

Each measurement contains 300 samples from the extracted profiler data. In order to avoid border effects, we measure each sample in the middle of the execution of the network. The measurements show that the timings are more varied at the startup time and the initialization of the models. Nonetheless, following the aforementioned strategy leads to reliable and accurate GPU computation times: the average and median measurements are matching with a small standard deviation, see Figure 3.5.

3.4.2 *Runtime of the different components*

On the CPU side, warping pixels from one frame to the next using optical flow can be easily parallelized on the CPU. Once split in a 3×3 or 4×4 grid, all the pixels from a full frame are remapped within a marginal time period (~ 0.15 ms on an Intel Core i7-5930K CPU @ 3.50GHz).

On the GPU side, we have two models: one for the full CNN and the other for the Refiner. ICNet [142] is structured around 3 branches: *Branch1* with very few convolutions operating at full resolution, *Branch2* that computes deep features starting from half the resolution, and *Branch4* that goes much deeper at even lower resolution. Figure 3.5 shows a speedup of almost two times for the inference time of the Refiner.

3.4.3 *Optical Flow and Labels Propagation*

3.4.3.1 *Optical Flow Benchmark*

Several operating points are suggested for DIS [61], with a set of parameters that trade off accuracy and runtime. For our experiments, we choose a set of parameters to achieve a small runtime: no variational refinement, finest scale $\theta_f = 2$, patch size $\theta_{ps} = 8$, gradient descent iterations $\theta_{it} = 12$. This allows us to run the optical flow computation on one of the cores of the CPU, on the

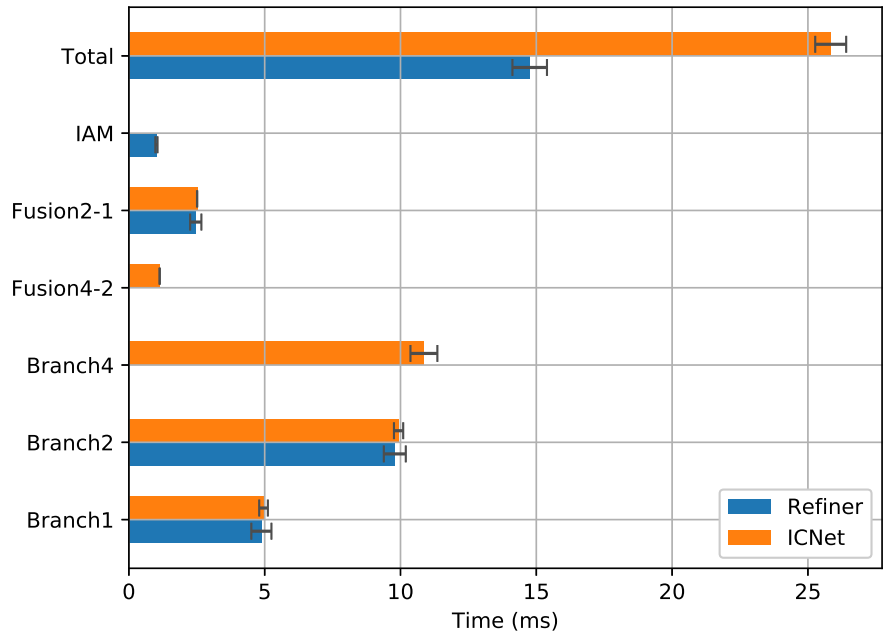


Figure 3.5 – Runtime of the Refiner with IAM compared to ICNet on a single Nvidia Titan Xp.

full frame resolution 2048×1024 in less than 5 ms. The goal is to compute the optical flow for five frames on one core, while the segmentation network is working ($\sim$ 25 ms, see Figure 3.5).

3.4.3.2 *Influence of the Optical Flow Algorithm*

The dense optical flow computation is of paramount importance to propagate the semantic information correctly. Figure 3.6 shows the comparison between DIS [61] at a fast operating point and Gunnar-Farneback [37] with a 2 layers pyramid, an averaging window of 9 pixels and 15 iterations. There is a substantial difference in the mIoU already after the first propagation.

Experiments with higher quality settings for DIS [61] showed marginal improvements (below 0.2%) on the mIoU even at high resolution, which motivated our choice for a faster operating point. With the ultra-fast setting, the drop per propagation on the highest resolution is between 1.0% and 1.5% (1.2% on average). This drop also tends to decrease when the resolution decreases, which is particularly interesting for the predictions and low-level forward propagation of the features since they operate at a resolution of 512×256 and 128×64 respectively.

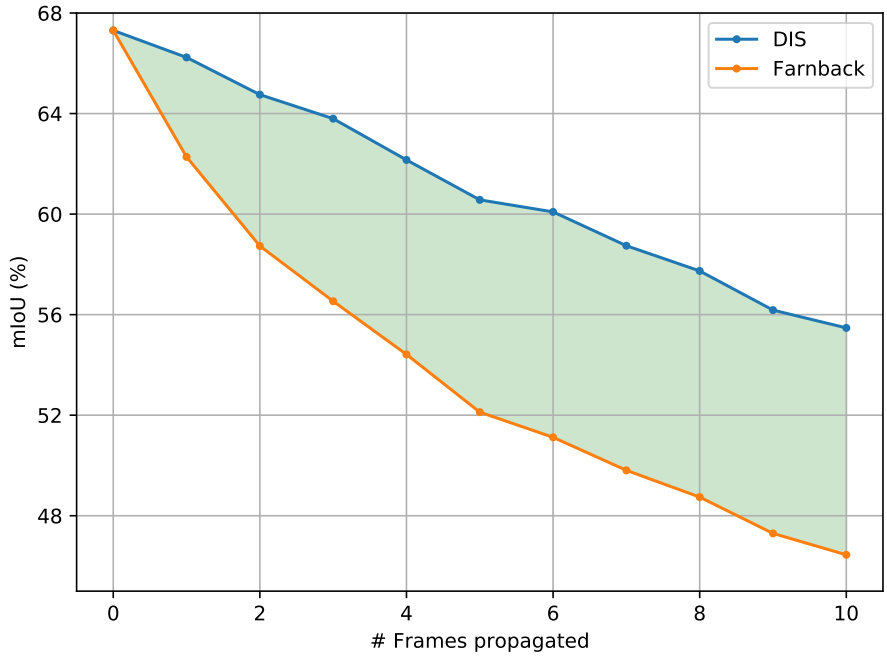


Figure 3.6 – Influence of a pure label propagation on the mIoU for 2048×1024, comparison Gunnar-Farnebäck vs. DIS.

3.4.3.3 *Uncertainties across the Evaluation Set*

The inconsistencies detected by forward-backward warping of the labels with the optical flow vary depending on the frame content and increase globally after each propagation. Figure 3.7 shows on the evaluation set of CityScapes [24] how the uncertainties are distributed depending on the number of propagations. This shows that even after 4 propagations, less than 5% of the flow computed is detected as inconsistent on average. For frame-to-frame propagation, this drops to less than 1%, which confirms that the optical flow is highly consistent.

3.4.4 *EVS Operating Points*

3.4.4.1 *Per Class Impact of Warping and Refinement*

As discussed before, a simple forward mapping of the predictions made by the segmentation network can bring an important speedup factor, at a cost of an overall marginally degraded segmentation quality. Although the drop in mIoU per propagation might seem marginal, it is directly correlated to the mistakes of

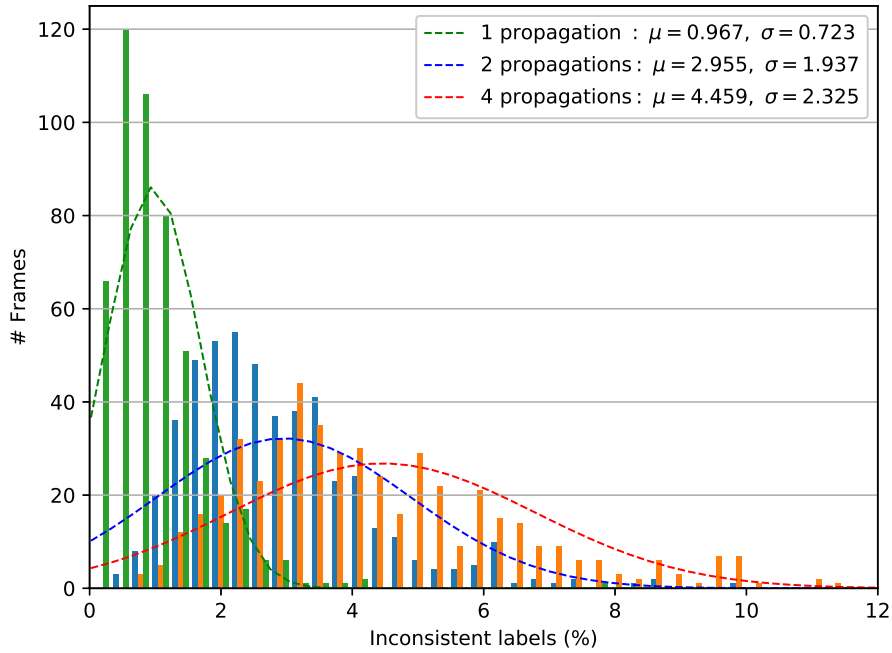


Figure 3.7 – Re-partition of frames on the Cityscapes [24] evaluation set as a function of their percentage of forward-backward inconsistent pixels from the optical flow, after 1, 2 and 4 propagations.

the optical flow (especially around boundaries of moving objects, thin objects and occlusions that may happen over time) and might have a big impact locally.

A per-class analysis (see Table 3.1) confirms that the errors due to optical flow propagation affect most classes only marginally (below 1% drop in IoU). Some of them are particularly affected by the wrong labeling: static thin objects (poles, traffic signs, traffic lights) and humans/small moving objects (person, rider, bike) are the most impacted classes (between 1% and 5% drop in IoU).

The Refiner is able to recover a large portion of the drop in IoU observed for those classes, especially after 1 frame propagation for poles, street signs and persons, even though the overall IoU gain is between 0.5% and 1%. Figure 3.8 shows these typical situations where the refinement has a clear visible impact on these specific classes and shows that our Refiner can recover missing parts:

- Thin objects such as poles, street signs, or traffic lights are not always captured or heavily distorted by the camera motion.
- Pedestrians on a crossing or cyclists on bikes are sometimes difficult for the optical flow to fully capture.

Method	Total	road	swalk	build.	wall	fence	pole	tlight	sign	veg.	terrain	sky	person	rider	car	truck	bus	train	mbike	bike
Baseline	67.3	97.4	79.5	89.4	49.1	51.7	46.1	47.9	61.1	90.3	58.6	93.4	69.9	43.3	91.4	64.8	75.8	59.9	43.9	65.4
EVS 03	66.2	97.3	78.9	89.1	51.1	52.1	41.5	46.6	60.4	89.8	59.2	93.1	66.9	41.7	90.5	64.1	75.2	52.1	44.5	64.2
EVS 02	66.8	97.3	79.1	89.3	51.0	52.3	44.5	47.1	61.4	90.2	59.5	93.2	69.0	42.5	90.9	63.9	75.5	52.5	45.0	64.9
Recovery	**+0.6**△	=	+0.2	+0.2	-0.1	+0.2	**+3.0**	**+0.5**	**+1.0**	+0.4	+0.3	+0.1	**+2.1**	**+0.8**	+0.4	-0.2	+0.3	+0.4	+0.5	**+0.7**
EVS 07	62.2	96.8	77.1	87.4	50.6	49.5	31.3	43.5	55.5	87.9	55.9	92.6	57.9	35.4	87.2	59.8	72.0	41.8	40.3	58.5
EVS 06	63.0	96.6	75.6	87.9	50.2	49.8	36.0	44.4	57.5	89.1	56.6	93.2	63.6	36.5	87.5	59.2	70.8	41.5	41.1	60.0
Recovery	**+0.8**△	-0.2	-1.5	+0.5	-0.4	+0.3	**+4.7**	**+0.9**	**+2.0**	+1.2	+0.7	+0.6	**+5.7**	**+1.1**	+0.3	-0.6	-1.2	-0.3	+0.8	**+1.5**

Table 3.1 – Per-class results on Cityscapes after propagating 1 or 4 times the labels with refinement (EVS 02 and EVS 06) or without (EVS 03 and EVS 07). The corresponding recovery achieved by the Refiner is explicitly mentioned for both cases.

- Missing parts due to occlusion and moving objects: a cyclist and bike passing in front of vegetation or two cars at a crossing.

Interestingly, the analysis also reveals that large static classes benefit from propagation (wall, fence, terrain) such that the IoU for these classes is higher than the IoU produced by the baseline, even without refinement (between 0.5% and 2% gain in IoU).

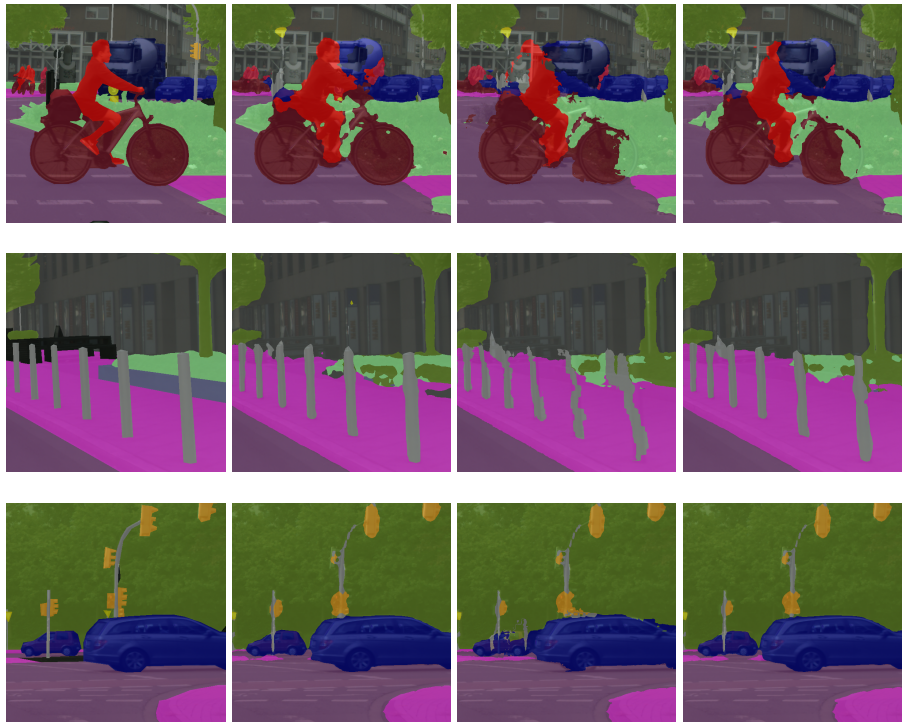
3.4.4.2 *Operating Point Comparison*

The structure of our EVS pipeline is defined by four parameters: the downscaling factor used by the segmentation network (**D**), the rate (every n^{th} frame) at which the full segmentation is computed (**S**), warping (**W**) and refinement (**R**). Table 3.2 summarizes these operating points and compares them with state-of-the-art methods in terms of accuracy, frame rate and speedup factor compared to our baseline ICNet [142].

3.5 CONCLUSION

In this work, we introduce an Efficient Video Segmentation pipeline that pushes the boundaries of real-time video semantic segmentation in terms of computational efficiency by combining the benefits of deep CNNs running on the GPU and a very fast optical flow running in parallel with the CPU. We propose different operating modes in order to focus either on frame rate or accuracy, from 67% mIoU at $\sim$ 40 Hz to 46% mIoU at $\sim$ 1000 Hz for 2048×1024 input images.

To compensate for the introduced errors in the predictions by the optical flow propagation around thin and moving objects (poles, persons, bikes, etc.), we propose a Refiner network to correct some errors and to generate a visually more appealing segmentation. The Refiner works with a dedicated Inconsistencies Attention Module which focuses the prediction refinement on the relevant regions of the image.



(a) Ground truth (b) ICNet (c) EVS Propagated (d) EVS P + Refined

Figure 3.8 – Benefits of the Refiner on the propagated predictions in three problematic situations for the optical flow: a person riding a bike, thin poles on the sidewalk, and occlusions from cars.

One of the strengths of our pipeline is that the segmentation network can be used as a black box method and can be replaced with any other segmentation network, bringing potentially more accuracy for the same speedups in the future. Moreover, our pipeline could benefit from a higher input frame rate because two consecutive frames are more similar and lead to a more accurate and reliable optical flow prediction (Cityscapes has a relatively small frame rate of 17 Hz). Furthermore, the IAM introduced in this chapter could be used in future works as a way to dynamically adapt the behavior of our pipeline depending on the input frames. In simple situations, the segmentation network and the Refiner could run less often such that the whole pipeline relies more on the optical flow when it is reliable. In more complex situations, the pipeline would then be able to force the re-segmentation more often to preserve a reasonable accuracy at the price of a lower frame rate.

	Method	Framerate	Speedup	D	S	W	R	mIoU
Video pipeline	EVS 14 (Ours)	1045 Hz	×27.1	0.5	17	✓	✗	46.0%
	EVS 13 (Ours)	677 Hz	×17.6	0.5	10	✗	✗	35.3%
	EVS 12 (Ours)	634 Hz	×16.5	0.5	10	✓	✗	50.7%
	EVS 11 (Ours)	387 Hz	×10.1	1.0	10	✗	✗	36.7%
	EVS 10 (Ours)	372 Hz	×9.7	1.0	10	✓	✗	56.2%
	EVS 09 (Ours)	339 Hz	×8.8	0.5	5	✗	✗	42.8%
	EVS 08 (Ours)	192 Hz	×5.0	1.0	5	✗	✗	46.4%
	EVS 07 (Ours)	190 Hz	×4.9	1.0	5	✓	✗	62.2%
	EVS 06 (Ours)	122 Hz	×3.2	1.0	5	✓	✓	63.0%
	EVS 05 (Ours)	115 Hz	×3.0	1.0	3	✓	✗	64.7%
	EVS 04 (Ours)	74 Hz	×1.9	1.0	3	✓	✓	65.6%
	EVS 03 (Ours)	77 Hz	×2.0	1.0	2	✓	✗	66.2%
	EVS 02 (Ours)	49 Hz	×1.2	1.0	2	✓	✓	66.8%
	EVS 01 (Ours)	37 Hz	×0.95	1.0	1	✓	✓	67.6%
	DVSN [131]	30 Hz	×0.8	-	-	-	-	62.6%
	Clockwork [106]	12 Hz	×0.3	-	-	-	-	64.4%
	LLVS [72]	6.6 Hz	×0.2	-	-	-	-	75.3%
	DFF [145]	5.6 Hz	×0.1	-	-	-	-	69.2%
	GRFP [90]	0.6 Hz	×0.02	-	-	-	-	81.3%
	NetWarp [41]	0.3 Hz	×0.01	-	-	-	-	80.6%
Single frame	ENet [93]	77 Hz	×1.9	-	-	-	-	58.3%
	ERFNet [103]	42 Hz	×1.1	-	-	-	-	69.7%
	ICNet[142]	39 Hz	Ref.–	-	-	-	-	67.3%
	SQ [114]	17 Hz	×0.4	-	-	-	-	59.8%
	SegNet [1]	15 Hz	×0.4	-	-	-	-	57.0%
	PSPNet [143]	0.8 Hz	×0.02	-	-	-	-	81.2%

Table 3.2 – Comparison of different EVS pipeline operating points. Numbers are reported on a Nvidia Titan Xp GPU and Intel Core i7-5930K CPU @3.50GHz for our pipeline and the reproduced ICNet [142]. Numbers for other methods are reported from their respective papers on various hardware.

4 LOCAL MEMORY ATTENTION FOR FAST VIDEO SEMANTIC SEGMENTATION

In this chapter, we propose a novel neural network module that transforms an existing single-frame semantic segmentation model into a video semantic segmentation pipeline. In contrast to prior works, we strive towards a simple, fast, and general module that can be integrated into virtually any single-frame architecture. Our approach aggregates a rich representation of the semantic information in past frames into a memory module. Information stored in the memory is then accessed through an attention mechanism.

In contrast to previous memory-based approaches, we propose a fast local attention layer, providing temporal appearance cues in the local region of prior frames. We further fuse these cues with an encoding of the current frame through a second attention-based module. The segmentation decoder processes the fused representation to predict the final semantic segmentation.

We integrate our approach into two popular semantic segmentation networks: ERFNet and PSPNet. We observe an improvement in segmentation performance on Cityscapes by 1.7% and 2.1% in mIoU respectively, while increasing the inference time of ERFNet by only 1.5ms.

4.1 INTRODUCTION

Semantic segmentation is one of the core computer vision tasks that paves the way toward scene understanding. It entails assigning a label to each pixel of an image, where a label generally represents an element in a predefined set of *classes*. It is useful in a growing number of applications, including augmented reality, surveillance, and robotics (autonomous cars and drones), where scene understanding is key to building better and safer systems. In these scenarios, however, the input to the vision pipeline most often consists of a video stream, and not only a single frame. It is natural to exploit the temporal dimension instead of processing each input frame independently. In this work, we therefore address the problem of *video* semantic segmentation.

Considering a sequential input instead of a single frame can be beneficial to solve the semantic segmentation task, as we can leverage the additional temporal dimension. Indeed, in most cases, one can reasonably assume a certain level of temporal consistency from one frame to the next. This temporal cue has the potential to reduce inference time by avoiding redundant computations,

improving the accuracy of the prediction, or finding a better trade-off between speed and accuracy. In practice, however, exploiting video data is a very challenging problem. Introducing the temporal dimension can easily lead to inertia, which can be problematic for dynamic content. In particular, the model should avoid propagating errors over time and allow for rapid changes in the scene produced by events such as occlusions or new objects. Efficiency is also a concern with video architectures, as they need to run fast enough and require a lot of data to train and evaluate.

