Efficiently Exploiting Space-Time Consensus for Object Segmentation and Tracking in Video

Elena Burceanu

coordinated by
Prof. Dr. Marius Leordeanu and Prof. Dr. Gheorghe Stefanescu

The Faculty of Mathematics and Computer Science,
University of Bucharest

ABSTRACT

"A process cannot be understood by stopping it. Understanding must move with the flow of the process, must join it and flow with it."

Frank Herbert, Dune

ACKNOWLEDGEMENTS

I want to thank first to my coordinators: Marius Leordeanu, for continually guiding me through the PhD experience, helping me improve my analytical skills, and teaching me how to approach a problem step by step, and Gheorghe Stefanescu for advises and openness in front of a new field of research.

I thank Bitdefender for fully supporting our theoretical research team over the years in the road towards putting Romania on the map of AI and Crypto top research, and in particular to our CTO, Bogdan Dumitru, without whom this would have been only a dream.

Many thanks to each member of the Bitdefender team for their sustained hard work, passion, and dedication, and for the great moments we shared over the years as friends and co-workers: reading groups, meetups, courses, summer schools, projects, papers, patents, and not least for teambuilding trips, going out and hikes. I want to thank the computer vision folks (Ema, Iulia, and Andrei) for in-depth and valuable debates and collaborations, the NLP guys (Florin and Mano) for motivation, great brainstorming, and successful projects, the RL guy (Florin) for all his support over the years and stubbornness in achieving valuable research. I thank the Crypto folks (Miruna, Radu, and Madalina) for numerous discussions on contributing by combining AI with Crypto. I also thank the others that were part of our team at some point and moved our dream further (Tudor, Stefan, Alexandra, Andu).

I thank Viorica, Razvan, and Doina for encouraging us as a team and their unconditioned support, availability, and implication in the Romanian AI landscape. I also thank Traian and Bogdan for facilitating our interactions with the local universities.

BIO AND PUBLICATIONS

Elena Burceanu (ilarele.github.io) has a strong background in Mathematics and Physics with awards at national and international contests. She has finished her BSc in Computer Science in 2011 and the MSc in Distributed Systems in 2013 at University Politehnica Bucharest, Romania. Elena is interested in understanding videos in an unsupervised manner, publishing in top AI conferences. Elena is the Scientific Coordinator for Theoretical Research in AI team at Bitdefender, a leading IT company worldwide in the security field.

1. A. Manolache, F. Brad, E. Burceanu, A. Barbalau, R. Ionescu, M. Popescu, Transferring BERT-like Transformers' Knowledge for Authorship Verification, **arXiv 2021**

2. E. Haller, E. Burceanu, M. Leordeanu, Self-Supervised Learning in Multi-Task Graphs through Iterative Consensus Shift, **BMVC 2021**

3. A. Manolache, F. Brad, E. Burceanu, DATE: Detecting Anomalies in Text via Self-Supervision of Transformers, **NAACL 2021** and **ICML 2021 - UDL Workshop**

4. E. Burceanu, SFTrack++: A Fast Learnable Spectral Segmentation Approach for Space-Time Consistent Tracking, **NeurIPS 2020 - Pre-registration Workshop** and **ICCV 2021 - SRVU Workshop**

5. E. Burceanu, M. Leordeanu, A 3D Convolutional Approach to Spectral Object Segmentation in Space and Time, **IJCAI 2020**

6. E. Burceanu, M. Leordeanu, Learning a Robust Society of Tracking Parts using Co-occurrence Constraints, **ECCV 2018 - Visual Object Tracking Workshop**

7. E. Burceanu, M. Leordeanu, Learning a Robust Society of Tracking Parts, **RAAI2018 workshop** and **arXiv 2017**

PATENTS (BOTH IN US AND UE)

Approved 2018 E. Burceanu, F. Brad, T. Rebedea, Systems And Methods For Translating Natural Language Sentences Into Database Queries

Submitted 2020 B.Cebere, M.Rosca, R.Titiu, E. Haller, E. Burceanu, M. Bolboceanu, Privacy-Preserving Domain Name Service (DNS)

Submitted 2021 A. Manolache, F. Brad, E. Burceanu, A. Novac, Anomaly Detection Systems And Methods

Submitted 2021 E. Haller, E. Burceanu, M. Bolboceanu, B.Cebere, M.Rosca, R.Titiu, Privacy-Preserving Image Distribution

Submitted 2021 E. Haller, E. Burceanu, M. Bolboceanu, B.Cebere, M.Rosca, R.Titiu, Image Distribution Using Composite Re-Encrypted Images

In prep 2021 E. Haller, E. Burceanu, M. Leordeanu, R.Prejbeanu, D.Cernat, Unsupervised Labeling and Domain Adaptation using Experts Consensus in Security

CONTENTS

ACRONYMS

CShift Consensus Shift.

SFSeg Spectral Filtering Segmentation.
SFSeg++ Spectral Filtering Segmentation with multi-channel learning.
SFTrack++ Spectral Filtering Tracking with multi-channel learning.
STP Society of Tracking Parts.

1 INTRODUCTION

The eyes are our most used sensor in the body, backed up by the most complex processing architecture in the brain, consisting of hundreds of millions of neurons. Compared with the tactile sense, auditory, taste, or smell, seeing is by far our most important help in interacting with the world [53], with almost 30% of the brain being involved in visual-related tasks. But this seems natural when up to 90% of the information processed in the brain is visual.

Our visual system design principles are shared among all primates [66], all having increased dependence on the vision rather than on smell, which is the most prevailing sense in mammals [64]. Based on those current findings in neuroscience, we can argue that the vision tasks, right from the nature of its input sensor, deal with the largest and the richest flow of information out there.

The brain is the organ responsible for learning, and vision uses a considerable share of its resources. Since we as humans evolve in understanding and interacting with the environment, relying primarily on our two visual sensors, we can infer that, if we want to build in computers capabilities similar to ours, we should focus on vision first.

A classic view in the cognitive psychology [51] emphasizes that the direct perception has a vibrant stream of information, already containing most of the complex information you need. This view is also being shared in some of the top computer vision labs [159]. Based on this hypothesis, as opposed to having an embodied structural representation bias, we can speculate that vision has the greatest potential to further make use of its input, without explicit supervision, either by using supervision intrinsic to data or just by correlating it with other sensors.

A natural approach from the machine learning community was to focus more on the most powerful predicting signal: vision, the richest source of information. Over time, computer vision laboratories exploited it and continuously challenged the status quo through novel problem formulation and innovative algorithms, following the latest hardware advances. But in the latest years, we see great efforts in switching from vision-only approaches to other modalities [100, 159], synchronizing multiple domains or tasks [190, 92], trying to simu-

late the senses consensus towards better generalization and lower amounts of labeled data.

1.1 MOTIVATION: TRACKING AND SEGMENTATION IN VISION

Object tracking and segmentation in video was a good subject to choose for my PhD subject. What intrigued me about it from the beginning and got my interest was that those tasks seemed really simple (or simple to do), but the best methods were fragile compared with the expectations: "How could the algorithm be so off the target?", "This is way too simple. Why didn't it work?". So it was clear that I was missing something when I evaluated the difficulty of the process for a computer. I was also attracted by the unsupervised setup (or at least the self-supervised one), where you don't rely so much on a large number of new labels, but you can find an intrinsic way (through the algorithm or the data) to guide the learning.

So I choose tracking as it represents one of the most straightforward proofs that you are getting some basic, physic interactions rules, that you are following something, and seems to be the first step in making sense out of a video. You do not need to put a label or to understand the problem explicitly. It is something we as humans (but also animals) do involuntarily all the time and without this primary task "solved" by the brain, our everyday life activities would look completely different (maybe also the dreams). For instance, tracking helps you isolate the pixels and the parts of an unseen object, allowing you to "reserve" a new class for it or further letting you put on a known label.

Tracking is a fundamental task that is learned by the brain. Babies are not born knowing how to track, but they master the skill pretty well after the first three months of life and can do it without specific supervision. They somehow learn to interpret their retina signals towards tracking, just by being exposed to the environment with all their senses.

seeing video "samples".

So it is not something we are born with but is something we are made for to acquire very fast, and we might have developed over time some kind of structural bias in our brain that facilitates this.

Of course, the skill is not perfect even for adults, and optical illusions best exploit this. Usually, the brain auto-completes the missing information instead of flagging it as missing or tries to adjust it to fit the expectations [45]. Thus,

the brain optimizes towards the sentiment of control and calm, allowing us to have a "fake" sensation of continuity in the observed world.

Nevertheless, while tracking is an effortless task for us humans, there is still a substantial qualitative gap between human and computer performance. This is true except for several standard classes (like pedestrians or vehicles), where the algorithm's effectiveness is due to focusing on the object class appearance, trained on a large amount of labeled data, rather than on the tracking task.

fundamental research

Tracking task is a basic block when building computer vision applications, and I resonate with the idea that for being able to create complex things, the first necessary thing to do is to really understand and try to improve some of its fundamental sub-tasks. The fundamental research is crucial because it puts structure and reasoning ahead of empirical validations. Otherwise, we might spin in a circle and miss the critical observations needed to evolve our current understanding and solutions. Of course, we should stay grounded in the real world (from which the real supervision comes) and validate the theoretical results in applied solutions, linking the concepts and making them valuable and tangible in the real world. So, tracking is not the cherry on the cake, but rather its foundation, the cake's genoise, the building block essential for having a cake, and later on, a cherry on top of it.

1.2 CONTRIBUTION AND IMPACT

video object understanding

The main focus of my thesis is to understand the objects in video, that continuously evolve in space and time by analyzing them under tracking and segmentation tasks context. We center our efforts towards exploiting the inherent spatio-temporal consistency of the object, observed under multiple levels by different parts of the system. Trusting their consensus allows us to rely only on a low quantity of supervision in the process.

CONTEXT IN THE LITERATURE We focus on finding the target in the video by extracting more helpful information based on space and time consistencies of different actors at multiple levels. This contrasts with other works in which the time axis is neglected or used only for some basic smoothing. Space and time consistent and natural integration is in fact one of the main points of interest in Computer Vision nowadays, not only in tracking. Therefore, our contributions focus on redesigning models to work with videos in a compact way, rather than frame by frame, managing to take full advantage of the temporal axis, as

efficiently as possible. While focusing on extracting valuable information by a consistent and efficient integration of the temporal axis with the spatial one, we reach other several aspects, which we prove further in our work that are key components in achieving this ambitious purpose:

A. SPACE–TIME CONSISTENCY We integrate in our work the space and time dimensions thoughtfully. In STP (Chapter 3) the roles in society are slowly but continuously updating over time, changing the part's priority in voting. SFSeg++ (Chapter 4) masters this spatio-temporal integration by seeing the video as a densely connected space-time volume in which we cluster the pixels uniformly over the three dimensions. SFTrack++ (Chapter 5) is similar from this point of view since we used the previous approach and further exploited it for the tracking task. CShift (Chapter 6) works only at the spatial level, but we will address the space-time multi-task graph in future work.