In comparison to its single-frame counterpart, video semantic segmentation has received much less attention despite its potential advantages. Existing approaches exploit video input by using either dense optical flow directly [131, 145, 96, 41] or by combining it with recurrent neural networks (RNNs) [113, 90]. However, these often lead to marginal improvements relative to the added complexity and increased frame rates of those architectures. In contrast, methods trying to reduce complexity and inference time by avoiding redundancies and propagating information between frames usually have unsatisfactory accuracy compared to their baselines [96, 72, 131]. In this work, we set out to address these issues.

We introduce a novel neural network architecture to convert single-frame semantic segmentation models into video pipelines. Our network consists of three components. Following the recent success of attention-based approaches [92, 69, 73, 50, 49], we employ a *Memory module* that aggregates semantic information from several past frames. It holds a rich representation of the past and is updated over time with incoming frames. Contrary to previous attention-based alternatives [69, 50, 49], we propose a local attention mechanism. It ensures efficient reading from the memory, given the current Query frame, while preserving the segmentation performance of the standard dot-product attention [117]. We combine the extracted memory representation with features from the Query frame itself with an attention-based *Fusion module*. The fused representation serves as input for the decoder to make the final prediction. Our approach does not rely on optical flow or other expensive operations and thus has a minimal impact on inference time. We integrate our approach into two popular semantic segmentation architectures, namely ERFNet [103] and PSPNet [143]. With ERFNet, our approach achieves a 1.67% increase in mIoU on Cityscapes [24] while only increasing inference time by 1.5ms.

4.2 RELATED WORK

The simplest way of tackling video semantic segmentation is to perform semantic segmentation independently in each frame. Although this approach ignores

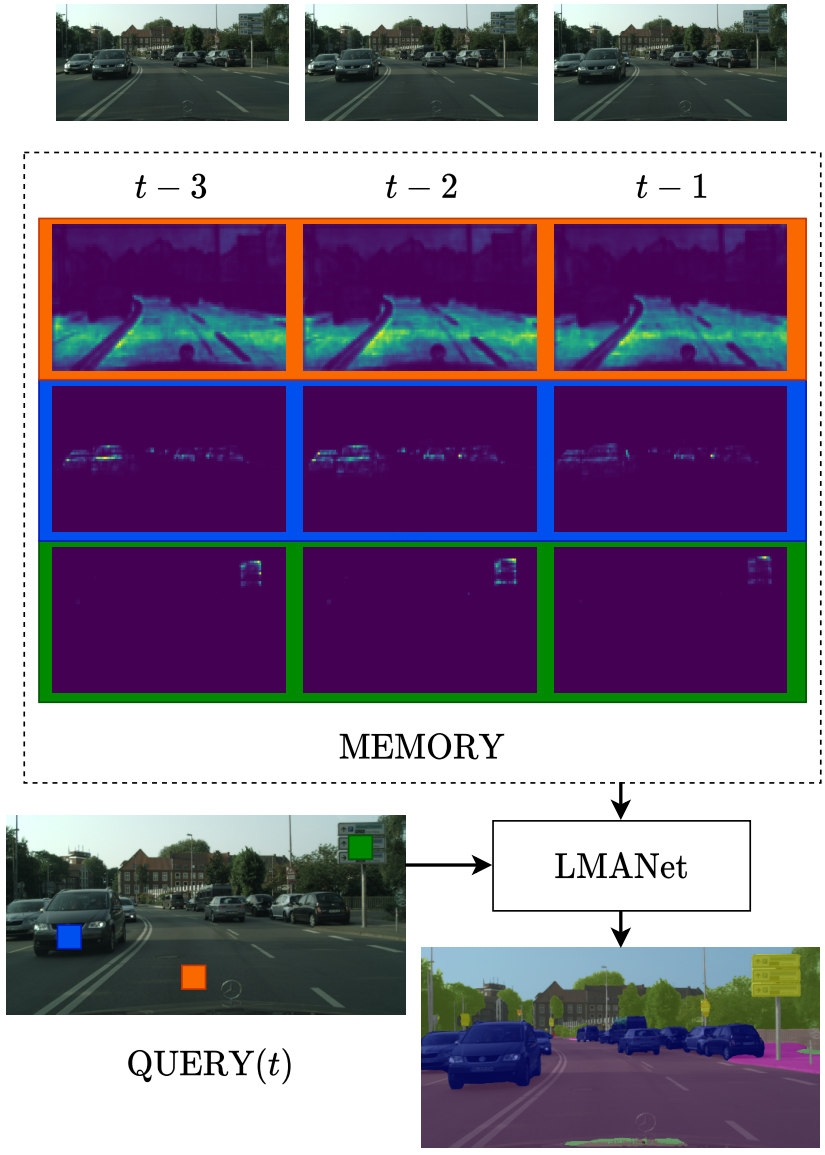


Figure 4.1 – We propose LMANet for fast video semantic segmentation. The Memory (top) consists of deep features extracted from previous frames. Features of the current frame (Query) are matched against the frames in the Memory to generate memory attention maps for each query location. Three example locations (orange, blue, green) are visualized with their respective attention maps in Memory. Our approach can thus efficiently integrate information from past frames to predict a final segmentation output.

the temporal dimension of the input, it provides a natural baseline to assess the performance of video segmentation methods. Dedicated video semantic segmentation pipelines have been receiving more attention in recent years. However, generating accurate dense annotations for video semantic segmentation is costly. Only synthetic data sets such as GTA5 [102] or Sintel [11] provide large-scale densely annotated training and test sets. Thus, most real-world video data sets are sparsely annotated [9, 43, 24]. Consequently, most video semantic segmentation approaches [80, 106, 72, 131, 96, 145, 113, 90, 41] evaluate their results based on a single frame annotated per sequence. Their basis is a single-frame classifier, such as a random forest or a CNN. Temporal consistency is introduced by considering consecutive predictions or propagating semantic information. The goal is to either improve the segmentation masks or to decrease the average inference time.

Some methods address video semantic segmentation by looking at the problem from a broader angle: using tracking [66], motion segmentation [91], or even exploiting the 3D structure of the scene. The latter usually rely on 3D point clouds to improve 2D segmentation predictions [10, 38, 63] which is computationally expensive and potentially error-prone. Other methods focusing on improving the segmented outputs model the temporal consistency by introducing inter-frame and intra-frame pixel connections in large graphs structures, mostly relying on 2D or 3D conditional random fields across the video inputs [15, 89, 62, 115].

Propagating semantic information and features was explored in different ways, for instance by using long short-term memorys (LSTMs) [80] to embed the temporal aspect in the pipeline structure or scheduling keyframes to fully segment and propagate semantic information in between [106, 72, 131, 96]. These methods in general offer different operating points leading to different trade-offs between efficiency and accuracy.

One of the popular ways to propagate information between frames is to explicitly use dense optical flow. This is usually done with two different objectives in mind. First, it may be used to reduce the average inference time. The authors of [131, 145] use state-of-the-art GPU methods [33, 53], adding to the complexity of their pipeline. In [96], the authors suggest a simpler approach relying on a fast optical flow running on CPU [61] to propagate semantics across frames and achieve higher frame rates. However, all of these methods reduce the segmentation accuracy. The second objective of using optical flow is to improve the baseline accuracy. For instance, some pipelines learn temporal consistency between consecutive feature maps with the help of gated recurrent units (GRUs) [113, 90]. Others warp features directly at different depths of their baseline [41]. Unfortunately, these frameworks have relatively small gains over

their baselines, and the added complex and slow optical flow structure around them makes them impractical to use in real-time scenarios.

The transformer model [117] was introduced as an alternative to RNNs, such as LSTMs and GRUs. Transformer networks and self-attention are mostly used in the field of Natural Language Processing but are rapidly gaining interest in computer vision topics such as image recognition [48], object detection [13], video object segmentation [92] or semantic segmentation [73].

Attention mechanisms were recently used for video semantic segmentation to leverage temporal information and improve both intra-frame and inter-frame semantic information. The authors of [69] introduce two attention mechanisms to combine features while others embed the attention mechanism in the backbone over several frames [50, 49]. In contrast, we aim at a simpler architecture and integrate a local attention mechanism to ensure efficiency.

4.3 PROPOSED METHOD

We propose Local Memory Attention Networks (LMANet), that transforms an existing single-frame semantic segmentation model into a video semantic segmentation pipeline. Our approach, summarized in Fig. 4.4, aggregates an encoded representation from several past frames into a Memory module described in section 4.3.1. We describe how this memory is accessed via an attention-based module in section 4.3.2 and 4.3.3. Finally, in section 4.3.4, we detail how the features map from the Query frame are fused with the aggregated features read from the Memory to provide an input for the Decoder.

4.3.1 *Memory*

Having a representation of the temporal information from previous frames is a way to leverage the additional dimension offered by videos. Previous works trying to model this use a combination of optical flow and GRUs [113, 90]. However, RNNs are often difficult to train, requiring long sequences. More importantly, the hidden state can easily get corrupted over time, severely affecting the performance during inference. In contrast, we propose to use a memory that holds semantic information from past frames as a core component of our framework. Following the success of attention-based models for video semantic segmentation [69, 50, 49], we construct the memory using *Keys* ($\mathcal{K}_{\mathrm{M}}$) and *Values* ($\mathcal{V}_{\mathrm{M}}$) for each cell, such that the dimensionality of the Keys is much lower than the one of the Values. This strategy allows us to efficiently read out semantic information, stored in the values, by matching keys.

We build and update the memory using consecutive inputs. For each input frame I_t, we generate 2 pairs of (Key,Value) features: one for the Memory (K^t_{M}, V^t_{M}) and one for the Query (K^t_{Q}, V^t_{Q}). Those features are all obtained from the same backbone Encoder output, each with its own simple convolution layer that preserves their spatial size, while reducing the dimensionality of the keys. To keep a simple and explicit temporal representation of the Memory, the two sets of feature maps $\mathcal{K}_{\mathrm{M}}$ and $\mathcal{V}_{\mathrm{M}}$ are obtained by concatenating the keys and values from the previous frames, as follows We define $\mathcal{V}_{\mathrm{M}} = [V^1_{\mathrm{M}}, \ldots, V^{L-1}_{\mathrm{M}}] \in \mathbb{R}^{H\times W\times D_{\mathrm{V}}\times(L-1)}$ and $\mathcal{K}_{\mathrm{M}} = [K^1_{\mathrm{M}}, \ldots, K^{L-1}_{\mathrm{M}}] \in \mathbb{R}^{H\times W\times D_{\mathrm{K}}\times(L-1)}$. With $\mathcal{S} = \{1, \ldots, L-1\}$ we denote the list of indices of the frames included in memory.

4.3.2 *Local Key Matching*

In order to segment a new frame, we need to access the semantic information from the Memory. This is done by matching the feature maps from the Memory and the Query, respectively K_{M} and K_{Q}. Those feature maps have a spatial size of $H \times W$ and a dimensionality of D_{K}. We let $K_{\mathrm{M}}(i,j)$ and $K_{\mathrm{Q}}(k,l)$ denote the features vectors of dimension $\mathbb{R}^{D_{\mathrm{K}}}$ at their respective spatial locations $(i,j), (k,l) \in \{1,\ldots,H\} \times \{1,\ldots,W\}$.

Previous approaches using transformers network architectures for NLP [117] or video object segmentation [92] perform an all-vs-all matching of their feature maps. However, this is memory-consuming and computationally expensive. The similarities between all pairs of spatial locations in the Memory and Query feature maps can be written as a 4D tensor $G(K_{\mathrm{M}}, K_{\mathrm{Q}}) \in \mathbb{R}^{H\times W\times H\times W}$ defined as

$$G(K_{\mathrm{M}}, K_{\mathrm{Q}}) = K_{\mathrm{M}}(i,j)^{\mathrm{T}} K_{\mathrm{Q}}(k,l). \tag{4.1}$$

Particularly in our case, global matching can be problematic for three reasons. The spatial size of feature maps and their dimensionality not only makes it harder to fit reasonable batch sizes and consecutive frames in GPU memory during training but also degrades inference time. Moreover, matching features at completely different locations can introduce wrong correlations between similar classes or instances in the scene.

To address these issues, we exploit the temporal prior given by our input video. In general, the content of the current frame at a given position is more likely to be found in a similar position in the previous few frames. For each feature location in the current frame, we thus only read from the memory in its spatial neighborhood. We show that this operation can be efficiently implemented as a correlation layer, commonly used in optical flow networks [33].

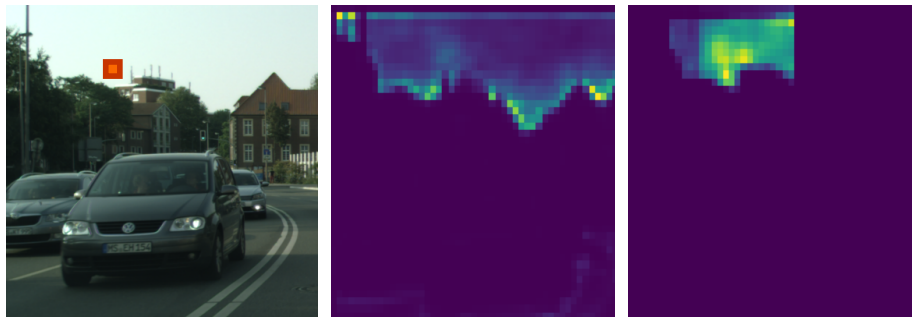


Figure 4.2 – For a given location (orange) in the query image (left), we compare the global correlation (middle) and the local correlation (right).

Formally, the pairwise similarities between all the locations in the Query and their corresponding neighborhood in the Memory can be written as a 4D tensor $C(K_{\mathrm{M}}, K_{\mathrm{Q}}) \in \mathbb{R}^{H\times W\times R\times R}$, where R defines the radius of the local neighborhood from a given location. This can be expressed as

$$C(K_{\mathrm{M}}, K_{\mathrm{Q}}) = K_{\mathrm{M}}(i,j)^{\mathrm{T}} K_{\mathrm{Q}}(i+k, j+l). \tag{4.2}$$

In this case, $(i,j) \in \{1,\dots,H\}\times\{1,\dots,W\}$ and $(k,l) \in \mathcal{R} = \{-R,\dots,R\}^2$ represents the search region for the correlation. (k,l) therefore represents the displacement relative to the reference frame location (i,j), constrained to a radius R.

We choose R such that $R << H, W$. While the local correlation cannot match features that are located outside the limited search region, we can immensely reduce the complexity in memory and computation time for feature maps of a large spatial size, from $\mathcal{O}((HW)^2)$ to $\mathcal{O}((HW)\times R^2)$.

4.3.3 *Local Memory Attention*

The Memory reading operation is the core part of our method. It enables us to access the relevant features contained in the Memory at the right location in an efficient way, thanks to the matching described in the previous section. It is performed in three steps. First, the Query key is matched against all the keys in the Memory. We thus define

$$C^t_{kl}(i,j) = C(K^t_{\mathrm{M}}, K_{\mathrm{Q}}) = K^t_{\mathrm{M}}(i,j)^{\mathrm{T}} K_{\mathrm{Q}}(i+k, j+l). \tag{4.3}$$

Then, we transform feature correlation maps into a set of probability maps using a SoftMax layer, expressed as

$$P_{kl}^{t}(i,j) = \frac{\exp(C_{kl}^{t}(i,j))}{\sum_{klt}\exp(C_{kl}^{t}(i,j))}. \tag{4.4}$$

The resulting probability maps represent the multiplicative attention weights, visualized in Figure 4.2.

These weights are crucial for the last step of the Memory Reading operation as they allow us to weight the features read from the Memory in both spatial and temporal dimensions. The module thus only reads from the parts of the Memory that contain the most relevant information for each location. Formally, the output tensor $\widetilde{\mathcal{V}_{\mathrm{M}}}$ read from the Memory at each location (i,j) can be written as

$$\widetilde{\mathcal{V}_{\mathrm{M}}}(i,j) = \sum_{t\in\mathcal{S}}\sum_{(k,l)\in\mathcal{R}} P_{kl}^{t}(i,j)V_{\mathrm{M}}^{t}(i+k,j+l). \tag{4.5}$$

The read operation is a weighted sum of all values of the memory V_{M}^{t} over the temporal domain $\mathcal{S}$ and the spatial domain $\mathcal{R}$. By construction, its dimensions are identical to the dimensions of the Query value tensor: $V_{\mathrm{Q}}, \widetilde{\mathcal{V}_{\mathrm{M}}} \in \mathbb{R}^{H\times W\times D_{\mathrm{V}}}$.

4.3.4 *Memory Fusion*

As described in the previous section, the semantic features read from the Memory $\widetilde{\mathcal{V}_{\mathrm{M}}}$ contain a rich information from the aggregation of the semantics of previous frames. The features extracted in the Query V_{Q} contain the encoded semantic segmentation of the current frame. The role of the Fusion module is to combine V_{Q} and $\widetilde{\mathcal{V}_{\mathrm{M}}}$ and provide a meaningful input the Decoder that can then make the most accurate prediction for the current frame. The naive approach that would consist in concatenating feature maps would not allow the network to select the best features from both sources of information. Indeed, the model ideally has to learn to trust more the Memory for parts of the scene that have a consistent behaviour over time (static classes, regular or slow motions, etc.) while taking into account the current frame features that might introduce sudden changes (objects entering the scene, occlusions, irregular motions, etc.).

In order to overcome this limitation, we extend the concatenation solution by adding one attention branch per Fusion input, see Fig. 4.3. Each branch uses its corresponding Fusion input and the concatenated inputs. Those attention branches, with the help of sigmoid activated gates inspired by RNNs, are controlling the information flowing through them. This structure therefore allows to select which features from the Query or from the Memory will be sent

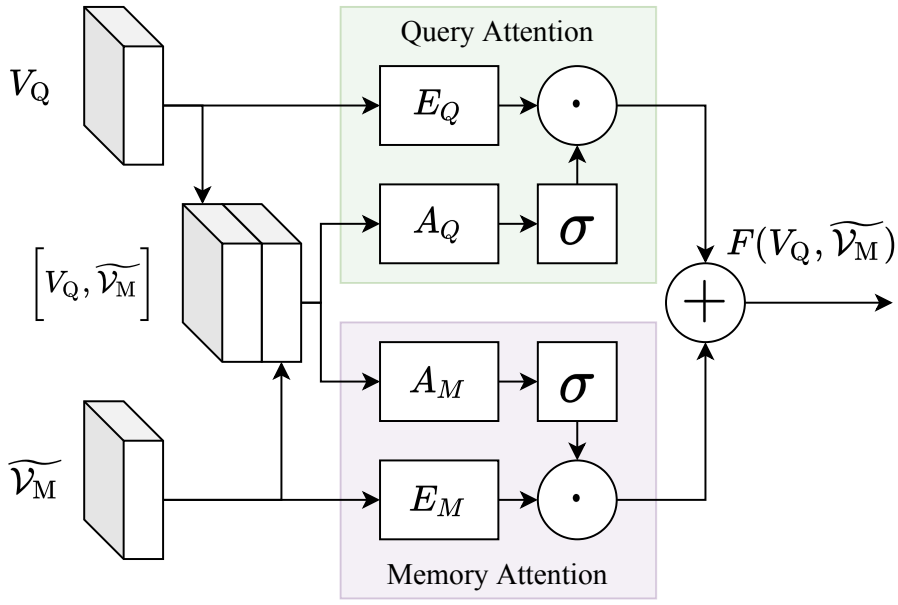


Figure 4.3 – Fusion of the features read from the memory and the current frame.

to the Decoder. Formally, if we let σ be the sigmoid activation function and '$\cdot$' be the element-wise tensor multiplication, the output of the Fusion module can be written as

$$\begin{aligned} F(V_Q, \widetilde{\mathcal{V}_M}) = \sigma\big(A_Q(V_Q, \widetilde{\mathcal{V}_M})\big) \cdot E_Q(V_Q) \\ + \sigma\big(A_M(V_Q, \widetilde{\mathcal{V}_M})\big) \cdot E_M(\widetilde{\mathcal{V}_M}). \end{aligned} \tag{4.6}$$

The Fusion output $F(V_Q, \widetilde{\mathcal{V}_M}) \in \mathbb{R}^{H \times W \times D}$ is a 3D tensor with the same dimensions as the Encoder output. E_Q, A_Q, A_M, E_M represent respectively the learned convolution layers of the Query attention and Memory attention branches, as shown in Figure 4.3.

4.3.5 *Training method*

Our goal is to enable single-frame semantic segmentation models to deal with video inputs, thus we need a simple and efficient training routine. However, two obstacles stand in the way. First, we have a limited amount of annotated data. Second, we need to fit a large enough amount of images on the GPU while keeping reasonable batch sizes.