B. THE POWER OF THE CONSENSUS We take the most out of the ensemble and exploit it by using a wide variety of combinations at different levels that enabled us with increasingly powerful ensembles. We start in STP by combining many weak parts computed over deep features with a strong end-to-end pathway. Next, we learn to combine state-of-the-art top methods, directly in SFSeg++ or in feature space in SFTrack++, based on natural connections becoming visible through clustering. In CShift, we take a step back and apply the ensemble at multiple levels. We combine towards consensus:

1. various architectures and complexity (from simple and similar UNet edges in the graph to complex experts)
2. multiple training datasets (coming with a different data distribution for each expert)
3. numerous domains, enriching the semantic diversity (from simple edges to complex segmentation)
4. multiple-paths reaching a destination, through varied intermediate representations

C. EXPLOITING MULTIPLE INTERMEDIATE REPRESENTATIONS We get the first informal hint on the value of having distinct representations in an ensemble in our Society of Tracking Parts (STP), where both intermediate features, but also the basic RGB input worked together. Then, SFTrack++ confirms that

for tracking, having a semantic segmentation level in the middle of the pipeline towards the final prediction helps a lot. So we pose the following question in CShift and validate the power of using multiple (13) representations from a variety of tasks (edge detectors, semantic segmentation, depth, etc.) when looking for consensus. This iteratively improves the performance and converges to a significantly better solution without additional supervision. We prove it only in the spatial domain and, as future work, we plan to expand the CShift multi-task graph over temporal dimension, allowing us to also have a tracking task.

D. A LIMITED QUANTITY OF SUPERVISION In our solutions, we chase a lower amount of supervision. We start with an unsupervised solution for tracking in STP (using only VGG [144] pretrained on ImageNet [34]). Next, in SFSeg single channel solution, we rely on natural properties and connections revealed by spectral clustering. We add a layer on top of the unsupervised spectral clustering solution (in SFSeg++ and SFTrack++) by combining it with a learning component over multiple channels to strengthen it using available labeled data. Later on, in CShift, we approach an unsupervised solution from the task point of view. We take advantage of existing knowledge and data from expert models, balancing both the need for supervision and its high cost. We update the initial pseudo-labels coming from experts using the consensus signal reached by the before mentioned powerful selection-based ensemble.

E. EXPERTS: MAKING USE OF EXISTING MODELS I think that using already existing knowledge, consolidated over many research years and multiple datasets is currently underestimated. So we gradually exploit this source of labeling and information over the thesis, starting from pretrained features in STP, simple or learned ensembles over expert models in SFSeg++ and SFTrack++, to using experts' consensus as supervision signal in CShift.

1.2.1 Detailed contributions in each individual work

Learning a Robust Society of Tracking Parts using Co-occurrence Constraints

In the first paper, we show the importance of having multiple complementary parts in an ensemble. We observe a novel theoretical property that allows us to introduce a computational trick for simultaneously learning many linear filters, with an efficient closed-form formulation. We combine a robust pathway composed of those multiple, but simple, parts learned on top of deep features with a more adaptive one, a deep net trained end-to-end. All those