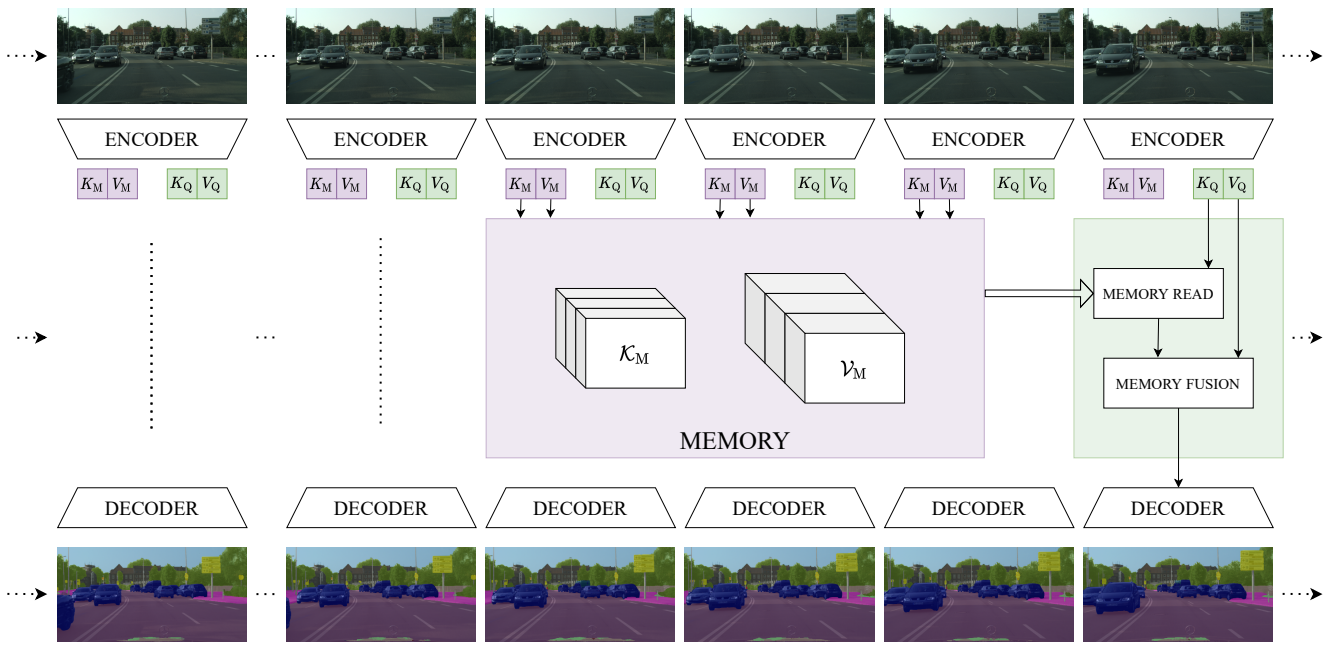


Figure 4.4 – Overview of our framework, built upon an existing Encoder-Decoder backbone. For each input frame, the Encoder output is used to create two *(Key, Value)* pairs. One is stored in the *Memory* module (violet) for future use and the other is used by the Memory Read and Fusion modules (green). The Memory Read module first accesses the relevant semantic information in the Memory. The information read from the memory is then combined with features of the current frame. The segmentation Decoder employs this fused representation to finally predict the segmentation mask.

Assuming that the backbone Encoder and Decoder are trained beforehand on single frames, we propose to train our module in two steps. First, we freeze the parameters of both the Encoder and the Decoder while we train our Memory layers and Fusion module from scratch for a small number of epochs. We choose the number of epochs between 20 and 50 so that the accuracy of our model reaches the range of the baseline accuracy without aiming for convergence. This allows the layers of our modules to get a rough initialization before we move on to the next step, where we jointly train our modules and the Decoder until convergence.

4.4 EVALUATION

4.4.1 *Experimental setup*

We run all our experiments on the Cityscapes dataset [24], as it is commonly used to benchmark both single frame and video semantic segmentation methods. It contains diverse urban street scenes across 50 different cities which makes it relevant for autonomous driving applications. Its training, validation, and testing sets contain respectively 2975, 500, and 1525 images with a resolution of 1024×2048 pixels. Each of those images represents the 20^{th} frame of a corresponding 30 frames video sequence. The training and validation sets provide dense ground truth annotations, but only for the 20^{th} frame of each sequence. We, therefore, train and evaluate our approach on video snippets but report mean Intersection over Union (mIoU) results on the provided 20^{th} frame.

Our experimental setup consists of several Intel Xeon Gold 6242 CPUs running at 3.5GHz and several NVIDIA GeForce RTX 2080Ti GPUs. We use Python 3.7, PyTorch 1.2, CUDA 10.2 and cuDNN 7.6. While simple inference can be run on a single GPU, training is usually performed on several GPUs to either speed up training or overcome GPU memory limitations. For training, we use the Adam optimizer with an initial learning rate of 2e-4. We use adaptive scheduling based on the training loss: the learning rate is kept constant until the loss reaches a minimum plateau, after which we decrease it by 20%. We keep this strategy until the learning rate reaches a minimum value of 1e-8.

4.4.2 *Ablation study*

We perform an ablation study to highlight the role of each component of our method and its impact on the performance of the pipeline. We thus focus on the critical parameters of our modules: the search region for the correlation layer,

the size of the Memory used during inference, and the fusion strategy. We run this ablation study using ERFNet [103] as the backbone for efficiency reasons.

Memory size $\mathcal{S}$: The first and foremost important aspect of our method is the size of its Memory. Especially in the context of autonomous driving, it is important to make sure to look at the past frames without holding on to too old information. At the same time, we have to make sure that the memory is long enough to have meaningful temporal information to use. This part of the study aims at determining the impact of the memory size S on the final prediction accuracy. The case $S = 0$ represents our single-frame baseline ERFNet [103]. Table 4.1 summarizes the results and highlights two aspects. First, all memory sizes improve the baseline from $+1.29\%$ to $+1.67\%$, which means that there is indeed meaningful information to retain from past frames and that our method uses it to improve the final prediction. Second, even though a single frame in the memory already improves the baseline, the best results are achieved with a bigger memory size, with a peak for $S = 4$.

mIoU(%)	$S = 0$	$S = 1$	$S = 2$	$S = 3$	$S = 4$	$S = 5$
LMA, $R = 21$	72.05	73.34	73.41	73.37	73.72	73.46

Table 4.1 – Impact of the size of the Memory on the results.

Search Region $\mathcal{R}$: This part of the study aims at validating the design choice of the keys matching step in our pipeline. As it is the pillar supporting the Memory reading operation, we want to compare an all-vs-all matching to a local one and measure the impact of the search region size on the resulting mIoU. Table 4.2 summarizes the results. These numbers should be compared to the single frame baseline ($S = 0$) which has a mIoU of 72.05%.

Here we investigate and validate three aspects of our approach. First, all settings combinations yield final mIoU improvements compared to the single frame baseline, ranging from $+0.85\%$ to $+1.67\%$. Second, we observe the importance of actually considering a local neighborhood for the features matching when considering temporal information: matching at the same location ($R = 1$) improves results from the baseline, but is far from the improvements observed with bigger search regions. Third, we confirm that using global matching is counterproductive, as local matching with a search radius between $R = 21$ and $R = 41$ seems to yield the best results. This suggests that local matching with a reasonable search region $\mathcal{R}$ is the best option to avoid mismatches that could happen across the whole image while accounting for the temporal information accumulated in the Memory.

mIoU(%)	G	$R = 41$	$R = 21$	$R = 11$	$R = 1$
$S = 3$	73.38	73.50	73.37	73.21	72.90
$S = 4$	73.49	73.64	73.72	73.21	72.94

Table 4.2 – Accuracy comparison between global (G) and local correlations with different search region sizes ($R << G$).

Fusion Strategy: The last part of our method combines features from the Query and the Memory, which hold different kinds of information. The former contains the current frame features while the latter is an aggregation of the information from past frames. Table 4.3 compares our results to a simple concatenation of the semantic maps and confirms that a more complex fusion with attention branches generates more accurate results ($+0.33\%$). This seems to indicate that the probabilistic weighting done by the Fusion is helpful to guide the Decoder.

Baseline	$\text{LMA}_{S=3,R=21}$ (Concat.)	$\text{LMA}_{S=3,R=21}$ (Fusion)
72.05%	73.14%	73.37%

Table 4.3 – mIoU comparison between concatenation and our Fusion.

4.4.3 *Memory and computation time*

As we strive towards a simple and general method for fast video semantic segmentation applications, an important aspect to tackle is its efficiency, both in terms of GPU memory consumption and inference time. We therefore study the impact of our method on those variables on the existing backbone ERFNet [103]. As reported previously, a relatively small memory size of 3 to 5 frames is reasonable for our benchmarks, so we focus on the impact of the search radius R, with a fixed memory size of $S = 3$ and report the results in Table 4.4.

These measurements confirm that an all-vs-all matching of the feature maps is not a viable option for our problem. Indeed, a 166% increase in GPU memory is not desirable during inference in a real-life scenario. It also would complicate the learning phase, where bigger batch sizes are usually beneficial. At the same time, we see that our pipeline adds a very small memory footprint onto the existing backbone when using a local correlation approach, only 13% for $R = 21$, which yields the highest mIoU.

	Baseline	G	$R = 41$	$R = 21$	$R = 11$
Total time T(s)	62.4	96.8	114	71.4	67.5
ΔT	-	+ 55%	+ 83%	+ 14%	+ 8%
GPU Mem. M(MiB)	1553	4125	2116	1749	1606
ΔM	-	+ 166%	+ 36%	+ 13%	+ 3%

Table 4.4 – Inference time and GPU memory consumption on the 500 validation sequences of Cityscapes [24]. Each sequence has a length of $L = 4$ frames and the Memory size $S = 3$. Results are displayed as an average of 10 runs under the same conditions.

	Single-frame baseline			Video approach				
Method	Backbone	mIoU (%)	Time (ms)	mIoU (%)	Time (ms)	ΔmIoU(%)	ΔT(ms)	Resolution
LMA$_{S=4,R=21}$	ERFNet [103]	72.05	10.1	73.72	11.6	+1.67	+1.5	1024×512
LMA$_{S=3,R=21}$	PSP-SS-SC [143]	76.34	406	78.48	758	+2.14	+352	2048×1024
TDNet [50]	BiseNet*18 [50]	73.8	20	75.0	21	+1.2	+1	n.a.
TDNet [50]	BiseNet*34 [50]	76.0	27	76.4	26	+0.4	-1	n.a.
FANet [49]	FANet-18 [49]	75.0	14	75.5	14	+0.5	+0	2048×1024
FANet [49]	FANet-34 [49]	76.3	17	76.7	17	+0.4	+0	2048×1024
GRFP [90]	PSP-SS-MC [143]	79.7	n.a	80.2	n.a	+0.5	+335	512×512
GRFP [90]	PSP-MS-MC [143]	80.9	n.a	81.3	n.a	+0.4	+335	512×512
NetWarp [41]	PSP-SS-MC [143]	79.4	3000	80.6	3040	+1.2	+40	713×713
NetWarp [41]	PSP-MS-MC [143]	80.8	30300	81.5	30500	+0.7	+200	713×713
LLVS [72]	ResNet-101 [46]	80.2	360	76.84	171	-3.36	-189	n.a
DVS [131]	PSP-SS-SC [143]	77.0	588	70.2	87	-6.8	-501	n.a
DVS [131]	PSP-SS-SC [143]	77.0	588	62.6	33	-14.4	-555	n.a
DFF [145]	ResNet-101 [46]	71.1	658	69.2	179	-1.9	-479	1024×512
EVS [96]	ICNet [142]	67.3	26	66.2	13	-1.1	-13	2048×1024
EVS [96]	ICNet [142]	67.3	26	67.6	27	+0.3	+1	2048×1024

Table 4.5 – Comparison between our methods and other existing methods. As we focus on the temporal aspect, we report the single-frame baseline for each method and the relative changes it brings in terms of mean IoU and inference time. The fields marked with "n.a" denote data that was not available in the corresponding publications.

Furthermore, we show that the search radius R is crucial when it comes to inference efficiency, and that a bigger correlation region directly impacts the inference time, while not necessarily yielding the best accuracy results. For a search radius of 21 pixels, we get a limited impact both on the GPU memory (+13%) and on the inference time (+14%).

4.4.4 *Comparison to state-of-the-art*

Since current state-of-the-art approaches conduct experiments with different baselines, backbones, input resolutions and hardware, providing a completely fair comparison is virtually impossible. In order to provide the fairest compari-

Method	Total	road	s.walk	build.	wall	fence	pole	tlight	sign	veg.	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle
ERFNet	72.05	97.65	81.28	90.68	49.31	54.89	59.93	62.31	72.07	91.25	60.83	93.36	75.91	53.28	92.84	72.77	78.77	63.78	46.35	71.74
+ LMA, $R = 21, S = 4$	73.72	97.73	81.81	90.83	51.65	55.32	60.87	63.48	74.21	91.50	61.17	93.43	77.08	57.97	93.21	75.63	81.66	69.51	51.18	72.48
PSPNet	76.34	98.05	84.71	92.47	54.66	60.71	64.05	71.22	79.09	92.64	63.49	94.71	82.46	62.43	94.96	73.87	82.51	58.70	61.70	78.04
+ LMA, $R = 21, S = 3$	78.48	98.06	85.06	92.73	60.13	62.57	64.20	70.22	78.20	92.40	64.17	94.35	81.68	63.16	95.14	80.77	87.93	76.86	67.24	76.21

Table 4.6 – Comparison with two backbones (ERFNet and PSPNet) on class-wise improvements brought with our method LMA (Ours).

ERFNet (Single frame baseline) LMANet (Ours) Ground truth

Figure 4.5 – Qualitative results comparison between our method $\text{LMA}_{4,21}$ and the single-frame ERFNet [103]. Different scenarios are highlighted in the orange boxes and show improvements over the baseline. The top example shows cleaner segmentation masks around a mixture of *barrier*, *motorbike*, and *car*. The bottom example shows a better classification of a *road sign* at the edge of the scene.

son, we report results from the original paper of the single-frame baseline along with the corresponding video-based improvements.

Based on the ablation study, we choose the settings that yield the best mIoU, which is with a search radius $R = 21$, and we compare our framework with other video semantic segmentation methods. To provide a better comparison and validate our approach on a different backbone, we apply our method to two of the most popular backbones, ERFNet [103] and PSPNet [143] and report results in Table 4.5.

ERFNet: While it does not reach the current state-of-the-art semantic segmentation accuracy, ERFNet [103] remains a good compromise between frame rate and accuracy compared to heavier and slower methods such as PSPNet [143] and DeepLab [18]. It is therefore important to see whether our approach can help increase the accuracy of its predictions while keeping a low inference time. Table 4.5 shows that our approach can improve the baseline mIoU by 1.67% while increasing the runtime by only 1.5 ms. This gain in accuracy is bigger than any other existing methods and the very good inference time inherent to ERFNet [103] is preserved. In comparison, EVS [96] improves its baseline by only 0.3% with a similar additional inference time. Attention-based approaches such as TDNet [50] or FANet [49] bring relatively smaller mean IoU improvements to their backbones, at the price of more complex architectures. Other methods starting from baselines below 80% [131, 145] still yield a mIoU vs. inference time ratio that is much worse than our approach.

Figure 4.5 shows the qualitative improvements that our approach can bring through the usage of the Memory with attention mechanisms. In some situations, it increases the temporal consistency of the predictions for fixed objects, for instance *barriers* and *traffic lights*.

PSPNet: The state-of-the-art and publicly available results for PSPNet [143] are achieved by running multiple inferences on image crops of resolution 713×713. In order to get the final prediction on the 1024×2048 resolution, the authors define a grid of 8 overlapping crops and combine the results of independent predictions. They also present results in two flavors, a single-scale version that simply runs at the original scale and another that combines 6 different scales together. We refer in the table to the single-scale and multi-scale versions of the multi-crop inference as *PSP-SS-MC* and *PSP-MS-MC*, respectively.

As we propose a general framework to work with video inputs, such a multi-scale and multi-crop inference on each frame is not practical. First, from an architecture complexity point of view, the implications it would have on our query/memory structure would have to be studied further. Second, it would multiply the average inference time per frame. Therefore, we focus our study

on a single-scale and single-crop version of PSPNet, as in DVSNet [131]. We refer to it as *PSP-SS-SC*.

In this setup, our approach not only improves the accuracy of a complex backbone but also generalizes to other single-frame models. Moreover, while our approach has a bigger impact on the inference time when using the PSPNet backbone, compared to the ERFNet case, the relative improvement we gain from our simple structure outperforms other more complex approaches.

4.4.5 *Class-wise analysis*

The mean intersection over union only offers a partial view of the improvements achieved by our approach. In this section, we propose an overview of the results we get with our LMANet for each class, on both backbones ERFNet [103] and PSPNet [143], see Table 4.6. Two aspects stand out with ERFNet [103]. First, the set of classes that were already well segmented (road, building, vegetation, terrain, sky, car) does not change too much from the baseline. They mostly get improved by a small margin or slightly degrade. Second, smaller and dynamic classes that are the most critical to understand the dynamics of the scene get improvements ranging from 0.5% to almost 6%. This is especially relevant in the context of autonomous driving and flying.

With PSPNet [143], the trend is similar but with some noticeable differences. While bigger classes, including moving dynamic ones (train, truck, bus, motorcycle, wall) are improving substantially, smaller objects do not benefit in the same manner as before. This might be due to the stronger feature representation provided by the ResNet-101 backbone along with the higher resolution of the feature map: $256 \times 128 \times 2048$ compared to $128 \times 256 \times 128$ for ERFNet.

4.5 CONCLUSION

We present a novel neural network architecture that transforms an existing single frame semantic segmentation model into a video semantic segmentation pipeline. In contrast to prior methods, we strive towards a simple structure that can easily be integrated and trained together with any single-frame semantic segmentation model. Our approach aggregates the semantic information from past frames into a memory module and uses local attention mechanisms both to access the features stored in the Memory and to fuse them with the input Query frame. We validate our approach on two popular semantic segmentation networks and show that we increase the segmentation performance of ERFNet on Cityscapes by almost 2% while preserving its real-time performance.

APPENDICES

In the following appendices, we provide detailed results and a deeper analysis of the experiments presented in the chapter. We provide the results obtained for each class and compare them for different operating points and backbones in section 4.A. Finally, we further investigate the memory size in section 4.B.

4.A CLASS-WISE ANALYSIS

The mean intersection over union only offers a partial view of the improvements achieved by our approach. In this section, we propose an overview of the results and improvements we get with our LMANet for each class, on both backbones ERFNet [103] and PSPNet [143] (Based on ResNet-101 [46]). We also show the impact of the Memory size and search radius on those classes.

4.A.1 *ERFNet backbone*

Influence of the Memory size S: For a fixed search radius $R = 21$, we analyze the impact of the Memory size on each individual class of the Cityscapes [24] validation set for ERFNet [103]. Table 4.7 summarizes the absolute IoU for each class while Figure 4.6 shows visually the relative improvements we obtain.

What stands out at first from the results is the fact that a set of classes that were already well segmented (road, building, vegetation, terrain, sky, car) do not change too much from the baseline. They mostly get improved by a small margin or slightly degrade. Other classes, on the other hand, get improvements ranging from 0.5% to almost 6%, which is very interesting, as many of those classes are either small and/or dynamic classes that are critical to understanding the dynamics of the scene in the case of autonomous driving. Besides, it seems that the best improvements are in general yielded for bigger Memory sizes, while on average improvements are less striking.

In that category, the following classes are worth mentioning for our best operating point ($S = 4, R = 21$): wall (+2.34%), traffic light/sign (+ 1.17% / + 2.14%), person/rider (+ 1.17% / + 4.69%), truck (+ 2.86%), bus (+ 2.89%), train (+ 5.73%), motorcycle (+ 4.83%). We can also note that, for that operating point, all the classes are improving compared to the single-frame baseline.

Influence of the Search Radius R: Similarly, for a fixed Memory size $S = 3$, we analyze the impact of the search radius on each individual class of the Cityscapes [24] validation set for ERFNet [103]. Table 4.8 summarizes

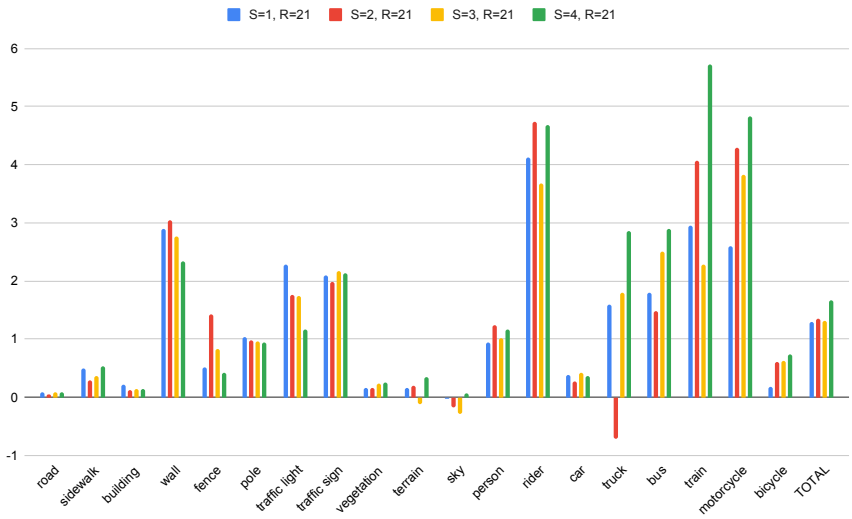


Figure 4.6 – Per-class IoU difference between ERFNet and LMA (Ours) for different Memory sizes S with a search radius $R = 21$.

Method	Total	road	s.walk	build.	wall	fence	pole	tlight	sign	veg.	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle
ERFNet	72.05	97.65	81.28	90.68	49.31	54.89	59.93	62.31	72.07	91.25	60.83	93.36	75.91	53.28	92.84	72.77	78.77	63.78	46.35	71.74
LMA, $S = 1$	73.34	97.73	81.77	90.90	52.2	55.41	60.96	64.60	74.17	91.42	61.00	93.34	76.85	57.40	93.23	74.36	80.56	66.74	48.95	71.92
LMA, $S = 2$	73.41	97.69	81.57	90.81	52.36	56.31	60.91	64.08	74.05	91.42	61.03	93.19	77.15	58.01	93.12	72.06	80.25	67.84	50.65	72.35
LMA, $S = 3$	73.37	97.73	81.64	90.82	52.08	55.72	60.90	64.06	74.24	91.49	60.72	93.07	76.93	56.96	93.27	74.57	81.27	66.06	50.17	72.36
LMA, $S = 4$	73.72	97.73	81.81	90.83	51.65	55.32	60.87	63.48	74.21	91.50	61.17	93.43	77.08	57.97	93.21	75.63	81.66	69.51	51.18	72.48

Table 4.7 – Comparison between ERFNet and LMA (Ours) for different Memory sizes S, with $R = 21$.

the absolute IoU for each class while Figure 4.7 shows visually the relative improvements we obtain.