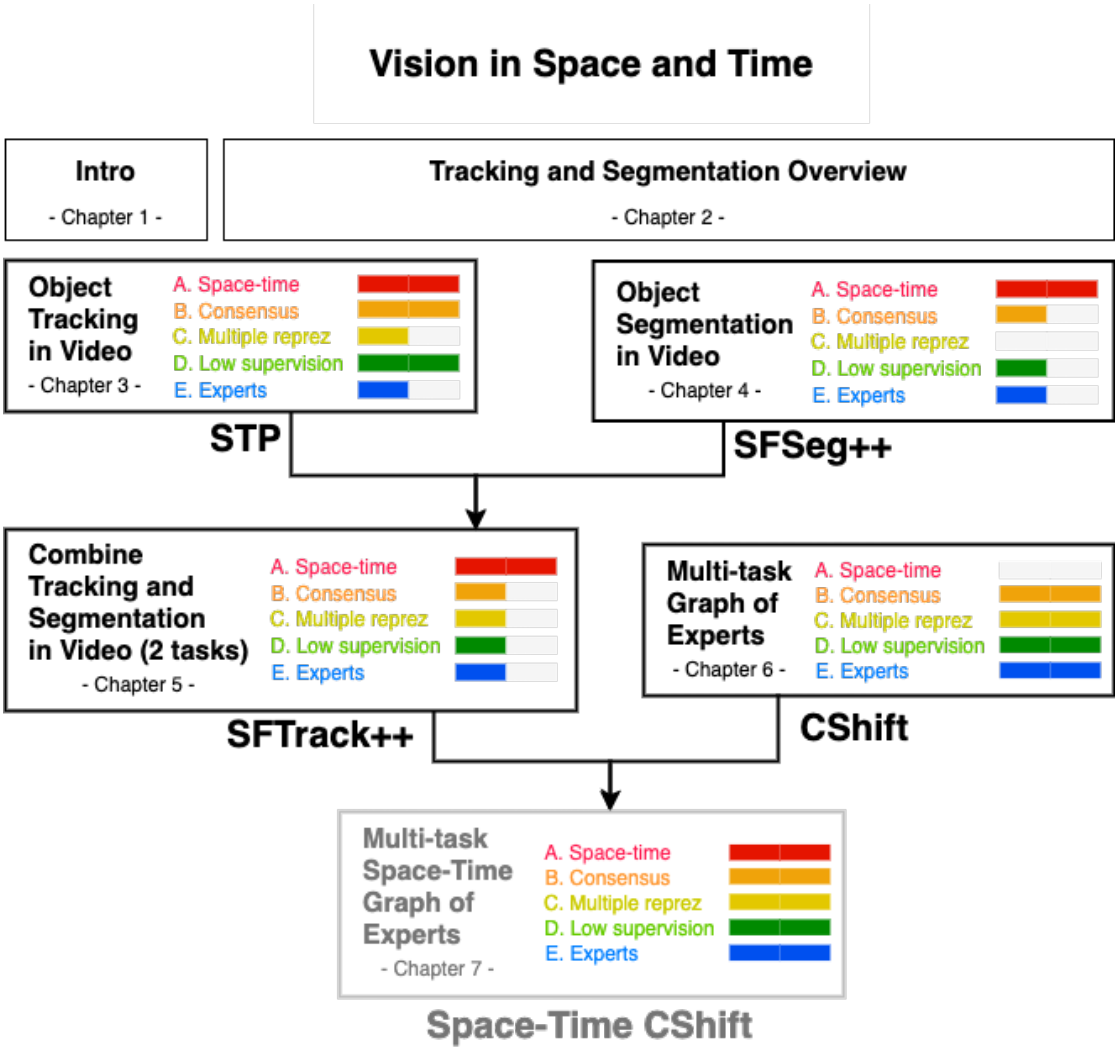

Figure 1: The storyline and contributions map of my thesis.

components, parts of the society, have their role and work together influencing the overall decisions. We show here that the ensemble's diversity and the number of parts are critical components in a tracking system. During this work, we identify the limitation caused by the rigid labeling in tracking. These bounding boxes introduce many noisy signals during each update and voting step of our unsupervised solution.

So we investigate next a more fine-grained task, the object segmentation. Having a segmentation mask for pointing the target object in the first frame is a clear advantage in front of the bulky bounding box, even though it is harder to label. We propose a solution for refining an input segmentation in an un-

A 3D Convolutional Approach to Spectral Object Segmentation in Space and Time

supervised fashion. We start by formulating the object segmentation problem as a spectral graph clustering in space and time video volume, in a similar way with the normalize cut [141]. Differently, we come with a theoretical contribution, an approximation that allows us to operate both on the temporal dimension and with a large spatial resolution. We started with a fully unsupervised solution, that refines an input segmentation based on both temporal and spatial dimensions, clustering the object target pixels in the video volume, iteratively towards convergence. Next, we introduce a learning module that combines over multiple input sources (*e.g.* different segmentation solutions), highlighted by the spectral clustering module. Finally, we combine the traditional algorithm for power iteration with an end-to-end neural net, resulting in a powerful ensemble.

SFTrack++: A Fast Learnable Spectral Segmentation Approach for Space-Time Consistent Tracking

We prove further that using the segmentation task as an intermediate domain for tracking improves the overall results. We integrate the above-proposed segmentation solution into tracking, showing that both the use of an additional, more fine-grained domain and maintaining an object consistency over space and time dimensions are also key components in tracking. The intuition here is based on the fact that the spectral clustering solution applied directly over the bulky bounding box with a lot of noisy content inside the box, serving as a positive label would not reach a better solution. But a mask closer to the natural object shape (like the segmentation) could take advantage of the clustering, going towards the natural contour of the object in an unsupervised manner. And we prove this intuition empirically.

Self-Supervised Learning in Multi-Task Graphs through Iterative Consensus Shift

In the light of these findings, we start investigating a more general one, extrapolating from one domain to multiple ones, reducing the supervision level and exploring in more depth the diversity at different levels (architecture, training data, representation, and prediction pipeline).

1.3 OPEN PROBLEMS

FUTURE WORK We further plan to extend this multi-domain graph to the temporal domain, where the tracking task will be a node for which we can test the graph's performance. The space-time consistency will be seen here through the "eyes" of a large and powerful system.

ETHICAL IMPACT The tracking task might be controversial from the ethical perspective. In this direction, I worked with my team at Bitdefender on so-

Privacy Preserving Image Distribution

lutions that guarantee privacy right from the ground, at the theoretical level. As a result, we have a patent-pending solution for Privacy-Preserving Image Distribution, a system that can operate directly on encrypted data.

1.4 OUTLINE

CHAPTER 1 In this current chapter, I start by presenting the importance of vision in human intelligence, followed by tracking and segmentation tasks' relevance in the broad computer vision field, my motivation when choosing this as my PhD subject, and the contribution and impact of my work.

CHAPTER 2 I present a brief outline of the tracking task, challenges, basic approaches, and some more innovative ideas in the field, but also the close connection between the tracking and the segmentation task.

CHAPTER 3 STP [14] is our first tracking solution. It tackles the problem of fast adapting to new appearances of the target while avoiding drifting to distractors. We address those challenges by proposing a system composed of two pathways, a conservative and a progressive one, working as a society to reach a consensus. The first is robust, composed of many parts, which are linear classifiers over deep pretrained features. We introduce a novel theoretical property that gives us a closed-form solution and efficiently computes all the classifiers simultaneously. Each linear part has a dynamic role in voting for the object center, updated in time based on co-occurrences with the final per frame system decision. In contrast, the second pathway is more adaptive, keeping track of the recent changes in the object's appearance. From the structural point of view, we use a convolutional neural network which we update based on the two pathways consensus signal. This agreement is also used further, as a supervision signal for training the linear parts of the society. I summarized this approach in Fig. 2. *STP*

CHAPTER 4 This chapter explores a mix of traditional algorithms for spectral clustering [16] integrated in nowadays deep learning, for an intermediate task to tracking, object segmentation. While it is way harder to annotate, we argue that object segmentation has more natural labels (the masks following the object contour) than rigid, error-prone bounding boxes in tracking. This aspect of being closer to the true shape of the object makes the segmentation task *SFSeg*

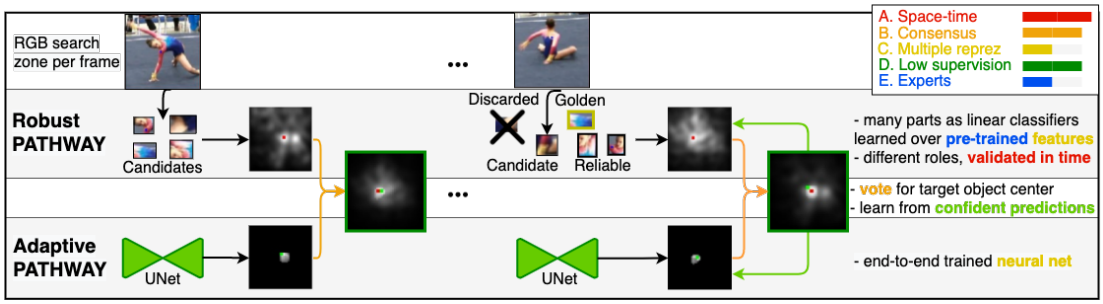

Figure 2: STP: ECCV 2018, Visual Object Tracking Workshop - Chapter 3.

SFSeg++

a proper one for our unsupervised spectral clustering approach in the video volume. We claim that the strongest cluster in the video's spatio-temporal pixel-level volume is the segmentation of the main object. We propose an approximation for computing the leading eigenvector representing this cluster in a very efficient way, orders of magnitude faster than classical approaches, making the dense pixel-level clustering approach for the object segmentation task in video possible. I emphasise key components in Fig. 3.

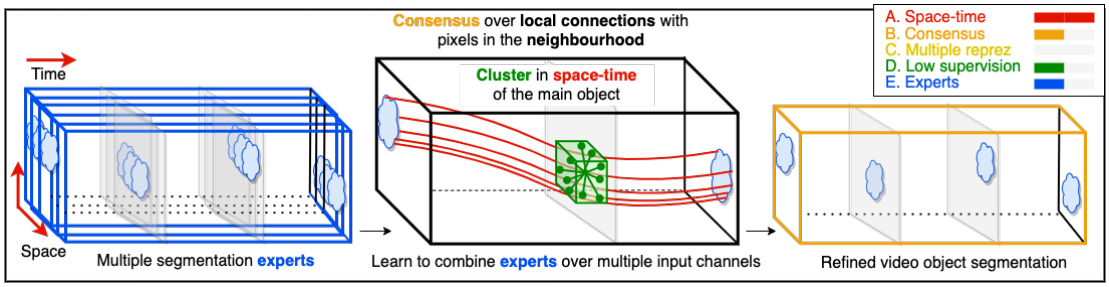

Figure 3: SFSeg++: part at IJCAI 2020, under review 2021 - Chapter 4.

CHAPTER 5 *SFTrack++*

After concluding on the segmentation task solutions described above, we integrate it as a halfway task towards tracking. Our experimental setup backed up our argument that segmentation is a useful intermediate representation for tracking, particularly the SFSeg 3D spectral approach that integrates the temporal dimension harmoniously and naturally with the spatial one. Fig. 4 visually presents the approach.

SFTrack++ work had two stages. I first submitted the proposal for NeurIPS 2020 Pre-registering workshop, where papers are evaluated based on the scientific interest, the soundness of the approach, and the experimental setup plan. This changes the usual focus from following the results against state-of-the-art, forcing you to clear up your direction and all the experiments beforehand.

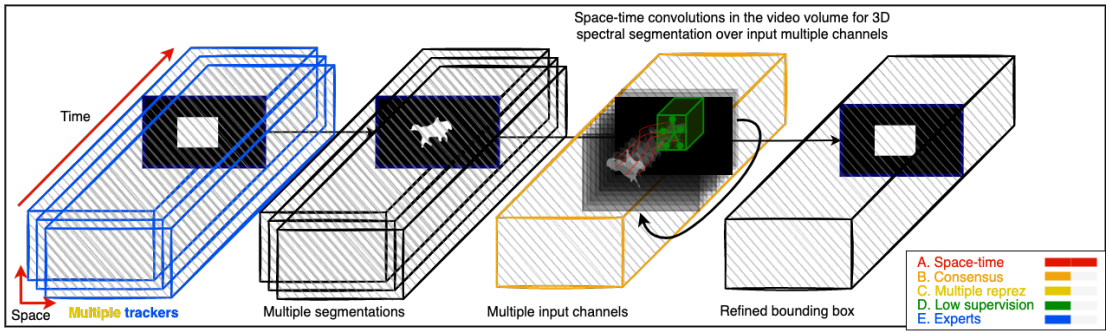

Figure 4: SFTrack++: NeurIPS 2020 Pre-registration Workshop - Chapter 5.

Next, after the proposal was accepted, I went on with the proposed experiments, and the results were positive. Thus, I proved that the approach was valuable and the hypothesis was correct.