The results are consistent with the previous section, the same classes follow the same trend, for all search radii. What is most interesting to see here, is that except for the two classes *train* and *bus*, the all-vs-all matching yield worse results compared to a local matching, which is the behavior we observed in our ablation study on average.

4.A.2 *PSPNet backbone*

Finally, we analyze the improvements obtained on the Cityscapes [24] validation set for PSPNet [143]. Table 4.9 summarizes the absolute IoU for each class while Figure 4.8 shows visually the relative improvements we have.

In this case, the trend is similar but with some noticeable differences. While bigger classes, including moving dynamic ones (train, truck, bus, motorcycle,

wall) are improving substantially, smaller objects do not benefit in the same manner as before. This might be due to the stronger feature representation provided by the ResNet-101 backbone along with the higher resolution of the feature map: $256 \times 128 \times 2048$ compared to $128 \times 256 \times 128$ for ERFNet.

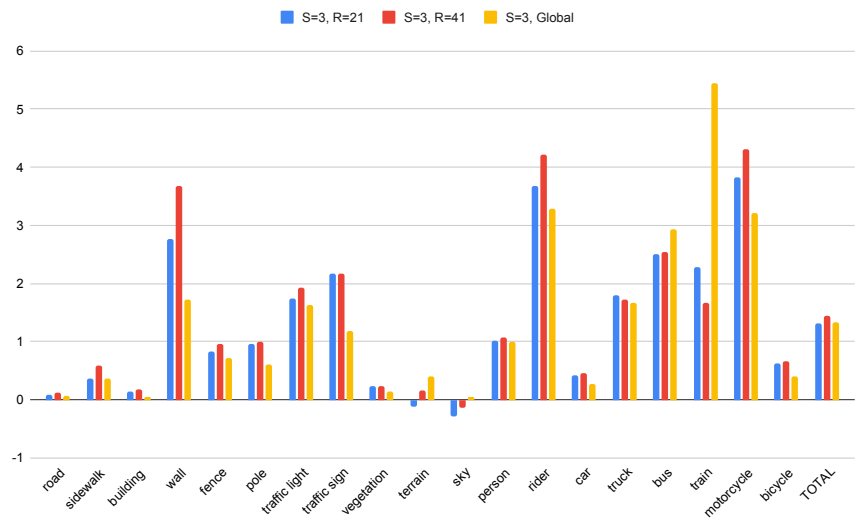


Figure 4.7 – Per-class IoU difference between our ERFNet baseline and LMA (Ours) for different search radius R with a Memory size $S = 3$.

Method	Total	road	s.walk	build.	wall	fence	pole	tlight	sign	veg.	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle
ERFNet	72.05	97.65	81.28	90.68	49.31	54.89	59.93	62.31	72.07	91.25	60.83	93.36	75.91	53.28	92.84	72.77	78.77	63.78	46.35	71.74
LMA, Global	73.38	97.71	81.65	90.72	51.04	55.61	60.54	63.95	73.25	91.39	61.23	93.41	76.91	56.56	93.11	74.43	81.71	69.22	49.57	72.15
LMA, $R = 41$	73.50	97.77	81.86	90.86	52.99	55.85	60.92	64.23	74.24	91.49	61.00	93.22	76.98	57.50	93.30	74.49	81.32	65.44	50.66	72.41
LMA, $R = 21$	73.37	97.73	81.64	90.82	52.08	55.72	60.90	64.06	74.24	91.49	60.72	93.07	76.93	56.96	93.27	74.57	81.27	66.06	50.17	72.36

Table 4.8 – Comparison between ERFNet and LMA (Ours) for different search radius R, with $S = 3$.

4.B INFLUENCE OF THE MEMORY SIZE DURING INFERENCE

This last experiment consists in evaluating the impact of performing inference with a different Memory size than the model was trained for. In this example, we trained our model with $S = 4$, and try the inference with bigger Memory sizes, as shown in Table 4.10.

This shows the robustness of our approach since the accuracy does not degrade with an increased Memory size. In fact, we even see a slight improvement

Method	Total	road	s.walk	build.	wall	fence	pole	tlight	sign	veg.	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle
PSPNet	76.34	98.05	84.71	92.47	54.66	60.71	64.05	71.22	79.09	92.64	63.49	94.71	82.46	62.43	94.96	73.87	82.51	58.70	61.70	78.04
LMA, $S = 3$	78.48	98.06	85.06	92.73	60.13	62.57	64.20	70.22	78.20	92.40	64.17	94.35	81.68	63.16	95.14	80.77	87.93	76.86	67.24	76.21

Table 4.9 – Comparison between PSPNet and LMA (Ours) with $S = 3$ and $R = 21$.

when using a much larger Memory, meaning that our method can still make use of the additional information in a meaningful way.

Method	$S = 4$	$S = 5$	$S = 6$	$S = 7$	$S = 8$	$S = 9$	$S = 10$	$S = 11$	$S = 12$	$S = 13$	$S = 14$	$S = 15$	$S = 16$	$S = 17$	$S = 18$	$S = 19$
LMA, $R = 21$	73.72	73.69	73.73	73.72	73.70	73.72	73.71	73.72	73.72	73.73	73.75	73.77	73.78	73.70	73.78	73.77

Table 4.10 – Impact of the size of the Memory during inference on the final mIoU(%), based on a model trained with an ERFNet backbone and a Memory size of $S = 4$.

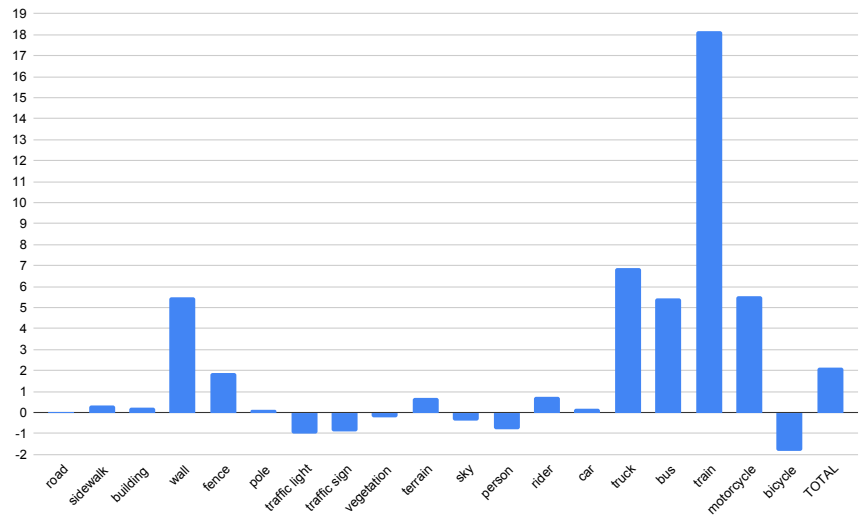


Figure 4.8 – Per-class IoU difference between PSPNet and LMA (Ours) with $S = 3$ and $R = 21$.

PART II

VIDEO OBJECT SEGMENTATION AND TRACKING

5 ROBUST TRACKING BY SEGMENTATION

Estimating the target extent poses a fundamental challenge in visual object tracking. Typically, trackers are *box-centric* and fully rely on a bounding box to define the target in the scene. In practice, objects often have complex shapes and are not aligned with the image axis. In these cases, bounding boxes do not provide an accurate description of the target and often contain a majority of background pixels.

In this chapter, we propose a *segmentation-centric* tracking pipeline that not only produces a highly accurate segmentation mask, but also internally works with segmentation masks instead of bounding boxes. Thus, our tracker is able to better learn a target representation that clearly differentiates the target in the scene from background content. In order to achieve the necessary robustness for the challenging tracking scenario, we propose a separate instance localization component that is used to condition the segmentation decoder when producing the output mask. We infer a bounding box from the segmentation mask, validate our tracker on challenging tracking datasets and achieve the new state of the art on LaSOT with a success AUC score of 69.7%. Since most tracking datasets do not contain mask annotations, we cannot use them to evaluate predicted segmentation masks. Instead, we validate our segmentation quality on two popular video object segmentation datasets.

5.1 INTRODUCTION

Visual object tracking is the task of estimating the state of a target object for each frame in a video sequence. The target is solely characterized by its initial state in the video. Current approaches predominately characterize the state itself with a bounding box. However, this only gives a very coarse representation of the target in the image. In practice, objects often have complex shapes, and undergo substantial deformations. Often, targets do not align well with the image axes, while most benchmarks use axis-aligned bounding boxes. In such cases, the majority of the image content inside the target's bounding box often consists of background regions that provide limited information about the object itself. In contrast, a segmentation mask precisely indicates the object's extent in the image (see Fig. 5.1 frames #1600 and #3200). Such information is vital in a variety of applications, including video analysis, video editing, and robotics.

In this work, we therefore develop an approach for accurate and robust target object segmentation, even in the highly challenging tracking datasets [35, 87].

While severely limiting the information about the target's state in the video, the aforementioned issues with the bounding box representation can itself lead to inaccurate bounding box predictions, or even tracking failure. To illustrate this, Fig. 5.1 shows two typical tracking sequences. The tracking method STARK [133] (first row) fails to regress bounding boxes that contain the entire object (#1600, #1400) or even starts tracking the wrong object (#0700). Conversely, segmentation masks are a better fit to differentiate pixels in the scene that belong to the background and the target. Therefore, a *segmentation-centric* tracking architecture designed to work internally with a segmentation mask of the target instead of a bounding box has the potential to learn better target representations, because it can clearly differentiate background from foreground regions in the scene.

A few recent tracking methods [121, 132] have recognized the advantage of producing segmentation masks instead of bounding boxes as final output. However, these trackers are typically *bounding-box-centric* and the final segmentation mask is obtained by a separate *box-to-mask* post-processing network. These methods do not leverage the accurate target definition of segmentation masks to learn a more accurate and robust internal representation of the target.

In contrast, most Video Object Segmentation (VOS) methods [92, 6] follow a *segmentation-centric* paradigm. However, these methods are not designed for challenging tracking scenarios. Typical VOS sequences consist only of a few hundred frames [101] whereas multiple sequences of more than ten thousand frames exist in tracking datasets [35]. Due to this setup, VOS methods focus on producing highly accurate segmentation masks but are sensitive to distractors, substantial deformations, and occlusions of the target object. Fig. 5.1 shows two typical tracking sequences where the VOS method LWL [6] (second row) produces a fine-grained segmentation mask of the wrong object (#3200) or is unable to detect only the target within a crowd (#0700, #1400).

We propose *Robust Visual Tracking by Segmentation* (RTS), a unified tracking architecture capable of predicting accurate segmentation masks. To design a *segmentation-centric* approach, we take inspiration from the aforementioned LWL [6] method. However, to achieve robust and accurate segmentation on Visual Object Tracking (VOT) datasets, we introduce several new components. In particular, we propose an instance localization branch trained to predict a target appearance model, which allows occlusion detection and target identification even in cluttered scenes. The output of the instance localization branch is further used to condition the high-dimensional mask encoding. This allows the segmentation decoder to focus on the localized target, leading to a more robust mask prediction. Since our proposed method contains a segmentation and an

instance memory that need to be updated with previous tracking results, we design a memory management module. This module first assesses the prediction quality, decides whether the sample should enter the memory, and triggers the model update when necessary.

Contributions: Our contributions are the following: **(i)** We propose a unified tracking architecture capable of predicting robust classification scores and accurate segmentation masks. We design separate feature spaces and memories to ensure optimal receptive fields and update rates for segmentation and instance localization. **(ii)** To produce a segmentation mask that agrees with the instance prediction, we design a fusion mechanism that conditions the segmentation decoder on the instance localization output and leads to more robust tracking performance. **(iii)** We introduce an effective inference procedure capable of fusing the instance localization output and mask encoding to ensure both robust and accurate tracking. **(iv)** We perform comprehensive evaluation and ablation studies of the proposed tracking pipeline on multiple popular tracking benchmarks. Our approach achieves the new state of the art on LaSOT with an *area-under-the-curve* (AUC) score of 69.7%.

5.2 RELATED WORK

Visual Object Tracking: Over the years, research in the field of visual tracking has been accelerated by the introduction of new and challenging benchmarks, such as LaSOT [35], GOT-10k [52], and TrackingNet [88]. This led to the introduction of new paradigms in visual object tracking, based on Discriminative Correlation Filters (DCFs), Siamese networks and Transformers.

One of the most popular type of approaches, DCF-based visual trackers [7, 47, 30, 78, 26, 122, 144, 4, 29] essentially solve an optimization problem to estimate the weights of the DCF that allow to distinguish foreground from background regions. The DCF is often referred to as the target appearance model and allows to localize the target in the video frame. More recent DCF approaches [4, 29] enable end-to-end training by unrolling a fixed number of the optimization iterations during *offline* training.

Siamese tracking methods have gained in popularity due to their simplicity, speed and end-to-end trainability [116, 3, 111, 146, 44, 124, 45, 68, 67]. These trackers learn a similarity metric using only the initial video frame and its annotation that allows to clearly identify the target *offline*. Since no *online* learning component is involved, these trackers achieve high frame rates at the cost of limited *online* adaptability to changes of the target's appearance. Nonetheless, several methods have been proposed to overcome these issues [116, 3, 68, 67].

Figure 5.1 – Comparison between the Visual Object Tracking (VOT) method Stark [133], the Video Object Segmentation (VOS) method LWL [6] and our proposed method on two tracking sequences from the LaSOT [35] dataset. The ground-truth annotation (□) is shown in each frame for reference. Our approach is more robust and predicts a more accurate target representation.

Very recently, Transformer-based trackers have achieved state-of-the-art performance on many datasets, often outperforming their rivals. This group of trackers typically uses a Transformer component in order to fuse information extracted from training and test frames. This produces discriminative features that allow to accurately localize and estimate the target in the scene [19, 135, 133, 123, 82].

Video Object Segmentation: Semi-supervised VOS is the task of classifying all pixels belonging to the target in each video frame, given only the segmentation mask of the target in the initial frame. The cost of annotating accurate segmentation masks is limiting the sequence length and number of videos contained in available VOS datasets. Despite the relatively small size of VOS datasets compared to other computer vision problems, new benchmarks such as Youtube-VOS [130] and DAVIS [101] accelerated the research progress in the last years.

Some methods rely on a learnt target detector [12, 120, 81], others learn how to propagate the segmentation mask across frames [129, 99, 70, 55]. Another group of methods uses feature matching techniques across one or multiple frames with or without using an explicit spatio-temporal memory [21, 51, 118, 92]. Recently, Bhat *et al.* [6] employed meta-learning approach, introducing an end-to-end trainable VOS architecture. In this approach, a few-shot learner predicts a learnable labels encoding. It generates and updates *online* the parameters of a segmentation target model that produces the mask encoding used to generate the final segmentation mask.

Joint Visual Tracking and Segmentation: A group of tracking methods have already identified the advantages of predicting a segmentation mask instead of a bounding box [132, 141, 121, 77, 125, 109]. Siam-RCNN is a box-centric tracker that uses a pretrained *box2seg* network to predict the segmentation mask given a bounding box prediction. In contrast, AlphaRefine represents a novel *box2seg* method that has been evaluated with many recent trackers such as SuperDiMP [29] and SiamRPN++ [67]. Further, Zhao *et al.* [141] focus on generating segmentation masks from bounding box annotations in videos using a spatio-temporal aggregation module to mine consistencies of the scene across multiple frames. Conversely, SiamMask [125] and D3S [77] are segmentation-centric trackers that produce a segmentation mask directly, without employing a *box2seg* module. In particular, SiamMask [125] is a fully-convolutional Siamese network with a separate branch which predicts binary segmentation masks supervised by a segmentation loss.

From a high-level view, the single-shot segmentation tracker D3S [77] is most related to our proposed method. Both methods employ two dedicated modules or branches; one for localization and one for segmentation. D3S adopts the

target classification component of ATOM [26], requiring online optimization of weights in a two-layer CNN. In contrast, we learn online the weights of a DCF similar to DiMP [4]. For segmentation, D3S [77] propose a feature matching technique that matches test frame features with background and foreground features corresponding to the initial frame. In contrast, we adopt the few-shot learning based model prediction proposed in LWL [6] to produce accurate segmentation masks. Furthermore, D3S proposes to simply concatenate the outputs of both modules whereas we learn a localization encoding to condition the segmentation mask decoding based on the localization information. Compared to D3S, we update not only the instance localization but also the segmentation models and memories. Hence, our method integrates specific memory management components.

5.3 METHOD

5.3.1 *Overview*

Video object segmentation methods can produce high quality segmentation masks but are typically not robust enough for video object tracking. Robustness becomes vital for medium and long sequences, which are most prevalent in tracking datasets [35, 87]. In such scenarios, the target object frequently undergoes substantial appearance changes. Occlusions and similarly looking objects are common. Hence, we propose to adapt a typical VOS approach with tracking components to increase its robustness. In particular, we base our approach on the Learning What to Learn (LWL) [6] method and design a novel and segmentation-centric tracking pipeline that estimates accurate object masks instead of bounding boxes. During inference, a segmentation mask is typically not provided in visual object tracking. Hence, we use STA [141] to generate a segmentation mask from the provided initial bounding box. An overview of our RTS method is shown in Fig. 5.2. Our pipeline consists of a backbone network, a segmentation branch, an instance localization branch and a segmentation decoder. For each video frame, the backbone first extracts a feature map x_{b}. These features are further processed into segmentation features x_{s} and classification features x_{c} to serve as input for their respective branch. The segmentation branch is designed to capture the details of the object with a high dimensional mask encoding, whereas the instance localization branch aims at providing a coarser but robust score map representing the target location. Both branches contain components learned online, trained on memories ($\mathcal{D}_{\mathrm{s}}$ and $\mathcal{D}_{\mathrm{c}}$) that store features and predictions of past frames. The instance localization branch has two purposes. The first is to control models and memories updating.

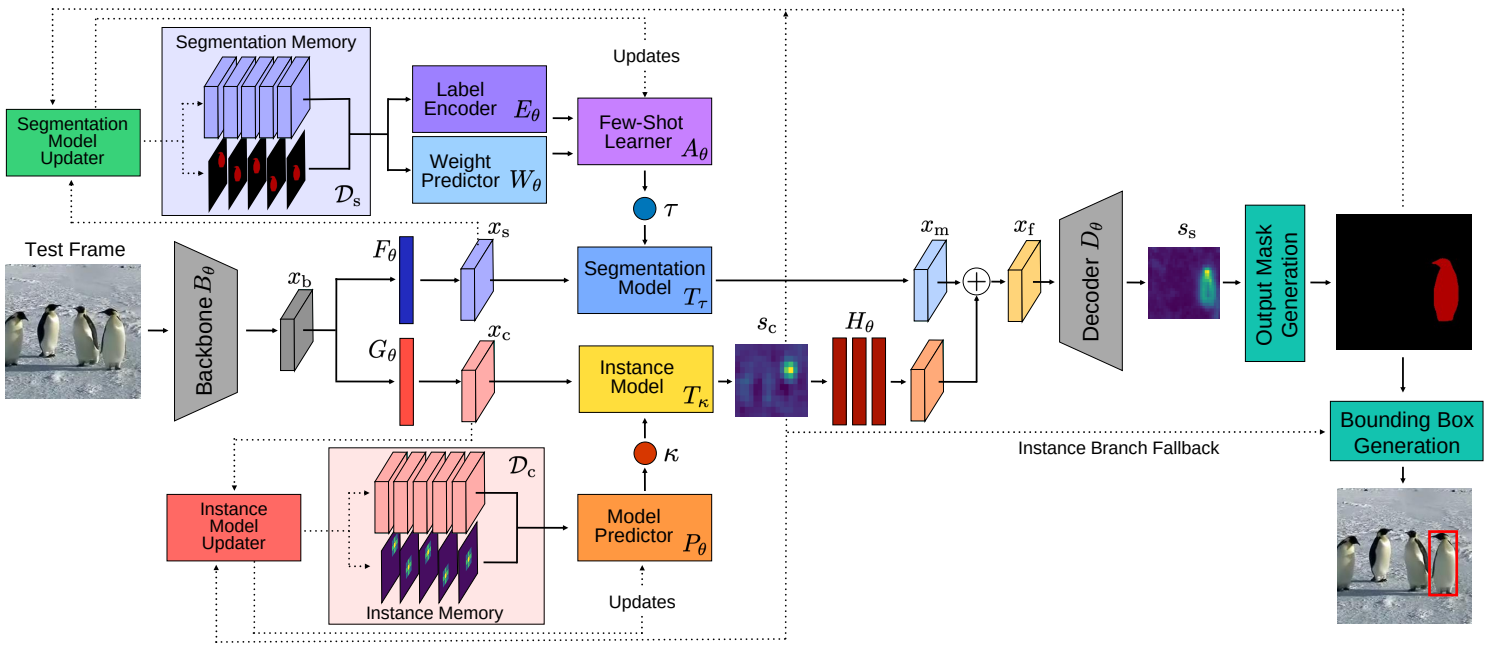


Figure 5.2 – Overview of our entire online tracking pipeline used for inference, see Sec 5.3.1.

The second is used to condition the segmentation mask decoder. To do so, we add instance localization information with a learnt score encoding produced by H_θ. The obtained segmentation scores and the raw instance model score map are then used to generate the final segmentation mask output.

5.3.2 *Segmentation Branch*

The architecture of the segmentation branch is adopted from LWL [6], and we briefly review it here. It consists of a segmentation sample memory $\mathcal{D}_\mathrm{s}$, a label generator E_θ, a weight predictor W_θ, a few-shot learner A_θ and a segmentation model T_τ. The goal of the few-shot learner A_θ is producing the parameters τ of the segmentation model T_τ such that the obtained mask encoding x_m contains the information needed to compute the final segmentation mask of the target object. The label mask encodings used by the few-shot learner are predicted by the label generator E_θ.

The few-shot learner is formulated through the following optimization problem, which is unrolled through steepest descent iterations in the network

$$L_\mathrm{s}(\tau) = \frac{1}{2} \sum_{(x_\mathrm{s}, y_\mathrm{s}) \in \mathcal{D}_\mathrm{s}} \left\| W_\theta(y_\mathrm{s}) \cdot \big(T_\tau(x_\mathrm{s}) - E_\theta(y_\mathrm{s})\big) \right\|^2 + \frac{\lambda_\mathrm{s}}{2} \|\tau\|^2, \qquad (5.1)$$

where $\mathcal{D}_\mathrm{s}$ corresponds to the segmentation memory, x_s denotes the segmentation features, y_s the segmentation masks and λ_s is a learnable scalar regularization parameter. The weight predictor W_θ produces sample confidence weights for each spatial location in each memory sample. Applying the optimized model parameters τ^* within the segmentation model produces the mask encoding $x_\mathrm{m} = T_{\tau^*}(x_\mathrm{s})$ for the segmentation features x_s.

LWL [6] feeds the mask encoding directly into the segmentation decoder to produce the segmentation mask. For long and challenging tracking sequences, only relying on the mask encoding may lead to an accurate segmentation mask, but often for the wrong object in the scene (see Fig 5.1). Since LWL [6] is only able to identify the target to a certain degree in challenging tracking sequences, we propose to condition the mask encoding based on an instance localization representation, described next.

5.3.3 *Instance Localization Branch*

The segmentation branch can produce accurate masks but typically lacks the necessary robustness for tracking in medium or long-term sequences. Especially challenging are sequences where objects similar to the target appear,

where the target object is occluded or vanishes from the scene for a short time. Therefore, we propose a dedicated branch for target instance localization, in order to robustly identify the target among distractors or to detect occlusions. A powerful tracking paradigm that learns a target-specific appearance model on both foreground and background information are discriminative correlation filters (DCF) [7, 47, 28, 4]. These methods learn the weights of a filter that differentiates foreground from background pixels represented by a score map, where the maximal value corresponds to the target's center.

Similar to the segmentation branch, we propose an instance localization branch that consists of a sample memory $\mathcal{D}_\mathrm{c}$ and a model predictor P_θ. The latter predicts the parameters κ of the instance model T_κ. The instance model is trained online to produce the target score map used to localize the target object. To obtain the instance model parameters κ we minimize the following loss function

$$L_\mathrm{c}(\kappa) = \sum_{(x_\mathrm{c}, y_\mathrm{c}) \in \mathcal{D}_\mathrm{c}} \big\| R\big(T_\kappa(x_\mathrm{c}), y_\mathrm{c}\big) \big\|^2 + \frac{\lambda_\mathrm{c}}{2} \|\kappa\|^2, \tag{5.2}$$

where $\mathcal{D}_\mathrm{c}$ corresponds to the instance memory containing the classification features x_c and the Gaussian labels y_c. R denotes the robust hinge-like loss [4] and λ_c is a fixed regularization parameter. To solve the optimization problem we apply the method from [4], which unrolls steepest descent iterations of the Gauss-Newton approximation of (5.2) to obtain the final model parameters κ^*. The score map can then be obtained with $s_\mathrm{c} = T_{\kappa^*}(x_\mathrm{c})$ by evaluating the target model on the classification features x_c.

5.3.4 *Instance-Conditional Segmentation Decoder*

In video object segmentation the produced mask encoding is directly fed into the segmentation decoder to generate the segmentation mask. However, solely relying on the mask encoding is not robust enough for the challenging tracking scenario, see Fig 5.1. Thus, we propose to integrate the instance localization information into the segmentation decoding procedure. In particular, we condition the mask encoding on a learned encoding of the instance localization score map.

First, we encode the raw score maps using a multi-layer CNN to learn a suitable representation. Secondly, we condition the mask encoding with the learned representation using element-wise addition. The entire conditioning procedure can be defined as $x_\mathrm{f} = x_\mathrm{m} + H_\theta(s_\mathrm{c})$, where H_θ denotes the CNN encoding the scores s_c, and x_m the mask encoding. The resulting features are

then fed into the segmentation decoder that produces the segmentation scores of the target object.

5.3.5 *Jointly Learning Instance Localization and Segmentation*

In this section, we describe our general training strategy and parameters. In particular, we further detail the segmentation and classification losses that we use for offline training.

Segmentation Loss: First, we randomly sample J frames from an annotated video sequence and sort them according to their frame IDs in increasing order to construct the training sequence $\mathcal{V} = \{(x_{\mathrm{b}}^j, y_{\mathrm{s}}^j, y_{\mathrm{c}}^j)\}_{j=0}^{J-1}$, where $x_{\mathrm{b}}^j = B_\theta(I^j)$ are the extracted features of the video frame I^j using the backbone B_θ, y_{s}^j is the corresponding segmentation mask and y_{c}^j denotes the Gaussian label at the target's center location. We start with entry $v_0 \in \mathcal{V}$ and store it in the segmentation $\mathcal{D}_{\mathrm{s}}$ and instance memory $\mathcal{D}_{\mathrm{c}}$ and obtain parameters τ^0 and κ^0 of the segmentation and instance model. We use these parameters to compute the segmentation loss for $v_1 \in \mathcal{V}$. Using the predicted segmentation mask, we update the segmentation model parameters to τ^1 but keep the instance model parameters fixed. Segmentation parameters typically need to be updated frequently to enable accurate segmentation. Conversely, we train the model predictor on a single frame only. The resulting instance model generalizes to multiple unseen future frames, ensuring robust target localization. The resulting segmentation loss for the entire sequence $\mathcal{V}$ can thus be described as follows

$$\mathcal{L}_{\mathrm{s}}^{\mathrm{seq}}(\theta; \mathcal{V}) = \sum_{j=1}^{J-1} \mathcal{L}_{\mathrm{s}}\Big(D_\theta\Big(T_{\tau^{j-1}}(x_{\mathrm{s}}^j) + H_\theta\big(T_{\kappa^0}(x_{\mathrm{c}}^j)\big)\Big),\ y_{\mathrm{s}}^j \Big), \tag{5.3}$$

where $x_{\mathrm{s}} = F_\theta(x_{\mathrm{b}})$ and $x_{\mathrm{c}} = G_\theta(x_{\mathrm{b}})$ and $\mathcal{L}_{\mathrm{s}}$ is the Lovasz segmentation loss [2].

Classification Loss: Instead of training our tracker only with the segmentation loss, we add an auxiliary loss to ensure that the instance module produces score maps localizing the target via a Gaussian distribution. These score maps are essential to update the segmentation and instance memories and to generate the final output. As explained before, we use only the first training $v_0 \in \mathcal{V}$ to optimize the instance model parameters. To encourage fast convergence, we use not only the parameters corresponding to the final iteration N_{iter} of the optimization method $\kappa^0_{(N_{\mathrm{iter}})}$ explained in Sec. 5.3.3, but also all the intermediate

parameters $\kappa^0_{(i)}$ of loss computation. The final target classification loss for the whole sequence $\mathcal{V}$ is defined as follows

$$\mathcal{L}_{\text{c}}^{\text{seq}}(\theta; \mathcal{V}) = \sum_{j=1}^{J-1} \left(\frac{1}{N_{\text{iter}}} \sum_{i=0}^{N_{\text{iter}}} \mathcal{L}_{\text{c}} \Big(T_{\kappa^0_{(i)}}(x_{\text{c}}^j),\ y_{\text{c}}^j \Big) \right), \tag{5.4}$$

where $\mathcal{L}_{\text{c}}$ is the hinge loss defined in [4]. To train our tracker we combine the segmentation and classification losses using the scalar weight η and minimize both losses jointly

$$\mathcal{L}_{\text{tot}}^{\text{seq}}(\theta; \mathcal{V}) = \mathcal{L}_{\text{s}}^{\text{seq}}(\theta; \mathcal{V}) + \eta \cdot \mathcal{L}_{\text{c}}^{\text{seq}}(\theta; \mathcal{V}). \tag{5.5}$$

Training Details: We use the train sets of LaSOT [35], GOT-10k [52], Youtube-VOS [130] and DAVIS [101]. For VOT datasets that only provide annotated bounding boxes, we use these boxes and STA [141] to generate segmentation masks and treat them as ground truth annotations during training. STA [141] is trained separately on YouTube-VOS 2019 [130] and DAVIS 2017 [98]. For our model, we use ResNet-50 with pre-trained MaskRCNN weights as our backbone and initialize the segmentation model and decoder weights with the ones available from LWL [6]. We train for 200 epochs and sample 15'000 videos per epoch, which takes 96 hours to train on a single Nvidia A100 GPU. We use the ADAM [56] optimizer with a learning rate decay of 0.2 at epochs 25, 115 and 160. We weigh the losses such that the segmentation loss is predominant but in the same range as the classification loss. We empirically choose $\eta = 10$. Further details about training and the network architecture are given in the appendix.

5.3.6 *Inference*

Memory Management and Model Updating: Our tracker consists of two different memory modules. A segmentation memory that stores segmentation features and predicted segmentation masks of previous frames. In contrast, an instance memory contains classification features and Gaussian labels marking the center location of the target in the predicted segmentation mask of the previous video frame. The quality of the predicted labels directly influences the localization and segmentation quality in future video frames. Hence, it is crucial to avoid contaminating the memory modules with predictions that do not correspond to the actual target. We propose the following strategy to keep the memory as clean as possible. (a) If the instance model is able to clearly localize the target (maximum value in the score map larger than $t_{s_c} = 0.3$)

and the segmentation model constructs a valid segmentation mask (at least one pixel above $t_{s_s} = 0.5$) we update both memories with the current predictions and features. (b) If either the instance localization or segmentation fail to identify the target we omit updating the segmentation memory. (c) If only the segmentation mask fails to represent the target but the instance model can localize it, we update the instance memory only. (d) If instance localization fails we do not update either memory. Further, we trigger the few-shot learner and model predictor after 20 frames have passed, but only if the corresponding memory has been updated.

Final Mask Output Generation: We obtain the final segmentation mask by thresholding the segmentation decoder output. To obtain the bounding box required for standard tracking benchmarks, we report the smallest axis-aligned box that contains the entire estimated object mask.

Inference Details: We set the input image resolution such that the segmentation learner features have a resolution of 52×30 (stride 16), while the instance learner operates on features of size 26×15 (stride 32). The learning rate is set to 0.1 and 0.01 for the segmentation and instance learner respectively. We use a maximum buffer of 32 frames for the segmentation memory and 50 frames for the instance memory. We keep the samples corresponding to the initial frame in both memories and replace the oldest entries if the memory is full. We update both memories for the first 100 video frames and afterwards only after every 20^{th} frame. We randomly augment the sample corresponding to the initial frame with vertical flip, random translation and blurring.

5.4 EVALUATION

Our approach is developed within the PyTracking [27] framework. The implementation is done with PyTorch 1.9 with CUDA 11.1. Our model is evaluated on a single Nvidia GTX 2080Ti GPU. Our method achieves an average speed of 30 FPS on LaSOT [35]. Each number corresponds to the average of five runs with different random seeds.

Method	Seg. Branch	Inst. Branch Conditioning	LaSOT [35]			NFS [42]			UAV123 [87]		
			AUC	P	NP	AUC	P	NP	AUC	P	NP
LWL [6]	✓	-	59.7	60.6	63.3	61.5	75.1	76.9	59.7	78.8	71.4
RTS	✓	✗	65.3	68.5	71.5	65.8	84.0	85.0	65.2	85.6	78.8
RTS	✓	✓	69.7	73.7	76.2	65.4	82.8	84.0	67.6	89.4	81.6

Table 5.1 – Comparison between our segmentation network baseline LWL and our pipeline, with and without Instance conditioning on different VOT datasets.

Inst. Branch Fallback	t_{s_c}	LaSOT [35]			NFS [42]			UAV123 [87]		
		AUC	P	NP	AUC	P	NP	AUC	P	NP
✗	0.30	69.3	73.1	75.9	65.3	82.7	84.0	66.3	87.2	80.4
✓	0.30	69.7	73.7	76.2	65.4	82.8	84.0	67.6	89.4	81.6
✓	0.20	68.6	72.3	75.0	65.3	82.7	83.9	67.0	88.7	80.7
✓	0.30	69.7	73.7	76.2	65.4	82.8	84.0	67.6	89.4	81.6
✓	0.40	69.1	72.7	75.6	63.3	79.7	81.7	67.1	89.1	80.7

Table 5.2 – Ablation on inference strategies. The first column analyzes the effect of using the instance branch as fallback for target localization if the segmentation branch is unable to detect the target ($\max(s_s) < t_{s_s}$). The second column shows the impact of different confidence thresholds t_{s_c}.

5.4.1 *Branch Ablation Study*

For the ablation study, we analyze the impact of the instance branch on three datasets and present the results in Tab. 5.1. First, we report the performance of LWL [6] since we build upon it to design our final tracking pipeline. We use the network weights provided by Bhat *et al.* [6] and the corresponding inference settings. We input the same segmentation masks obtained from the initial bounding box for LWL as used for our method. We observe that LWL is not robust enough for challenging tracking scenarios. The second row in Tab. 5.1 corresponds to our method but we omit the proposed instance branch. Hence, we use the proposed inference components and settings and train the tracker as explained in Sec. 5.3.5, but with conditioning removed. We observe that even without the instance localization branch our tracker can achieve competitive performance on all three datasets (*e.g.* +5.6% on LaSOT). Fully integrating the instance localization branch increases the performance even more (*e.g.* +4.4 on LaSOT). Thus, we conclude that adapting the baseline method to the tracking domain improves the tracking performance. To boost the performance and achieve state-of-the-art results, an additional component able to increase the tracking robustness is required.

5.4.2 *Inference Parameters*

In this part, we ablate two key aspects of our inference strategy. First, we study the effect of relying on the instance branch if the segmentation decoder is unable to localize the target ($\max(s_s) < t_{s_s}$). Second, we study different values for t_{s_c} that determines whether the target is detected by the instance model, see Tab. 5.2.

	RTS	ToMP 101 [82]	ToMP 50 [82]	Keep Track [83]	STARK ST-101 [133]	Alpha Refine [132]	TransT [19]	Siam R-CNN [121]	Tr DiMP [123]	Super DiMP [27]	STM Track [40]	Pr DiMP [29]	LWL [6]	DM Track [140]	LTMU [25]	DiMP [4]	Ocean [139]	D3S [77]
Precision	73.7	**73.5**	72.2	70.2	72.2	68.8	69.0	68.4	66.3	65.3	63.3	60.8	60.6	59.7	57.2	56.7	56.6	49.4
Norm. Prec	76.2	**79.2**	**78.0**	77.2	76.9	73.8	73.8	72.2	73.0	72.2	69.3	68.8	63.3	66.9	66.2	65.0	65.1	53.9
Success (AUC)	69.7	**68.5**	67.6	67.1	67.1	65.9	64.9	64.8	63.9	63.1	60.6	59.8	59.7	58.4	57.2	56.9	56.0	49.2
Δ AUC to Ours	-	**↑1.2**	**↑2.1**	**↑2.6**	**↑2.6**	**↑3.8**	**↑4.8**	**↑4.9**	**↑5.8**	**↑6.6**	**↑9.1**	**↑9.9**	**↑10.0**	**↑11.3**	**↑12.5**	**↑12.8**	**↑13.7**	**↑20.5**

Table 5.3 – Comparison to the state of the art on the LaSOT [35] test set in terms of AUC score. The methods are ordered by AUC score.

	RTS	STA [141]	LWL [6]	PrDiMP-50 [29]	DiMP-50 [4]	SiamRPN++ [67]
$SR_{0.50}(\%)$	**94.5**	**95.1**	92.4	89.6	88.7	82.8
$SR_{0.75}(\%)$	**82.6**	**85.2**	82.2	72.8	68.8	-
$AO(\%)$	**85.2**	**86.7**	84.6	77.8	75.3	73.0

Table 5.4 – Results on the GOT-10k validation set [52] in terms of Average Overlap (AO) and Success Rates (SR) for overlap thresholds of 0.5 and 0.75.

If the segmentation branch cannot identify the target, using the instance branch improves tracking performance on all datasets (*e.g.* $+1.3\%$ on UAV123). Furthermore, Tab. 5.2 shows that our tracking pipeline achieves the best performance when setting $t_{s_c} = 0.3$ whereas smaller or larger values for t_{s_c} decrease the tracking accuracy. Hence, it is important to find a suitable trade-off between frequently updating the model and memory to quickly adapt to appearance changes and updating only rarely to avoid contaminating the memory and model based on wrong predictions.

5.4.3 *Comparison to the state of the art*

Assessing segmentation accuracy on tracking datasets is not possible since only bounding box annotations are provided. Therefore, we compare our approach on six VOT benchmarks and validate the segmentation masks quality on two VOS datasets.

LaSOT [35]: We evaluate our method on the test set of the LaSOT dataset, consisting of 280 sequences with 2500 frames on average. Thus, the benchmark challenges the long term adaptability and robustness of trackers. Fig. 5.3 shows the success plot reporting the overlap precision OP with respect to the overlap threshold T. Trackers are ranked by AUC score. In addition, Tab. 5.3 reports the precision and normalized precision for all compared methods. Our method outperforms the state-of-the-art ToMP-50 [82] and ToMP-101 [82] by large margins ($+1.2\%$ and $+2.1\%$ AUC respectively). Our method is not only as robust as KeepTrack (see the success plot for $T < 0.2$) but also estimates far more accurate bounding boxes than any tracker ($0.8 < T < 1.0$).

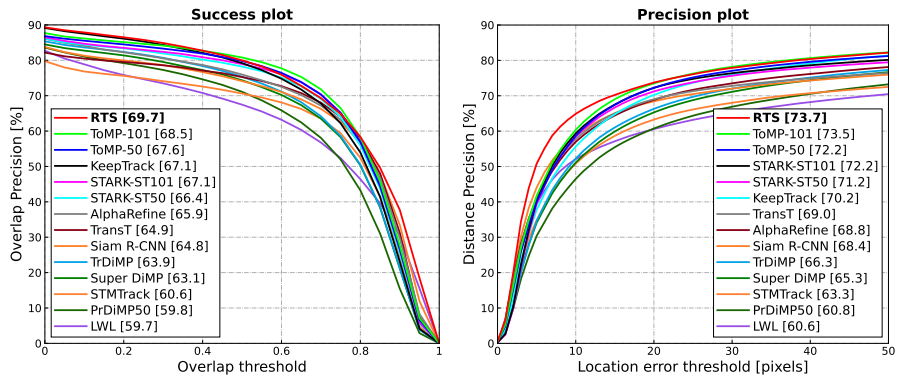


Figure 5.3 – Success (left) and Precision (right) plots on LaSOT [35] with other state-of-the-art methods. The AUCs for all methods are ordered and reported in the legend. Our method outperforms all existing approaches, both in Overlap Precision (left) and Distance Precision (right).

	RTS	ToMP 101 [82]	ToMP 50 [82]	Keep Track [83]	STARK ST101 [133]	STARK ST50 [133]	STA [141]	LWL [6]	TransT [19]	Siam R-CNN [121]	Alpha Refine [132]	STM Track [40]	DTT [135]	Tr DiMP [123]	Super DiMP [27]	Pr DiMP [29]	D3S [77]
Precision	79.4	78.9	78.6	73.8	-	-	79.1	78.4	**80.3**	**80.0**	78.3	76.7	78.9	73.1	73.3	70.4	66.4
Norm. Prec	86.0	86.4	86.2	83.5	**86.9**	86.1	84.7	84.4	**86.7**	85.4	85.6	85.1	85.0	83.3	83.5	81.6	76.8
Success (AUC)	**81.6**	81.5	81.2	78.1	**82.0**	81.3	81.2	80.7	81.4	81.2	80.5	80.3	79.6	78.4	78.1	75.8	72.8
Δ AUC to Ours	-	**↑0.1**	**↑0.4**	**↑3.5**	**↓0.4**	**↑0.3**	**↑0.4**	**↑0.9**	**↑0.2**	**↑0.4**	**↑1.1**	**↑1.3**	**↑2.0**	**↑3.2**	**↑3.5**	**↑5.8**	**↑8.8**

Table 5.5 – Comparison to the state of the art on the TrackingNet [88] test set in terms of AUC scores, Precision and Normalized Precision.

GOT-10k [52]: The large-scale GOT-10k dataset contains over 10.000 shorter sequences. Since we train our method on several datasets instead of only GOT-10k *train*, we evaluate it on the *val* set only, which consists of 180 short videos. We compile the results in Tab. 5.4. Our method ranks second for all metrics, falling between two VOS-oriented methods, $+0.6\%$ over LWL [6] and -1.5% behind STA [141]. Our tracker outperforms other trackers by a large margin.

TrackingNet [88]: We compare our approach on the test set of the TrackingNet dataset, consisting of 511 sequences. Tab. 5.5 shows the results obtained from the online evaluation server. Our method outperforms most of the existing approaches and ranks second in terms of AUC, close behind STARK-ST101 [133] which is based on a ResNet-101 backbone. Note that we outperform STARK-ST50 [133] that uses a ResNet-50 as backbone. Also, we achieve a higher precision score than other methods that produce a segmentation mask output such as LWL [6], STA [141], Alpha-Refine [132] and D3S [77].

UAV123 [87]: The UAV dataset consists of 123 test videos that contain small objects, target occlusion, and distractors. Small objects are particularly chal-

	RTS	ToMP 101 [82]	ToMP 50 [82]	Keep Track [83]	CRACT [36]	STARK ST101 [133]	TrDiMP [123]	TransT [19]	STARK ST50 [133]	Super DiMP [27]	Pr DiMP [29]	STM Track [40]	Siam AttN [137]	Siam R-CNN [121]	KYS [5]	DiMP [4]	LWL [6]
UAV123	67.6	66.9	69.0	**69.7**	66.4	68.2	67.5	**69.1**	**69.1**	67.7	68.0	64.7	65.0	64.9	–	65.3	59.7
NFS	65.4	**66.7**	**66.9**	66.4	62.5	66.2	66.2	65.7	65.2	64.8	63.5	–	–	63.9	63.5	62.0	61.5

Table 5.6 – Comparison with state-of-the-art on the UAV123 [87] and NFS [42] datasets in terms of AUC score.