CHAPTER 6 We explore in more depth the importance of having other domains as intermediate representations of input. We build a multi-task graph where each node has a different view of the input, somehow rephrasing the initial information from other points of view (domains). For instance, starting from a rgb view, some examples for other nodes in the graph are depth, edges, semantic segmentation, surface normals. One of the main difficulties here is having a labeled dataset with annotations for all domains, for each rgb sample. We address it by using pre-trained expert models for the direct edge (rgb $\rightarrow$ domain). We train the transformation edges using those pseudo labels in the first iteration. We update them afterward by introducing a selection-based consensus algorithm, in a fully unsupervised manner. We focus on the consensus shift over iterations among multiple and very diverse pathways and models. *CShift*

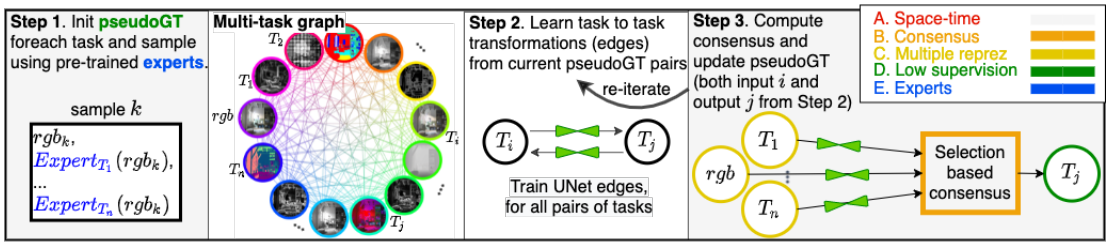

Figure 5: CShift: under review 2021 - Chapter 6

CHAPTER 7 As future work, we extend in Chapter 7 the multi-task graph in the temporal domain, adding tracking and other time-dependent tasks. We plan to explore architectures that integrate the temporal dimension differently, while keeping the amount of computations manageable. The first variation is around seeing an input node as a volume in space-time, versus having a different node for each moment in time. Another direction keeps track of the number of connections in the past and tests whether to keep one only for the currently predicting task or one for each task. To reduce the number of edges we need to train, I will also consider distilling the graph at a certain timestamp in one hidden representation and using it as input for the next frame. Finally, I will integrate the per-frame multi-task graph over time, by combining our two solutions: CShift and multi-channel SFSeg. I show several variants to explore the spatio-temporal graph connections in Fig. 6.

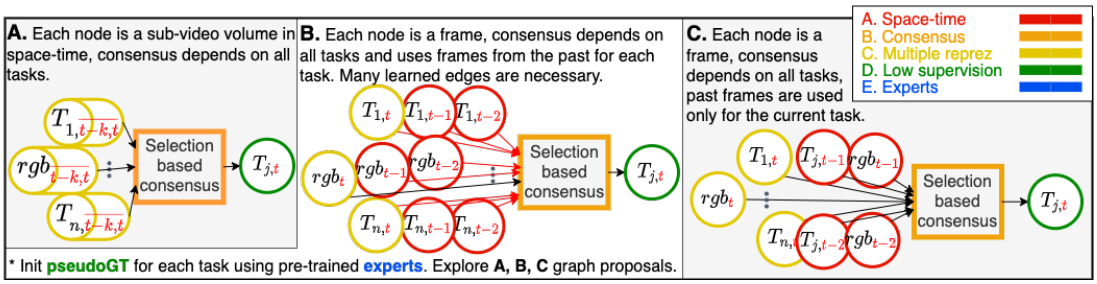

Figure 6: Temporal CShift: Chapter 7.

Ethical tracking and segmentation

I also add at the end an ethical consideration discussion on the implications for tracking and segmentation tasks but also a method to avoid the serious privacy issues that come to light. First, I show the problems that could arise when technical solutions are not used for the greater good. Then, I briefly present a cryptographic layer that can be added on top of tracking and segmentation and can solve most of the concerns. We have a patent-pending application for this formulation.

2 VISUAL TRACKING AND SEGMENTATION OVERVIEW

2.1 TRACKING DESCRIPTION

Computer Vision deals with making computers achieve an in-depth understanding of the world, using the visual field (images or videos). It all starts from the RGB space of visual input, with a multitude of other possibilities to augment via sensors or other derived domains (*e.g.* depth, grayscale, halftone, surface normals). But, except for the spatial dimension of the input, there is also a temporal one, a video containing more information than just the simple sum of the frames, *e.g.* the relation between them. Object tracking in video addresses this exact interconnection between frames, aiming to find not only the link between them but, intrinsically, also the transformations to which a tracked object of interest is passing through by focusing and detecting it in a video at each point in time.

Object tracking is a fundamental task in computer vision, with many applications basing on it and requiring an excellent performance (*e.g.* autonomous driving, visual surveillance, augmented reality, traffic control, gesture recognition).

COMMON PROBLEMS While object tracking is a seamless task for humans, it is not the case for computers. There are a lot of difficult cases, where the algorithms show their limitations. For instance, sudden changes from frame to frame (in the tracked object appearance), background clutter that could make the model jump on distractors or drift over time, or camera motion that unexpectedly changes the focus point are all interesting and currently challenging cases.

2.1.1 Formal definition

Online single object tracking consists in finding the correct trajectory of an object (or a part of it) in a video, meaning its position in space as a function in time. Let $\mathbf{X}$ be a video, a volume of pixels in space-time, with $N = N_f \times H \times W$

pixels, where N_f is the number of frames and $H \times W$ is the frame size. Let $bbox_t = (x_t, y_t, w_t, h_t)$ be the prediction at frame t, where $bbox_0$ is the first bounding box that sets up the tracking region. The tracker does not have access to the entire video, but only to the current frame t. It uses its current state implementation to recall past predictions or other information considered important by the author for the next frame's predictions. Let $state_0$ be the initial tracker state derived from the initialization with $bbox_0$. So the recurrent equation that unfolds the tracking trajectory is the following:

$$bbox_t, state_t = tracker(X_t, state_{t-1}). \tag{1}$$

2.1.2 Evaluation

A tracker is evaluated taking into account multiple metrics, the most used one being the average overlap of the prediction with the ground-truth bounding box (Intersection over Union). Some benchmarks also measure the number of failures. Others even restart the tracker when they consider a fail (when a tracker misses the target for a while, failing to recover). Speed is also an important factor, mainly for real-world application scenarios. Another aspect of tracking is the tracker's internal state, which should have a manageable size even for very long videos.

IoU accuracy

number of failures

speed

ONLINE VS. OFFLINE TRACKING The offline approach involves the tracker having access from the start to the entire video. For example, it can be used to analyze a recorded soccer game or as a pretext task for self-supervised learning of better video embeddings [40]. Nevertheless, this is a less common approach and, if it is not specified otherwise, tracking refers to online tracking as formally defined above in Sec.2.1.1.

OFFLINE LEARNING TRACKERS Most of today's applications rely on a very performant appearance model for the specific category of the object of interest (*e.g.* car, pedestrian, dog, or other animal class, a product class). Usually, the detection problem for several classes of objects is very well solved, based on large, supervised datasets. In this case, the difficulty is differentiating between instances of the same class, becoming a matching problem between instances from consecutive frames (*e.g.* cluttered frames with multiple, close and similar cars or pedestrians). Offline learning for trackers refers to the fact that all learning is done a-priori, offline. And at test time, on a new video, the model is only applied, nothing is updated. The advantage of this approach is that it

focus on specific classes

is very fast and can benefit from a large amount of training data, but only for specific classes, known in advance. The major disadvantage of this approach is that it cannot adapt to a new kind of object.

ONLINE LEARNING TRACKERS In contrast, online training in trackers focuses on the general object tracking problem, where you try to understand the task rather than a particular object class appearance. Here, you need to build an appearance model on the fly during tracking instead of relying on a very precise, a-priory learned model that recognizes the objects within a class. This is the direction on which I center tracking in my thesis.

focus on the tracking task

TRACKING-BY-DETECTION PARADIGM Once with the large advances in object detection, most of the newly proposed tracking pipelines moved to a tracking-by-detection approach. First, an independent detector is used to obtain detections within a search zone (usually near the target position from the last frame). Next, the tracker predicts a bounding box using those candidates, performing associations with the current target trajectory.

2.2 SEGMENTATION IS ESSENTIAL IN TRACKING

The key in tracking is finding the right balance between having an adaptive approach while being robust. If you are too robust, you fail to adapt to new appearances and situations. Otherwise, if you are too adaptive, you risk too much to be fooled by distractors and slowly accumulate errors into the target object representation. The major problem that appears in tracking is drifting and it also derives directly from the task definition. The tracker is initialized with a bounding box that, besides the main object of interest, contains noise like close to object background pixels. Those arguments and empirical observations motivate us to use a more refined representation for the tracked object, the object's segmentation. This is a more natural and precise representation compared with the rigid bounding box. We show empirically that by enforcing a segmentation representation in the hidden layers of the pipeline is of great help in our tracking setup.

drifting

2.2.1 Formal definition

Different from tracking, object segmentation in video is way more meticulous. Instead of describing the object's position in time using two points in each frame, you need to precisely indicate each point situated on the object's contour. When the contour and the volume inside it are a convex set, this represents the full object. But when it does not, It becomes difficult to define the object (*e.g.* when the object has two disjoint parts). So the proper formulation for the object segmentation task is to have a pixel-wise prediction map for each frame, where the value in each pixel represents the probability of that pixel to be part of the target object or not. In a similar manner with tracking, we define the input and the output for a step in the segmentation algorithm that unfolds the entire video object segmentation in Fig. 2. Starting from video $\mathbf{X}$ of $N_f \times H \times W$ size, where N_f is the number of frames and $H \times W$ is the frame size, let $segm\_map_t : H \times W \rightarrow [0, 1]$ be this pixel-level segmentation map. The first state, $state_0$ is initialized using the ground truth segmentation map, $segm\_map_0$.

$$segm\_map_t, state_t = segmentation(X_t, state_{t-1}). \tag{2}$$

For a uniform approach over tasks, we use the same convention for tracking. For this, $bbox_t$ in eq. 1 is replaced with $bbox\_map_t : H \times W \rightarrow [0, 1]$.

2.2.2 Evaluation

Nowadays, the most common way of measuring the quality of object segmentation for one frame is Intersection over Union (IoU) at pixel-level between the ground truth segmentation and the predicted segmentation (also called Jaccard similarity score). For video, we average over all frames. This way of simply using an average over a per frame metric is not very informative and does not adapt properly to the temporal dimension. Therefore, we introduce in Chapter 4, TCONT: a Temporal consistency metric that focuses on a video seen as a whole as opposed to a video seen just as a set of frames. Briefly, its formula is based on transforming the predictions with optical flow in both directions. For each frame, we average over those transformations and the original prediction and compute the IoU between ground truth and the average. Ideally, the average should land on a similar map, which represents a consistent representation.

IoU accuracy

TCONT

We come to this after observing that some methods could have a high IoU per frame, but missing the desired coherence and consistency of the object over time. And those limitations become highly visible specifically when you look at segmentation using the older tracking evaluation metrics like robustness. For instance, in real applications, the consequence of failing to track in one frame could lead to total failure, so you need to put more weight on having consistent predictions in time because a low IoU in one frame (one mistake) might impact the entire video.