	RTS	STARK ST-50 +AR [133]	STARK-ST-101-+AR [133]	LWL [59]	STA [141]	Ocean Plus [59]	Fast Ocean [59]	Alpha Refine [59]	RPT [59]	AFOD [59]	D3S [59]	STM [59]
Robustness	**0.845**	0.817	0.789	0.798	0.824	0.842	0.803	0.777	**0.869**	0.795	0.769	0.574
Accuracy	0.710	**0.759**	**0.763**	0.719	0.732	0.685	0.693	0.754	0.700	0.713	0.699	0.751
EAO	0.506	0.505	0.497	0.463	**0.510**	0.491	0.461	0.482	**0.530**	0.472	0.439	0.308
Δ EAO to Ours	-	**↑0.001**	**↑0.009**	**↑0.043**	**↓0.004**	**↑0.015**	**↑0.045**	**↑0.024**	**↓0.024**	**↑0.034**	**↑0.067**	**↑0.198**

Table 5.7 – Results on the VOT2020-ST [59] challenge in terms of Expected Average Overlap (EAO), Accuracy and Robustness.

lenging in a segmentation setup. Tab. 5.6 shows the achieved results in terms of success AUC. Our method achieves competitive results on UAV123, close to TrDiMP [123] or SuperDiMP [27]. It outperforms LWL [6] by a large margin.

NFS [42]: The NFS dataset (30FPS version) contains 100 test videos with fast motions and challenging sequences with distractors. Our method achieves an AUC score that is only 1% below the current best method KeepTrack [83] while outperforming numerous other trackers, including STARK-ST50 [133] (+0.2) SuperDiMP [4] (+0.6) and PrDiMP [29] (+1.9).

VOT 2020 [59]: Finally, we evaluate our method on the VOT2020 short-term challenge. It consists of 60 videos and provides segmentation mask annotations. For the challenge, the multi-start protocol is used and the tracking performance is assessed based on accuracy and robustness. We compare with the top methods on the leader board and include more recent methods in Tab. 5.7. In this setup, our method ranks 2nd in Robustness, thus outperforming most of the other methods. In particular, we achieve a higher EAO score than STARK [133], LWL [6], AlphaRefine [132] and D3S [77].

YouTube-VOS 2019 [130]: We use the validation set which consist of 507 sequences. They contain 91 object categories out of which 26 are *unseen* in the training set. The results presented in Tab. 5.8 were generated by an online server after uploading the raw results. On this benchmark, we want to validate the quality of the produced segmentation masks rather than to achieve the best accuracy possible. Hence, we use the same model weight as for VOT without further fine tuning.

Method	YouTube-VOS 2019 [130]					DAVIS 2017 [101]		
	$\mathcal{G}$	$\mathcal{J}_{\text{seen}}$	$\mathcal{J}_{\text{unseen}}$	$\mathcal{F}_{\text{seen}}$	$\mathcal{F}_{\text{unseen}}$	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
RTS	79.7	77.9	75.4	82.0	83.3	80.2	77.9	82.6
LWL [6]	81.0	79.6	76.4	83.8	84.2	81.6	79.1	84.1
STA [141]	80.6	-	-	-	-	-	-	-
STM [92]	79.2	79.6	73.0	83.6	80.6	81.8	79.2	84.3
RTS (Box)	70.8	71.1	65.2	74.0	72.8	72.6	69.4	75.8
LWL (Box) [6]	-	-	-	-	-	70.6	67.9	73.3
Siam-RCNN [121]	67.3	68.1	61.5	70.8	68.8	70.6	66.1	75.0
D3S [125]	-	-	-	-	-	60.8	57.8	63.8
SiamMask [77]	52.8	60.2	45.1	58.2	47.7	56.4	54.3	58.5

Table 5.8 – Results on the Youtube-VOS 2019 [130] and DAVIS 2017 [101] datasets. The table is split in two parts to separate methods using bounding box initialization or segmentation masks initialization, in order to enable a fair comparison.

When using the provided segmentation masks for initialization, we observe that our method performs slightly worse than LWL [6] and STA [141] (-1.3 $\mathcal{G}$, -0.9 $\mathcal{G}$) but still outperforms the VOS method STM [92] (+0.5 $\mathcal{G}$). We conclude that our method can generate accurate segmentation masks. When using bounding boxes to predict both the initialization and segmentation masks, we outperform all other methods by a large margin. This confirms that even with our bounding-box initialization strategy, RTS produces accurate segmentation masks.

DAVIS 2017 [101]: Similarly, we compare our method on the validation set of DAVIS 2017 [101], which contains 30 sequences. We do not fine tune the model for this benchmark. The results are shown in Tab. 5.8 and confirm the observation made above that RTS is able to generate accurate segmentation masks. Our method is competitive in the mask-initialization setup. In the box-initialization setup however, our approach outperforms all other methods in $\mathcal{J}\&\mathcal{F}$, in particular the segmentation trackers like SiamMask [125] (+16.2) and D3S [77] (+11.8).

5.5 CONCLUSION

We introduced RTS, a robust, end-to-end trainable, segmentation-driven tracking method that is able to generate accurate segmentation masks. Compared to the traditional bounding box outputs of classical visual object trackers, segmentation masks enable a more accurate representation of the target's shape and extent. The proposed instance localization branch helps increasing the robustness of our tracker to enable reliable tracking even for long sequences of thousands of frames. Our method outperforms previous segmentation-driven

tracking methods by a large margin, and it is competitive on several VOT benchmarks. In particular, we set a new state of the art on the challenging LaSOT [35] dataset with a success AUC of 69.7%. Competitive results on two VOS datasets confirm the high quality of the generated segmentation masks.

APPENDICES

In these appendices, we provide further details on various aspects of our tracking pipeline. First, we provide additional architectural and inference details in Sections 5.A and 5.B. Second, we provide additional ablation studies, in particular on the loss weighting parameter η on different benchmarks to show the importance of the auxiliary instance localization loss in Section 5.C. Then, we provide success plots for different VOT benchmarks as well as a detailed analysis of our results on LaSOT [35] by comparing our approach against the other state-of-the-art methods for all the dataset attributes in Section 5.D. Finally, we provide some additional visual comparison to other trackers on Figure 5.8.

5.A ADDITIONAL ARCHITECTURE DETAILS

Classification Scores Encoder H_θ: First, we describe in Figure 5.4 the architecture of the Classification Scores Encoder H_θ. It takes as input the $H \times W$-dimensional scores predicted by the Instance Localization (*Classification*) branch and outputs a 16 channels deep representation of those scores. The score encoder consists of a convolutional layer followed by a max-pool layer with stride one and two residual blocks. The output of the residual blocks has 64 channels. Thus, the final convolutional layer reduces the number of channels of the output to 16 to match the encoded scores with the mask encoding. All the convolutional layers use (3×3) kernels with a stride of one to preserve the spatial size of the input classification scores.

Segmentation Decoder D_θ: The segmentation decoder has the same structure has in LWL [6]. Together with the backbone, it shows a U-Net structure and mainly consists of four decoder blocks. It takes as input the extracted ResNet-50 backbone features and the combined encoding x_f from both the instance localization branch ($H_\theta(s_\mathrm{c})$) and the segmentation branch (x_m), with $x_\mathrm{f} = x_\mathrm{m} + H_\theta(s_\mathrm{c})$. Since the encoded instance localization scores have a lower spatial resolution than the mask encoding x_m, we upscale the encoded instance localization scores using a bilinear interpolation before adding it with the mask encoding x_m. We refer the reader to [6] for more details about the decoder structure.

Segmentation Branch: We use the same architectures for the feature extractor F_θ, the label encoder E_θ, the weight predictor W_θ, the few-shot learner A_θ and

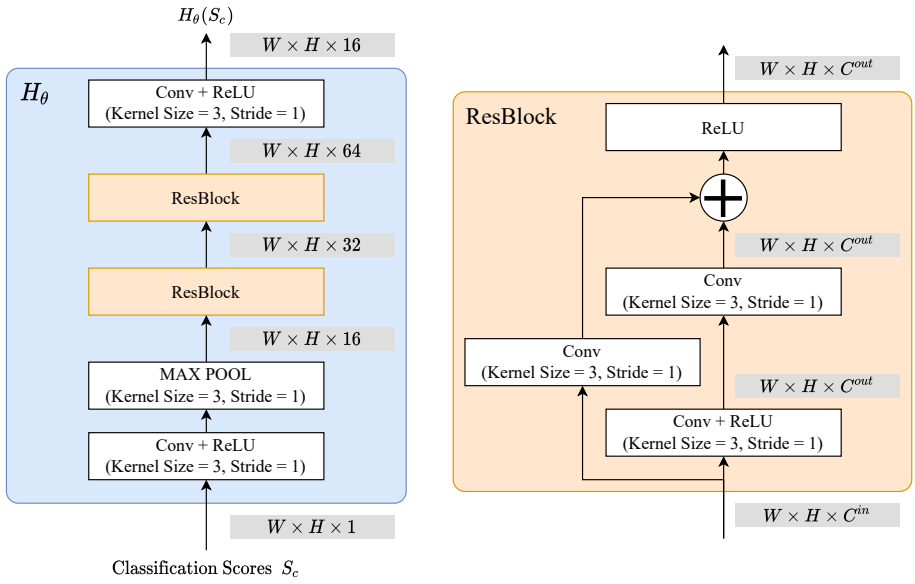


Figure 5.4 – Classification Scores Encoder H_θ.

the segmentation model T_τ as proposed in LWL [6]. Hence, we refer the reader to [6] for more details.

Instance Localization Branch: We use the same architectures for the feature extractor G_θ, the model predictor P_θ and the instance model T_κ as proposed in DiMP [4]. Hence, we refer the reader to [4] for more details.

5.B ADDITIONAL INFERENCE DETAILS

Search region selection: The backbone does not extract features on the full image. Instead, we sample a smaller image patch for extraction, which is centered at the current target location and 6 times larger than the current estimated target size, when it does not exceed the size of the image. The estimation of the target state (position and size) is therefore crucial to ensure an optimal crop. In most situations, the segmentation output is used to determine the target state since it has a high accuracy. The *target center* is computed as the center of mass of the predicted per-pixel segmentation probability scores. The *target size* is computed as the variance of the segmentation probability scores.

If the segmentation branch cannot find the target (as described in this chapter), but the instance branch still outputs a high enough confidence score, we use it to update the target position. This is particularly important in sequences where

the target is becoming too small for some time, but we can still track the target position.

When both branch cannot find the target, the internal state of the tracker is not updated. We upscale the search area based on the previous 60 valid predicted scales. This is helpful in situations where the size of the object shrinks although its size does not change. This typically happens during occlusions, or if the target goes out of the frame partially or completely.

5.C ADDITIONAL ABLATIONS

In this section, we provide additional ablation studies related to our method, first on the weighting of the segmentation and classification losses used for training, second on the parameters that might make a difference specifically for VOS benchmarks like Youtube-VOS [130].

Weighting segmentation and classification losses: For this ablation, we study the weighting of the segmentation loss $\mathcal{L}_{\text{s}}$ and the instance localization loss $\mathcal{L}_{\text{c}}$ in the total loss $\mathcal{L}_{\text{tot}}$. It used to train our model and its influence on the overall performance during tracking. We recall that

$$\mathcal{L}_{\text{tot}} = \mathcal{L}_{\text{s}} + \eta \cdot \mathcal{L}_{\text{c}}. \tag{5.6}$$

Table 5.9 shows the results when training the tracker with three different values of η on five VOT datasets. First, we examine the case where we omit the auxiliary instance localization loss ($\eta = 0.0$). This means that the whole pipeline is trained for segmentation and the instance branch is not trained to produce specifically accurate localization scores. We observe that this setting leads to the lowest performance on all tested datasets, often by a large margin. Secondly, we test a dominant segmentation loss ($\eta = 0.4$), since the segmentation branch needs to be trained for a more complex task than the instance branch. We see a performance gain for almost all datasets. Thus, employing the auxiliary loss to train the instance localization branch helps to improve the tracking performance. We observe that using the auxiliary loss leads to localization scores generated during inference that are sharper, cleaner and localize the center of the target more accurately. Finally, we put an even higher weight on the classification term ($\eta = 10$). This setup leads to an even more accurate localization, and leads to the best results on average. Thus, we set $\eta = 10$ to train our tracking pipeline.

Fine-tuning on Youtube-VOS [35]: In this section, we analyze whether we can gear our pipeline towards VOS benchmarks. To do that, we take our model and inference parameters, and modify them slightly. On the one hand, the model

η	LaSOT [35] AUC	GOT-10k [52] AO	TrackingNet [88] AUC	NFS [42] AUC	UAV123 [87] AUC
0.0	67.7	84.0	81.2	63.7	64.7
0.4	69.8	84.0	81.4	66.2	67.4
10	69.7	85.2	81.6	65.4	67.6

Table 5.9 – Ablation on the classification vs. segmentation loss weighting on different datasets in terms of AUC (area-under-the-curve) and AO (average overlap)

.

		YouTube-VOS 2019 [130]				**DAVIS 2017 [101]**		
Method	$\mathcal{G}$	$\mathcal{J}_{\text{seen}}$	$\mathcal{J}_{\text{unseen}}$	$\mathcal{F}_{\text{seen}}$	$\mathcal{F}_{\text{unseen}}$	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
RTS	79.7	77.9	75.4	82.0	83.3	80.2	77.9	82.6
RTS (YT-FT)	80.3	78.8	76.2	82.9	83.5	80.3	77.7	82.9
LWL [6]	81.0	79.6	76.4	83.8	84.2	81.6	79.1	84.1
STA [141]	80.6	-	-	-	-	-	-	-
STM [92]	79.2	79.6	73.0	83.6	80.6	81.8	79.2	84.3

Table 5.10 – Results on the Youtube-VOS 2019 [130] and DAVIS 2017 [101] datasets with a fined tuned model and inference parameters referred as *RTS (YT-FT)*.

is fined-tuned for 50 epochs using Youtube-VOS [130] only for both training and validation. We also increase the initialization phase from 100 to 200 frames, and remove the relative target scale change limit from one frame to the next. In our standard model, we limit that scale change to 20% for increased robustness.

The results are presented in Table 5.10 for Youtube-VOS [130] and Davis [101]. We observe that the performances between both of our models stay very close for Davis, but that the fine-tuned model is getting closer to the baseline LWL [6] for Youtube-VOS. The more frequent updates seem to help, and not restricting the scale change of objects from frame to frames seems to play a role, since we get an improvement of 0.6 in $\mathcal{G}$ score.

5.D ADDITIONAL EVALUATION RESULTS

In this section, we provide additional plots of our approach on different benchmarks, and a attribute analysis on LaSOT [35].

Success plots for LaSOT [35], NFS [42] and UAV123 [87]: We provide in Figure 5.5 all the plots for the metrics we report for LaSOT [35] in the chapter: *Success*, *Normalized Precision* and *Precision* plots. For completeness, we provide the success plots for NFS [42] and UAV123 [87] in Figure 5.6.

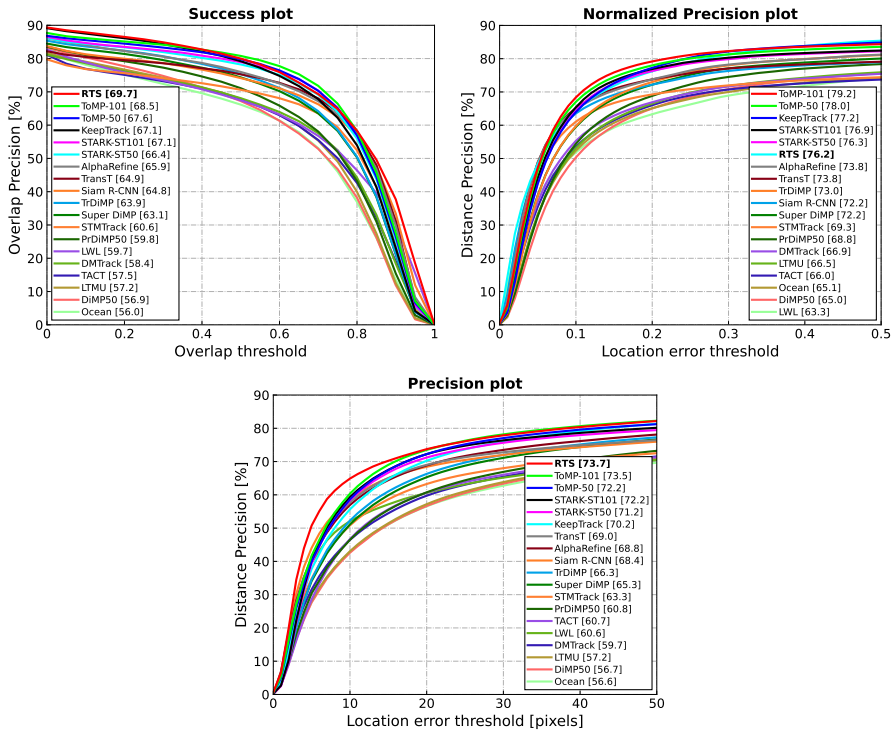


Figure 5.5 – Success, precision and normalized precision plots on LaSOT [35]. Our approach outperforms all other methods by a large margin in AUC, reported in the legend.

Attribute analysis on LaSOT [35]: In this section, we focus on the dataset sequences attributes. We compare our approach to numerous other trackers, and provide the detailed results in Table 5.11. Furthermore, we highlight the strength of our approach in Figure 5.7 by focusing the comparison only to the two current state-of-the-art methods ToMP-101 and ToMP-50 [82].

There are 14 attributes provided for LaSOT [35] sequences, representing different kind of challenges the tracker has to deal with in different situations. Compared to existing trackers, our method achieves better AUC scores in 11 out of 14 attributes. In particular, we outperform ToMP-50 [82] and ToMP-101 [82] by a large margin for the following attributes: *Camera Motion* (+4.2% and +2.7%), *Scale Variation* (+1.8% and 0.9%), *Deformation* (+3.1% and +2.2%), *Motion Blur* (+3.1% and +2.5%) and *Aspect Ratio Change* (+1.7% and +1.0%). Our method is only outperformed on three attributes by KeepTrack [83] and ToMP [82] for *Fast Motion* (-2.3% to -4.1%) and for *Illumination Variation* (-0.3% to -1.4%). For *Background Clutter*, RTS outperforms ToMP-50 [82] by 2.3% and fall just behind ToMP-101 [82] (-0.1%).

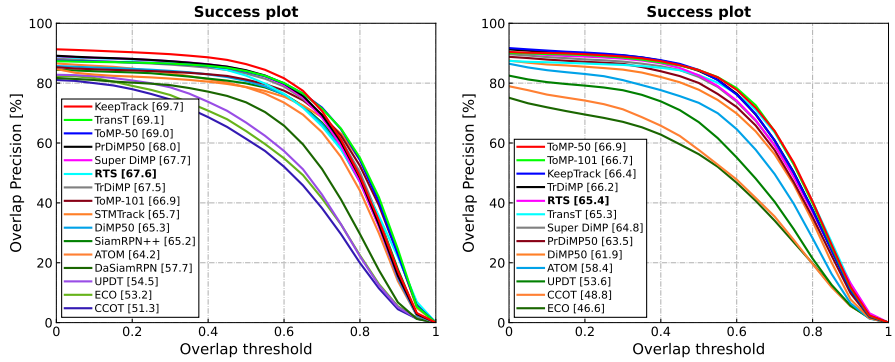


Figure 5.6 – Success plots on the UAV123 [87] (left) and NFS [42] (right) datasets in terms of overall AUC score, reported in the legend.

	Illumination Variation	Partial Occlusion	Deformation	Motion Blur	Camera Motion	Rotation	Background Clutter	Viewpoint Change	Scale Variation	Full Occlusion	Fast Motion	Out-of-View	Low Resolution	Aspect Ratio Change	Total
LTMU [25]	56.5	54.0	57.2	55.8	61.6	55.1	49.9	56.7	57.1	49.9	44.0	52.7	51.4	55.1	57.2
LWL [6]	65.3	56.4	61.6	59.1	64.7	57.4	53.1	58.1	59.3	48.7	46.5	51.5	48.7	57.9	59.7
PrDiMP50 [29]	63.7	56.9	60.8	57.9	64.2	58.1	54.3	59.2	59.4	51.3	48.4	55.3	53.5	58.6	59.8
STMTrack [40]	65.2	57.1	64.0	55.3	63.3	60.1	54.1	58.2	60.6	47.8	42.4	51.9	50.3	58.8	60.6
SuperDiMP [4]	67.8	59.7	63.4	62.0	68.0	61.4	57.3	63.4	62.9	54.1	50.7	59.0	56.4	61.6	63.1
TrDiMP [123]	67.5	61.1	64.4	62.4	68.1	62.4	58.9	62.8	63.4	56.4	53.0	60.7	58.1	62.3	63.9
Siam R-CNN [121]	64.6	62.2	65.2	63.1	68.2	64.1	54.2	65.3	64.5	55.3	51.5	62.2	57.1	63.4	64.8
TransT [19]	65.2	62.0	67.0	63.0	67.2	64.3	57.9	61.7	64.6	55.3	51.0	58.2	56.4	63.2	64.9
AlphaRefine [132]	69.4	62.3	66.3	65.2	70.0	63.9	58.8	63.1	65.4	57.4	53.6	61.1	58.6	64.1	65.3
KeepTrack Fast [83]	**70.1**	63.8	66.2	65.0	70.7	65.1	60.1	67.6	66.6	59.2	57.1	63.4	62.0	65.6	66.8
KeepTrack [83]	**69.7**	64.1	67.0	**66.7**	71.0	65.3	61.2	66.9	66.8	**60.1**	**57.7**	**64.1**	62.0	65.9	67.1
STARK-ST101 [133]	67.5	65.1	68.3	64.5	69.5	66.6	57.4	68.8	66.8	58.9	54.2	63.3	59.6	65.6	67.1
ToMP-50 [82]	66.8	64.9	68.5	64.6	70.2	67.3	59.1	67.2	67.5	59.3	56.1	63.7	61.1	66.5	67.6
ToMP-101 [82]	69.0	**65.3**	**69.4**	65.2	**71.7**	**67.8**	**61.5**	**69.2**	**68.4**	59.1	**57.9**	**64.1**	**62.5**	**67.2**	**68.5**
RTS	68.7	**66.9**	**71.6**	**67.7**	**74.4**	**67.9**	**61.4**	**69.7**	**69.3**	**60.5**	53.8	**66.3**	**62.7**	**68.2**	**69.7**

Table 5.11 – LaSOT [35] attribute-based analysis. Each column corresponds to the results computed on all sequences in the dataset with the corresponding attribute. Our method outperforms all others in 12 out of 14 attributes.

Visual results on LaSOT [35]: Figure 5.8 shows additional visual results compared to other state-of-the-art trackers on 6 different sequences of LaSOT [35].

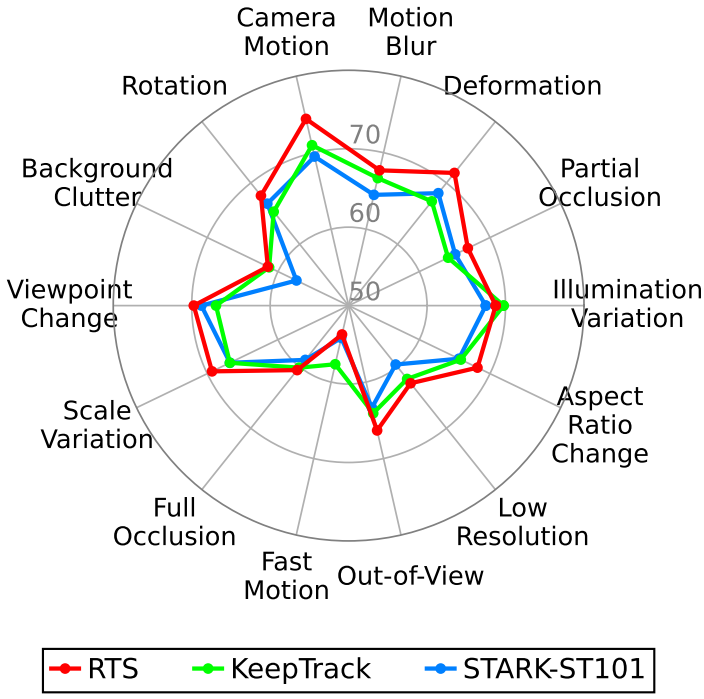


Figure 5.7 – Attributes comparison on LaSOT [35].

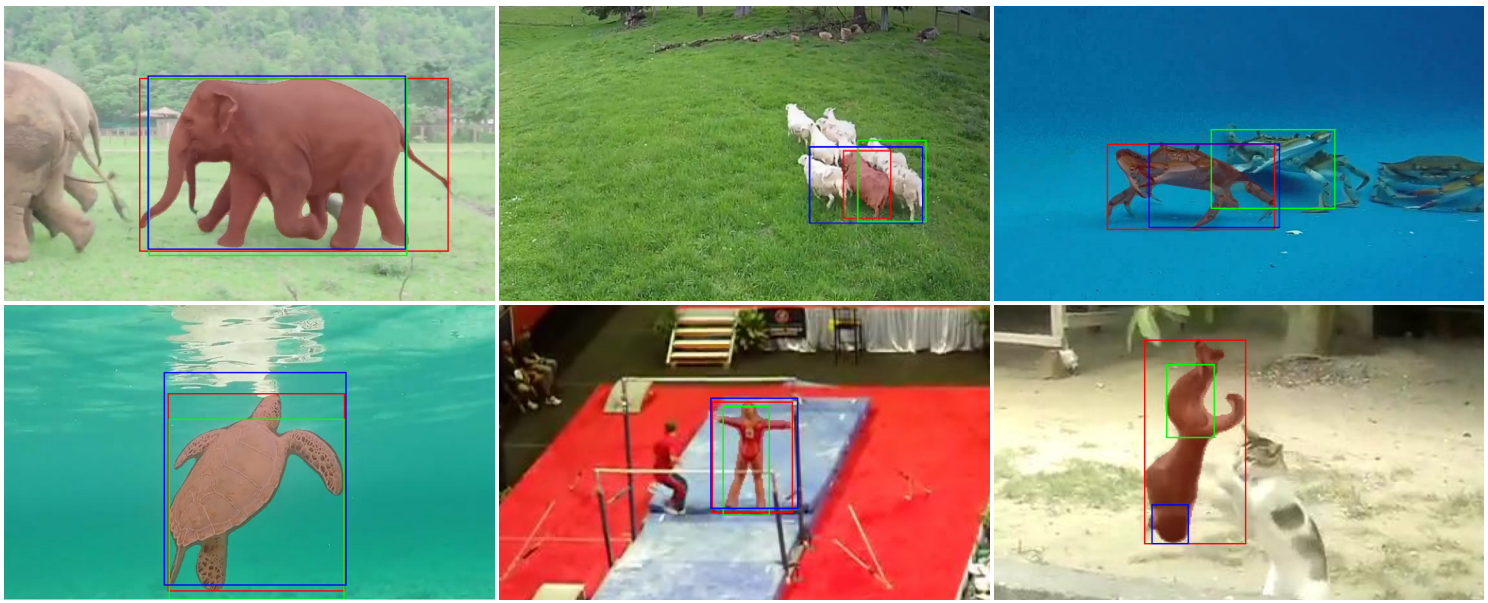


Figure 5.8 – Qualitative results on LaSOT [35] of our approach compared to the previous state-of-the-art methods KeepTrack [83] and STARK-ST101 [133]. As they do not produce segmentation masks, we represent ours as a red overlay and print for all methods the predicted bounding boxes with the following color code: KeepTrack STARK-ST101 RTS

6 DISCUSSION

6.1 SUMMARY OF CONTRIBUTIONS

In this thesis, we explored several approaches to using spatio-temporal information to improve video scene understanding, in the scope of Semantic Segmentation and Visual Object Tracking and Segmentation. The first part of the thesis presented architectures to convert single-frame semantic segmentation networks into video semantic segmentation pipelines. To that end, chapter 3 relied on optical flow to propagate semantic features and focused on decreasing the processing time for each frame. Chapter 4 aimed at improving the segmentation quality by directly storing and matching semantic features from previous frames in a dedicated memory that can be used through attention mechanisms.

The second part of the thesis focused on VOT and its relationship with VOS in chapter 5. We proposed to put the segmentation at the core of the tracking architecture in contrast with most existing methods and to combine it with a simpler localization task to increase its robustness. The resulting pipeline showed state-of-the-art performances and hopefully paved the way for different approaches in VOT.

Chapter 3 presented EVS [96], which proposed to balance the computation load both on the GPU and on the CPU in parallel, to push the standards in terms of speed. Based on a fast optical flow method, it additionally introduced a lightweight module to guide and refine the propagation of semantic information. Our design offers several operating points to trade off segmentation quality versus frame rate and showed substantial speed improvements at a relatively small cost in segmentation. Overall, we showed the benefits of balancing the computation between GPU and CPU.

Chapter 4 presented LMANet [94], which demonstrated how to improve the accuracy of a single-frame semantic segmentation method with a novel, generic and simple structure. Our approach is based on explicitly modeling the semantic information aggregated from previous frames matched in space and time to improve the current frame segmentation. LMANet [94] successfully increased the segmentation score from two popular segmentation networks by almost 2% on the popular Cityscapes [24] dataset, while preserving their frame rates.

Chapter 5 introduced RTS [95], a robust segmentation-driven tracking pipeline that is end-to-end trainable and capable of generating segmentation masks. Our method relies on a dual-branch architecture with distinct feature spaces and memories to better optimize meta parameters for each task: localization and segmentation. Having segmentation masks at the core of the pipeline enables the tracker to learn a richer representation of the targets. Our approach showed state-of-the-art results on several popular VOT benchmarks, even outperforming by a large margin more recent transformer-based approaches on LaSOT [35], with an AUC score of 69.7%.

6.1.1 *Open-Source Contributions*

Following the trend of most research papers in computer vision nowadays, the source code for chapters 4 and 5 has been made public on GitHub. This should help other researchers to access all the details of the proposed methods both in terms of parameters and practical implementation. Moreover, open-sourced methods should reduce significantly the time needed to reproduce results and develop new approaches on top of them, thus accelerating the pace of research in that field. In particular, we contributed online to the following repositories:

— **https://github.com/mattpfr/lmanet**
This repository contains the code to train and test LMANet [94] described in chapter 4. It can also be used out of the box, as pre-trained models are also provided.

— **https://github.com/visionml/pytracking**
The repository contains the implementation and the scripts to train, test, debug, and visualize RTS [95] described in chapter 5. Pre-trained models and a compilation of results are also available at this address. Note that several other popular visual object trackers and visual object segmentation models are available at this URL.

6.2 LIMITATIONS AND POSSIBLE FUTURE RESEARCH

In this thesis, we proposed a few methods to improve existing single-frame semantic segmentation networks in the context of videos. We also proposed a unique framework to bridge the gap between Visual Object Tracking and Video Object Segmentation. In this section, we discuss the limitations of the presented works and open the discussion on possible research directions that could be pursued in the future.

6.2.1 *EVS*

Chapter 3 explored the simple, yet powerful idea to use optical flow to propagate semantic information across frames. As we focused on pushing the boundaries of video semantic segmentation in terms of frame rate, we selected DIS [61], an optical flow that could run extremely fast on the CPU, together with an efficient single-frame semantic segmentation network, ICNet [142]. Although our approach can conceptually and practically always be applied, we can highlight three main limitations.

First, our method showed interesting results when it was written, but it relied on two other components which could be replaced by other and newer approaches. It would indeed be interesting to study more combinations of optical flow and semantic segmentation methods since this research field moves quite quickly and the trade-offs exposed in our work might be different now.

Second, this chapter showed that working toward increasing inference speed is quite difficult in research. This is because, on the one hand, it requires being thorough when measuring and reporting time, because it depends on a lot of factors: CPU type, GPU model, current CPU load, library versions, tools used to measure, border effects, etc. On the other hand, comparing results against other works in a fair way either requires having all the details about the procedure used by other teams, or reproducing these methods locally. These difficulties limit the reasoning beyond the scope of those works and their assumptions, compared to industry applications.

Third, there seems to be a global trend toward providing means to boost inference times for models deployed in the industry for real-world applications. For instance, on the hardware side, dedicated chips for optical flow computation are now embedded in GPUs, which could replace in our case the optical flow running on the CPU. Generally, the usage of dedicated hardware chips for Deep Learning pipelines is becoming more and more common. Besides, from the software side, the use of the TensorRT SDK and the advancements in the optimizations made for deployment can substantially reduce inference times once a model is deployed. This is adding another layer of complexity when it comes to measuring the actual final performance.

6.2.2 *LMANet*

Chapter 4 presented a novel architecture to transform a single-frame semantic segmentation model into a video segmentation pipeline. This architecture, although simple and generic, showed an interesting boost in segmentation

accuracy, at a minimal cost in terms of inference time. We believe that it could be interesting to refresh and continue this work in two different ways.

First, attention mechanisms are now quite commonly used for various tasks and it would make sense to consider the best practices from other newer methods. In particular, it would be interesting to review and experiment with different ways of representing semantic features, as well as improving querying and matching mechanisms. The fusion module is also a key component that could be refined to improve the quality of the segmentation output.

Second, it would be a good idea to study how our approach behaves with more recent semantic segmentation models for single frames. Hopefully, using a better baseline would mean having semantic features of better quality that could further improve the final segmentation result.

6.2.3 *RTS*

Chapter 5 introduced a state-of-the-art visual object tracking pipeline that places segmentation at its core instead of having it as a side component or a post-processing step. Although its segmentation-centric approach is fairly rare in the literature, with only two older methods [77, 125] using similar approaches, it proved to be very promising. Indeed, we showed that RTS [95] could compete head-to-head with other trackers, in some cases outperforming methods with bigger backbones and transformer-based architectures [82, 133]. In several benchmarks, we observed that our method could not only predict more accurate bounding boxes and richer segmentation outputs but also remain robust across very long videos. Therefore, we hope that our method will help pave the way for more segmentation-centric approaches and slowly bridge the gap between Visual Object Tracking and Video Object Segmentation.

Going forward, we believe our method could be improved by tackling its inherent limitations while preserving the segmentation-centric approach. We identify three main research directions that could be interesting to follow and lead to improvements.

One of the limitations of our architecture is the multi-objects handling: in the current version, each object is processed separately, which means that in scenarios where multiple targets of interest are present, the computational cost per frame can quickly become prohibitive. Future works would need to explore the same concept with multi-objects segmentation and tracking, especially how to store and relate features from different targets. This would potentially help both processing time and distractors tracking, since currently, similar objects (*distractors*) can be a source of mistakes and lead to losing the target completely.

The second biggest limitation of our architecture is when it "loses" the target, which can occur for various reasons: distractors are close to the real target, the camera motion is challenging and introduces substantial changes in appearance, the target is occluded for some time, the target is too fast, the object goes out of the camera view, etc. In the current version, our method handles the situation by a rather simplistic strategy that consists in not updating its internal representation of the target, and waiting until it reappears in the same region where it was lost. Although this is a reasonable strategy for different scenarios, it is vulnerable to distractors in the scene and can lead to a permanent loss of tracking if the object goes out of the search region or re-appears somewhere else in the image. A global mechanism for re-initialization would be needed to handle such scenarios and recover the target in more cases.

Finally, it would be very interesting to review newer architectures for both tracking and segmentation. Visual Object Tracking and Video Object Segmentation are two extremely dynamic fields that also saw big improvements coming from transformer-based architectures. We believe that using more up-to-date components and techniques from VOS and VOT, and considering the previous limitations while designing a new architecture could yield interesting results.

BIBLIOGRAPHY

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 39 (2016), pp. 2481–2495 (cit. on pp. 14, 27).

[2] Maxim Berman, Amal Rannen Triki, and Matthew B. Blaschko. "The Lovász-Softmax Loss: A Tractable Surrogate for the Optimization of the Intersection-Over-Union Measure in Neural Networks." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on p. 62).

[3] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip HS Torr. "Fully-Convolutional Siamese Networks for Object Tracking." In: *Proceedings of the European Conference on Computer Vision Workshops (ECCVW)*. 2016 (cit. on p. 55).

[4] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. "Learning Discriminative Model Prediction for Tracking." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019 (cit. on pp. 55, 58, 61, 63, 66, 68, 72, 76).

[5] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. "Know Your Surroundings: Exploiting Scene Information for Object Tracking." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020 (cit. on p. 68).

[6] Goutam Bhat, Felix Järemo Lawin, Martin Danelljan, Andreas Robinson, Michael Felsberg, Luc Van Gool, and Radu Timofte. "Learning What to Learn for Video Object Segmentation." In: *European Conference on Computer Vision ECCV*. 2020 (cit. on pp. 6, 54, 56–58, 60, 63–69, 71, 72, 74, 76).

[7] David S. Bolme, J. Ross Beveridge, Bruce A. Draper, and Yui Man Lui. "Visual object tracking using adaptive correlation filters." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2010 (cit. on pp. 55, 61).

[8] Adam Botach, Evgenii Zheltonozhskii, and Chaim Baskin. "End-to-End Referring Video Object Segmentation With Multimodal Transformers." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 4985–4995 (cit. on p. 6).

[9] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. "Semantic object classes in video: A high-definition ground truth database." In: *Pattern Recognition Letters* 30 (2009), pp. 88–97 (cit. on pp. 16, 32).

[10] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. "Segmentation and Recognition Using Structure from Motion Point Clouds." In: *The European Conference on Computer Vision (ECCV)*. 2008 (cit. on pp. 14, 32).

[11] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. "A naturalistic open source movie for optical flow evaluation." In: *The European Conference on Computer Vision (ECCV)*. 2012 (cit. on pp. 2, 16, 32).

[12] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. "One-shot video object segmentation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 221–230 (cit. on p. 57).

[13] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. "End-to-End Object Detection with Transformers." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. 2020 (cit. on p. 33).

[14] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. "Emerging Properties in Self-Supervised Vision Transformers." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 9650–9660 (cit. on pp. 2, 6).

[15] A. Y. C. Chen and J. J. Corso. "Temporally consistent multi-class video-object segmentation with the Video Graph-Shifts algorithm." In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2011 (cit. on p. 32).

[16] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 40.4 (2018), pp. 834–848 (cit. on p. 13).

[17] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation." In: *The European Conference on Computer Vision (ECCV)*. 2018 (cit. on p. 14).

[18] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation." In: *The European Conference on Computer Vision (ECCV)*. 2018 (cit. on p. 45).

[19] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. "Transformer Tracking." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021 (cit. on pp. 57, 66–68, 76).

[20] Yuhua Chen, Wen Li, Xiaoran Chen, and Luc Van Gool. "Learning Semantic Segmentation From Synthetic Data: A Geometrically Guided Input-Output Adaptation Approach." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 (cit. on p. 2).

[21] Yuhua Chen, Jordi Pont-Tuset, Alberto Montes, and Luc Van Gool. "Blazingly fast video object segmentation with pixel-wise metric learning." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 1189–1198 (cit. on p. 57).

[22] Zhiming Chen, Kean Chen, Weiyao Lin, John See, Hui Yu, Yan Ke, and Cong Yang. "Piou loss: Towards accurate oriented object detection in complex environments." In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*. Springer. 2020, pp. 195–211 (cit. on p. 7).

[23] J. Cheng, Y.-H. Tsai, S. Wang, and M.-H. Yang. "SegFlow: Joint Learning for Video Object Segmentation and Optical Flow." In: *IEEE International Conference on Computer Vision (ICCV)*. 2017 (cit. on p. 16).

[24] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. "The Cityscapes Dataset for Semantic Urban Scene Understanding." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cit. on pp. 2, 3, 12, 16, 17, 21, 23, 24, 30, 32, 39, 42, 47, 48, 79).

[25] Kenan Dai, Yunhua Zhang, Dong Wang, Jianhua Li, Huchuan Lu, and Xiaoyun Yang. "High-Performance Long-Term Tracking With Meta-Updater." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 (cit. on pp. 66, 76).

[26] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. "ATOM: Accurate Tracking by Overlap Maximization." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 (cit. on pp. 55, 58).

[27] Martin Danelljan, Goutam Bhat, Christoph Mayer, and Matthieu Paul. *PyTracking: Visual tracking library based on PyTorch.* `https://github.com/visionml/pytracking`. 2019 (cit. on pp. 64, 66–68).

[28] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. "ECO: Efficient Convolution Operators for Tracking." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on p. 61).

[29] Martin Danelljan, Luc Van Gool, and Radu Timofte. "Probabilistic Regression for Visual Tracking." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 (cit. on pp. 55, 57, 66–68, 76).

[30] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. "Beyond Correlation Filters: Learning Continuous Convolution Operators for Visual Tracking." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2016 (cit. on p. 55).

[31] Achal Dave, Tarasha Khurana, Pavel Tokmakov, Cordelia Schmid, and Deva Ramanan. "TAO: A Large-Scale Benchmark for Tracking Any Object." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020 (cit. on p. 2).

[32] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. "Mot20: A benchmark for multi object tracking in crowded scenes." In: *arXiv preprint arXiv:2003.09003* (2020) (cit. on p. 7).

[33] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. "FlowNet: Learning Optical Flow with Convolutional Networks." In: *The IEEE International Conference on Computer Vision (ICCV)*. 2015 (cit. on pp. 16, 32, 34).

[34] Zhihao Duan, Ozan Tezcan, Hayato Nakamura, Prakash Ishwar, and Janusz Konrad. "Rapid: rotation-aware people detection in overhead fisheye images." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 636–637 (cit. on p. 7).

[35] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. "LaSOT: A High-Quality Benchmark for Large-Scale Single Object Tracking." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 (cit. on pp. 4, 54–56, 58, 63–67, 70, 71, 73–78, 80).

[36] Heng Fan and Haibin Ling. "CRACT: Cascaded Regression-Align-Classification for Robust Tracking." In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 7013–7020 (cit. on p. 68).

[37] Gunnar Farnebäck. "Two-Frame Motion Estimation Based on Polynomial Expansion." In: *Image Analysis, 13th Scandinavian Conference, SCIA 2003, Halmstad, Sweden, June 29 - July 2, 2003, Proceedings*. Ed. by Josef Bigün and Tomas Gustavsson. Vol. 2749. Lecture Notes in Computer Science. Springer, 2003, pp. 363–370 (cit. on pp. 16, 22).

[38] G. Floros and B. Leibe. "Joint 2D-3D temporally consistent semantic segmentation of street scenes." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012 (cit. on p. 32).

[39] Georgios Floros and Bastian Leibe. "Joint 2D-3D Temporally Consistent Semantic Segmentation of Street Scenes." In: 2012 (cit. on p. 14).

[40] Zhihong Fu, Qingjie Liu, Zehua Fu, and Yunhong Wang. "STMTrack: Template-Free Visual Tracking With Space-Time Memory Networks." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021 (cit. on pp. 66–68, 76).

[41] Raghudeep Gadde, Varun Jampani, and Peter V. Gehler. "Semantic Video CNNs through Representation Warping." In: *The IEEE International Conference on Computer Vision (ICCV)*. 2017 (cit. on pp. 16, 27, 30, 32, 42).

[42] Hamed Kiani Galoogahi, Ashton Fagg, Chen Huang, Deva Ramanan, and Simon Lucey. "Need for Speed: A Benchmark for Higher Frame Rate Object Tracking." In: *The IEEE International Conference on Computer Vision (ICCV)*. 2017 (cit. on pp. 64, 65, 68, 74, 76).

[43] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. "Vision meets Robotics: The KITTI Dataset." In: *The International Journal of Robotics Research (IJRR)* (2013) (cit. on pp. 16, 32).

[44] Qing Guo, Wei Feng, Ce Zhou, Rui Huang, Liang Wan, and Song Wang. "Learning Dynamic Siamese Network for Visual Object Tracking." In: *The IEEE International Conference on Computer Vision (ICCV)*. 2017 (cit. on p. 55).

[45] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. "Towards a better match in siamese network based visual object tracker." In: *Proceedings of the European Conference on Computer Vision Workshops (ECCVW)*. 2018 (cit. on p. 55).

[46] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cit. on pp. 42, 47).

[47] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. "High-Speed Tracking with Kernelized Correlation Filters." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 37.3 (2015), pp. 583–596 (cit. on pp. 55, 61).

[48] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. "Local relation networks for image recognition." In: *The IEEE International Conference on Computer Vision (ICCV)*. 2019 (cit. on p. 33).

[49] P. Hu, Federico Perazzi, Fabian Caba Heilbron, Oliver Wang, Zhe Lin, Kate Saenko, and S. Sclaroff. "Real-Time Semantic Segmentation With Fast Attention." In: *RA-L* 6 (2021), pp. 263–270 (cit. on pp. 30, 33, 42, 45).

[50] Ping Hu, Fabian Caba, Oliver Wang, Zhe Lin, Stan Sclaroff, and Federico Perazzi. "Temporally Distributed Networks for Fast Video Semantic Segmentation." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 (cit. on pp. 30, 33, 42, 45).

[51] Yuan-Ting Hu, Jia-Bin Huang, and Alexander G Schwing. "Videomatch: Matching based video object segmentation." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 54–70 (cit. on p. 57).

[52] Lianghua Huang, Xin Zhao, and Kaiqi Huang. "GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 43.5 (2021), pp. 1562–1577 (cit. on pp. 55, 63, 66, 67, 74).

[53] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on pp. 16, 32).

[54] Lei Ke, Martin Danelljan, Henghui Ding, Yu-Wing Tai, Chi-Keung Tang, and Fisher Yu. "Mask-Free Video Instance Segmentation." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*. 2023 (cit. on p. 6).

[55] Anna Khoreva, Rodrigo Benenson, Eddy Ilg, Thomas Brox, and Bernt Schiele. "Lucid data dreaming for object tracking." In: *The DAVIS Challenge on Video Object Segmentation*. 2017 (cit. on p. 57).

[56] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2014 (cit. on p. 63).

[57] Alexander Kirillov, Ross B. Girshick, Kaiming He, and Piotr Dollár. "Panoptic Feature Pyramid Networks." In: *CoRR* abs/1901.02446 (2019) (cit. on p. 14).

[58] Matej Kristan, Aleš Leonardis, Jiří Matas, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Hyung Jin Chang, Martin Danelljan, Luka Čehovin Zajc, Alan Lukežič, et al. "The Tenth Visual Object Tracking VOT2022 Challenge Results." In: *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VIII*. Springer. 2023, pp. 431–460 (cit. on p. vii).

[59] Matej Kristan, Aleš Leonardis, Jiří Matas, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Martin Danelljan, Luka Čehovin Zajc, Alan Lukežič, Ondrej Drbohlav, Linbo He, Yushan Zhang, Song Yan, Jinyu Yang, Gustavo Fernández, and et al. "The Eighth Visual Object Tracking VOT2020 Challenge Results." In: *Proceedings of the European Conference on Computer Vision Workshops (ECCVW)*. 2020 (cit. on p. 68).

[60] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105 (cit. on p. 13).

[61] Till Kroeger, Radu Timofte, Dengxin Dai, and Luc Van Gool. "Fast Optical Flow using Dense Inverse Search." In: *The European Conference on Computer Vision (ECCV)*. 2016 (cit. on pp. 12, 16, 17, 21, 22, 32, 81).

[62] A. Kundu, V. Vineet, and V. Koltun. “Feature Space Optimization for Semantic Video Segmentation.” In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cit. on p. 32).

[63] Abhijit Kundu, Y. Li, F. Dellaert, F. Li, and James M. Rehg. “Joint Semantic Segmentation and 3D Reconstruction from Monocular Video.” In: *The European Conference on Computer Vision (ECCV)*. 2014 (cit. on p. 32).

[64] Abhijit Kundu, Yin Li, Frank Dellaert, Fuxin Li, and James M. Rehg. “Joint Semantic Segmentation and 3D Reconstruction from Monocular Video.” In: *Computer Vision – ECCV 2014*. Ed. by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars. Cham: Springer International Publishing, 2014, pp. 703–718 (cit. on p. 14).

[65] Zihang Lai and Weidi Xie. “Self-supervised learning for video correspondence flow.” In: *arXiv preprint arXiv:1905.00875* (2019) (cit. on p. 2).

[66] J. Lezama, K. Alahari, J. Sivic, and I. Laptev. “Track to the Future: Spatio-temporal Video Segmentation with Long-range Motion Cues.” In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2011 (cit. on pp. 14, 32).

[67] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. “SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 (cit. on pp. 55, 57, 66).

[68] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. “High Performance Visual Tracking With Siamese Region Proposal Network.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on p. 55).

[69] Jiangyun Li, Yikai Zhao, Jun Fu, Jiajia Wu, and Jing Liu. “Attention-guided network for semantic video segmentation.” In: *IEEE Access* 7 (2019), pp. 140680–140689 (cit. on pp. 30, 33).

[70] Xiaoxiao Li and Chen Change Loy. “Video object segmentation with joint re-identification and attention-aware mask propagation.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 90–105 (cit. on p. 57).

[71] Xiaoxiao Li, Ziwei Liu, Ping Luo, Chen Change Loy, and Xiaoou Tang. "Not All Pixels Are Equal: Difficulty-Aware Semantic Segmentation via Deep Layer Cascade." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on p. 14).

[72] Yule Li, Jianping Shi, and Dahua Lin. "Low-Latency Video Semantic Segmentation." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on pp. 14, 27, 30, 32, 42).

[73] Yifan Liu, Chunhua Shen, Changqian Yu, and Jingdong Wang. "Efficient Semantic Video Segmentation with Per-Frame Inference." In: *The European Conference on Computer Vision (ECCV)*. Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. 2020 (cit. on pp. 30, 33).

[74] Ziwei Liu, Xiaoxiao Li, Ping Luo, Chen Change Loy, and Xiaoou Tang. "Semantic Image Segmentation via Deep Parsing Network." In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 1377–1385 (cit. on p. 13).

[75] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. 2015, pp. 3431–3440 (cit. on pp. 13, 14).

[76] Bruce D. Lucas and Takeo Kanade. "Optical Navigation by the Method of Differences." In: *Proceedings of the 9th International Joint Conference on Artificial Intelligence. Los Angeles, CA, USA, August 1985*. Ed. by Aravind K. Joshi. Morgan Kaufmann, 1985, pp. 981–984 (cit. on p. 16).

[77] Alan Lukezic, Jiri Matas, and Matej Kristan. "D3S - A Discriminative Single Shot Segmentation Tracker." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 (cit. on pp. 57, 58, 66–69, 82).

[78] Alan Lukezic, Tomás Vojír, Luka Cehovin Zajc, Jiri Matas, and Matej Kristan. "Discriminative Correlation Filter Tracker with Channel and Spatial Reliability." In: *International Journal of Computer Vision (IJCV)* 126.7 (2018), pp. 671–688 (cit. on p. 55).

[79] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. "Multiple object tracking: A literature review." In: *Artificial intelligence* 293 (2021), p. 103448 (cit. on p. 7).

[80] Behrooz Mahasseni, Sinisa Todorovic, and Alan Fern. "Budget-Aware Deep Semantic Video Segmentation." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017) (cit. on pp. 14, 32).

[81] Kevis-Kokitsi Maninis, Sergi Caelles, Yuhua Chen, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. "Video object segmentation without temporal information." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 41.6 (2018), pp. 1515–1530 (cit. on p. 57).

[82] Christoph Mayer, Martin Danelljan, Goutam Bhat, Matthieu Paul, Danda Pani Paudel, Fisher Yu, and Luc Van Gool. "Transforming Model Prediction for Tracking." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 8731–8740 (cit. on pp. vii, 57, 66–68, 75, 76, 82).

[83] Christoph Mayer, Martin Danelljan, Danda Pani Paudel, and Luc Van Gool. "Learning Target Candidate Association To Keep Track of What Not To Track." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 13444–13454 (cit. on pp. 66–68, 75, 76, 78).

[84] Christoph Mayer, Matthieu Paul, and Radu Timofte. "Adversarial feature distribution alignment for semi-supervised learning." In: *Computer Vision and Image Understanding* 202 (2021), p. 103109 (cit. on p. vii).

[85] Sachin Mehta, Mohammad Rastegari, Anat Caspi, Linda Shapiro, and Hannaneh Hajishirzi. "ESPNet: Efficient Spatial Pyramid of Dilated Convolutions for Semantic Segmentation." In: *The European Conference on Computer Vision (ECCV)*. 2018 (cit. on p. 14).

[86] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. "MOT16: A benchmark for multi-object tracking." In: *arXiv preprint arXiv:1603.00831* (2016) (cit. on p. 7).

[87] Matthias Mueller, Neil Smith, and Bernard Ghanem. "A Benchmark and Simulator for UAV Tracking." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2016 (cit. on pp. 54, 58, 64, 65, 67, 68, 74, 76).

[88] Matthias Müller, Adel Bibi, Silvio Giancola, Salman Al-Subaihi, and Bernard Ghanem. "TrackingNet: A Large-Scale Dataset and Benchmark for Object Tracking in the Wild." In: *The European Conference on Computer Vision (ECCV)*. 2018 (cit. on pp. 55, 67, 74).

[89] R. D. Nijs, Sebastian Ramos, Gemma Roig, X. Boix, L. V. Gool, and K. Kühnlenz. "On-line semantic perception using uncertainty." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2012) (cit. on p. 32).

[90] David Nilsson and Cristian Sminchisescu. "Semantic Video Segmentation by Gated Recurrent Flow Propagation." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on pp. 14, 16, 27, 30, 32, 33, 42).

[91] Peter Ochs, Jitendra Malik, and Thomas Brox. "Segmentation of Moving Objects by Long Term Video Analysis." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 36.6 (2014), pp. 1187–1200 (cit. on pp. 14, 32).

[92] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. "Video Object Segmentation Using Space-Time Memory Networks." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019 (cit. on pp. 30, 33, 34, 54, 57, 69, 74).

[93] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. "ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation." In: *CoRR* abs/1606.02147 (2016) (cit. on pp. 14, 27).

[94] Matthieu Paul, Martin Danelljan, Luc Van Gool, and Radu Timofte. "Local Memory Attention for Fast Video Semantic Segmentation." In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 1102–1109 (cit. on pp. vii, 79, 80).

[95] Matthieu Paul, Martin Danelljan, Christoph Mayer, and Luc Van Gool. "Robust visual tracking by segmentation." In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*. Springer. 2022, pp. 571–588 (cit. on pp. vii, 80, 82).

[96] Matthieu Paul, Christoph Mayer, Luc Van Gool, and Radu Timofte. "Efficient video semantic segmentation with labels propagation and refinement." In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2020, pp. 2873–2882 (cit. on pp. vii, 30, 32, 42, 45, 79).

[97] Duo Peng, Yinjie Lei, Munawar Hayat, Yulan Guo, and Wen Li. "Semantic-Aware Domain Generalized Segmentation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 2594–2605 (cit. on p. 2).

[98] 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: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cit. on pp. 16, 63).

[99] Federico Perazzi, Anna Khoreva, Rodrigo Benenson, Bernt Schiele, and Alexander Sorkine-Hornung. "Learning video object segmentation from static images." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2663–2672 (cit. on p. 57).

[100] Trung T. Pham, Thanh-Toan Do, Niko Sünderhauf, and Ian Reid. "SceneCut: Joint Geometric and Object Segmentation for Indoor Scenes." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 3213–3220 (cit. on p. 5).

[101] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. "The 2017 DAVIS Challenge on Video Object Segmentation." In: *arXiv:1704.00675* (2017) (cit. on pp. 2, 54, 57, 63, 69, 74).

[102] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. "Playing for Data: Ground Truth from Computer Games." In: *The European Conference on Computer Vision (ECCV)*. 2016 (cit. on pp. 2, 16, 32).

[103] Eduardo Romera, J. M. Álvarez, L. M. Bergasa, and Roberto Arroyo. "ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation." In: *T-ITS* 19 (2018), pp. 263–272 (cit. on pp. 4, 14, 27, 30, 40–42, 44–47).

[104] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 3234–3243 (cit. on p. 2).

[105] Sunando Sengupta, Eric Greveson, Ali Shahrokni, and Philip H. S. Torr. "Urban 3D semantic modelling using stereo vision." In: *2013 IEEE International Conference on Robotics and Automation (ICRA)* (2013), pp. 580–585 (cit. on p. 14).

[106] Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. "Clockwork Convnets for Video Semantic Segmentation." In: *Proceedings of the European Conference on Computer Vision Workshops (ECCVW)*. Ed. by Gang Hua and Hervé Jégou. 2016 (cit. on pp. 14, 27, 32).

[107] Jianbo Shi and Jitendra Malik. "Normalized Cuts and Image Segmentation." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 22.8 (2000), pp. 888–905 (cit. on p. 14).

[108] Kevin Smith, Daniel Gatica-Perez, J Odobez, and Sileye Ba. "Evaluating multi-object tracking." In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)-workshops*. IEEE. 2005, pp. 36–36 (cit. on p. 7).

[109] Jeany Son, Ilchae Jung, Kayoung Park, and Bohyung Han. "Tracking-by-Segmentation With Online Gradient Boosting Decision Tree." In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015 (cit. on p. 57).

[110] Niko Sünderhauf, Trung T. Pham, Yasir Latif, Michael Milford, and Ian Reid. "Meaningful maps with object-oriented semantic mapping." In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 5079–5085 (cit. on p. 5).

[111] Ran Tao, Efstratios Gavves, and Arnold W. M. Smeulders. "Siamese Instance Search for Tracking." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cit. on p. 55).

[112] P. Tokmakov, K. Alahari, and C. Schmid. "Learning Motion Patterns in Videos." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on p. 16).

[113] Pavel Tokmakov, Karteek Alahari, and Cordelia Schmid. "Learning Video Object Segmentation with Visual Memory." In: *The IEEE International Conference on Computer Vision (ICCV)*. 2017 (cit. on pp. 14, 30, 32, 33).

[114] Michael Treml, Jose A. Arjona-Medina, Thomas Unterthiner, Rupesh Durgesh, Felix Friedmann, Peter Schuberth, Andreas Mayr, Martin Heusel, Markus Hofmarcher, Michael Widrich, Bernhard Nessler, and Sepp Hochreiter. "Speeding up Semantic Segmentation for Autonomous Driving." In: *NIPS workshop*. 2016 (cit. on pp. 14, 27).

[115] Subarna Tripathi, Serge J. Belongie, Y. Hwang, and T. Nguyen. "Semantic video segmentation: Exploring inference efficiency." In: *ISOCC* (2015), pp. 157–158 (cit. on p. 32).

[116] Jack Valmadre, Luca Bertinetto, Joao Henriques, Andrea Vedaldi, and Philip H. S. Torr. "End-To-End Representation Learning for Correlation Filter Based Tracking." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on p. 55).

[117] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is All you Need." In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 5998–6008 (cit. on pp. 30, 33, 34).

[118] Paul Voigtlaender, Yuning Chai, Florian Schroff, Hartwig Adam, Bastian Leibe, and Liang-Chieh Chen. "Feelvos: Fast end-to-end embedding learning for video object segmentation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9481–9490 (cit. on p. 57).

[119] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. "Mots: Multi-object tracking and segmentation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 7942–7951 (cit. on p. 7).

[120] Paul Voigtlaender and Bastian Leibe. "Online adaptation of convolutional neural networks for video object segmentation." In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2017 (cit. on p. 57).

[121] Paul Voigtlaender, Jonathon Luiten, Philip H.S. Torr, and Bastian Leibe. "Siam R-CNN: Visual Tracking by Re-Detection." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 (cit. on pp. 54, 57, 66–69, 76).

[122] Guangting Wang, Chong Luo, Xiaoyan Sun, Zhiwei Xiong, and Wenjun Zeng. "Tracking by Instance Detection: A Meta-Learning Approach." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 (cit. on p. 55).

[123] Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. "Transformer Meets Tracker: Exploiting Temporal Context for Robust Visual Tracking." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021 (cit. on pp. 57, 66–68, 76).

[124] Qiang Wang, Zhu Teng, Junliang Xing, Jin Gao, Weiming Hu, and Stephen J. Maybank. "Learning Attentions: Residual Attentional Siamese Network for High Performance Online Visual Tracking." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on p. 55).

[125] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. "Fast online object tracking and segmentation: A unifying approach." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 (cit. on pp. 57, 69, 82).

[126] Wenguan Wang, Hongmei Song, Shuyang Zhao, Jianbing Shen, Sanyuan Zhao, Steven CH Hoi, and Haibin Ling. "Learning unsupervised video object segmentation through visual attention." In: *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 3064–3074 (cit. on p. 2).

[127] Jiannan Wu, Yi Jiang, Peize Sun, Zehuan Yuan, and Ping Luo. "Language As Queries for Referring Video Object Segmentation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 4974–4984 (cit. on p. 6).

[128] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. "Wider or Deeper: Revisiting the ResNet Model for Visual Recognition." In: *CoRR* abs/1611.10080 (2016) (cit. on p. 13).

[129] Seoung Wug Oh, Joon-Young Lee, Kalyan Sunkavalli, and Seon Joo Kim. "Fast video object segmentation by reference-guided mask propagation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7376–7385 (cit. on p. 57).

[130] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. *YouTube-VOS: A Large-Scale Video Object Segmentation Benchmark*. 2018 (cit. on pp. 2, 57, 63, 68, 69, 73, 74).

[131] Yu-Syuan Xu, Tsu-Jui Fu, Hsuan-Kung Yang, and Chun-Yi Lee. "Dynamic Video Segmentation Network." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on pp. 14, 27, 30, 32, 42, 45, 46).

[132] B. Yan, Dong Wang, Huchuan Lu, and Xiaoyun Yang. "Alpha-Refine: Boosting Tracking Performance by Precise Bounding Box Estimation." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021 (cit. on pp. 54, 57, 66–68, 76).

[133] Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. "Learning Spatio-Temporal Transformer for Visual Tracking." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 10448–10457 (cit. on pp. 54, 56, 57, 66–68, 76, 78, 82).

[134] Xue Yang, Xiaojiang Yang, Jirui Yang, Qi Ming, Wentao Wang, Qi Tian, and Junchi Yan. "Learning high-precision bounding box for rotated object detection via kullback-leibler divergence." In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 18381–18394 (cit. on p. 7).

[135] Bin Yu, Ming Tang, Linyu Zheng, Guibo Zhu, Jinqiao Wang, Hao Feng, Xuetao Feng, and Hanqing Lu. "High-Performance Discriminative Tracking With Transformers." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 9856–9865 (cit. on pp. 57, 67).

[136] Chao Yu, Zuxin Liu, Xin-Jun Liu, Fugui Xie, Yi Yang, Qi Wei, and Qiao Fei. "DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments." In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1168–1174 (cit. on p. 5).

[137] Yuechen Yu, Yilei Xiong, Weilin Huang, and Matthew R. Scott. "Deformable Siamese Attention Networks for Visual Object Tracking." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 (cit. on p. 68).

[138] Sukmin Yun, Hankook Lee, Jaehyung Kim, and Jinwoo Shin. "Patch-Level Representation Learning for Self-Supervised Vision Transformers." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 8354–8363 (cit. on pp. 2, 6).

[139] Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. "Ocean: Object-Aware Anchor-Free Tracking." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020 (cit. on p. 66).

[140] Zikai Zhang, Bineng Zhong, Shengping Zhang, Zhenjun Tang, Xin Liu, and Zhaoxiang Zhang. "Distractor-Aware Fast Tracking via Dynamic Convolutions and MOT Philosophy." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021 (cit. on p. 66).

[141] Bin Zhao, Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. "Generating Masks From Boxes by Mining Spatio-Temporal Consistencies in Videos." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 13556–13566 (cit. on pp. 6, 57, 58, 63, 66–69, 74).

[142] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. "ICNet for Real-Time Semantic Segmentation on High-Resolution Images." In: *The European Conference on Computer Vision (ECCV)*. 2018 (cit. on pp. 12, 14, 17, 20, 21, 25, 27, 42, 81).

[143] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. "Pyramid Scene Parsing Network." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on pp. 4, 14, 27, 30, 42, 45–48).

[144] Linyu Zheng, Ming Tang, Yingying Chen, Jinqiao Wang, and Hanqing Lu. "Learning Feature Embeddings for Discriminant Model Based Tracking." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020 (cit. on p. 55).

[145] X. Zhu, Y. Xiong, Jifeng Dai, L. Yuan, and Y. Wei. "Deep Feature Flow for Video Recognition." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017) (cit. on pp. 16, 27, 30, 32, 42, 45).

[146] Zheng Zhu, Qiang Wang, Li Bo, Wei Wu, Junjie Yan, and Weiming Hu. "Distractor-aware Siamese Networks for Visual Object Tracking." In: *The European Conference on Computer Vision (ECCV)*. 2018 (cit. on p. 55).

[147] Adrian Ziegler and Yuki M. Asano. "Self-Supervised Learning of Object Parts for Semantic Segmentation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 14502–14511 (cit. on pp. 2, 6).