2.2.3 Types of segmentation

Historically, object segmentation in video is more connected with salience, object discovery, foreground-background segmentation, formulated in the context of having access to the entire video, namely offline video object segmentation. But more recently, the main approach is also the online segmentation, in a similar way with tracking (with online or offline learning). Another direction in which segmentation solutions differ is the degree of detail. While some solutions work on a lower density grid, super-pixel level, or patches, our segmentation solution in Chapter 4 is very efficient, making the pixel-level, dense approach possible.

2.3 APPROACHES OVER THE YEARS

Very fast template matching using correlation filters, new and larger datasets enabling deep learning models, siamese nets used for template matching, IoU predictors, and bounding box regressors contributed over time to the significant advances in the field of object tracking. The main types of algorithms used in recent years were based on Correlation Filters, Basic Neural Networks, and in particular Siamese Networks. Next, I will briefly present a representative method for each of the approaches.

2.3.1 Discriminative Correlation Filters

Correlation-based trackers started with MOSSE [11] and involve applying the correlation operator between an input frame and the searching template (element-wise multiplication) in the Fourier domain. KCF [63] notices and takes advantage of the redundancies in the data matrix (when using multiple

patches from a frame), by using a circulant matrix. This approach taken in the Fourier space makes the solution both performant (due to exploiting all negative examples), but also extremely fast (up to 400 FPS). Next, CCOT [32] moves the problem in the contiguous space, making it possible to integrate feature maps of different resolutions, giving up speed. This chapter of discriminative correlation filters in online tracking ends up with ECO [29], which optimizes CCOT. It allows a fast adaptation to the appearance model with cheap updates (in the Fourier space) for each frame, with a multi-scale and similar sample grouping mechanism.

2.3.2 Basic Neural Networks

MDNet [122] is the first neural net approach that obtained effective representations for tracking. The core idea is that the neural network model has two parts, the first one is composed of domain-independent layers and the final one has domain-specific layers. The first part is pre-trained on a set of videos with tracking ground-truth. In contrast, the second is updated online using the current video, differing from one video sequence to another. This way, the online update is fast. In addition, the algorithm contains hard negative mining and both short-term negatives and long-term positives, a multi-scale bounding box regressor component trained only on highly confident frames. A more recent and efficient implementation for MDNet achieves promising results on recent benchmarks while running at 5 FPS.

2.3.3 Siamese Trackers

Siamese trackers rethink the correlation aspect of the tracking. To update the model given the target bounding boxes is time-consuming. SiamFC [9] is the first to pre-train the feature extractor similar to its final use (so that the correlation will make sense) and tracks based on cross-correlation between a template and a larger search area in the frame. All training is done offline, and no adaptations are performed online. SiamRPN [94] introduces the Region Proposal Network in the pipeline and SiamMask [171] a new branch for additional supervision, predicting the segmentation mask of the tracked object. Since most operations are done offline, the siamese approaches are very fast. Another important advantage is the easiness of building the training dataset, composed of image pairs, sampled from single images and labels as positive or negative.

2.3.4 Joint Segmentation and Tracking

Several solutions propose segmentation-based tracking or joint segmentation and tracking, each task enforcing the other. They are based on contour matching [24, 146] and propagation [5] or probabilistic formulations like Conditional Random Field (CRF) [98], CNN backbones common among the two tasks [172] or graph-based approaches [13, 75]. Our proposed solution is based on clustering using a graph representation of the video. It differs from previous approaches because we manage to find a good approximation of the problem such that the pixel-level clustering in video becomes tractable and the temporal dimension is naturally integrated into the formulation, while also achieving good state-of-the-art performance.

2.4 STATE-OF-THE-ART METHODS

Next, I will briefly introduce some of the best approaches over the standard benchmarks in tracking (each new valuable solution reports results for 5-7 datasets).

SIAMBAN Siamese Box Adaptive Network for Visual Tracking [20] replaces the need for multi-scale predictions or pre-defined anchor boxes for accurate bounding box predictions by using the fully convolutional network properties. The system predicts the object class and regresses the bounding box through two parallel network heads.

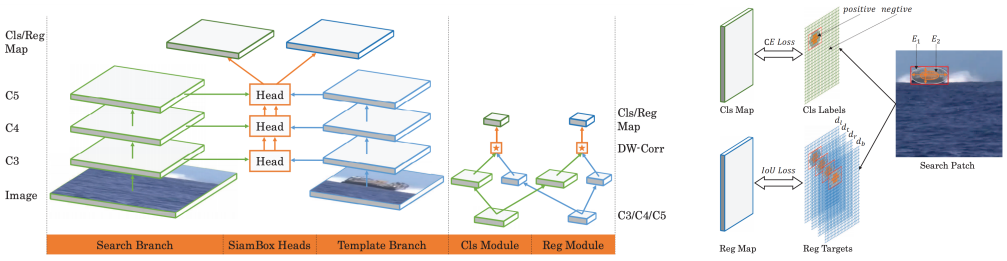

Figure 7: SiamBAN [20] pipeline.

ATOM AND PRDIMP ATOM [30] proposes a dedicated head for target estimation and refinement, based on an IoU regressor that iteratively improves the bounding box target prediction for each forward step. This learns to predict

the overlap between the target object and an estimated bounding box. Different from ATOM, PrDimp [33] formulates the uncertainty in the target estimation as probabilistic regression and integrates it in the tracking pipeline from ATOM.

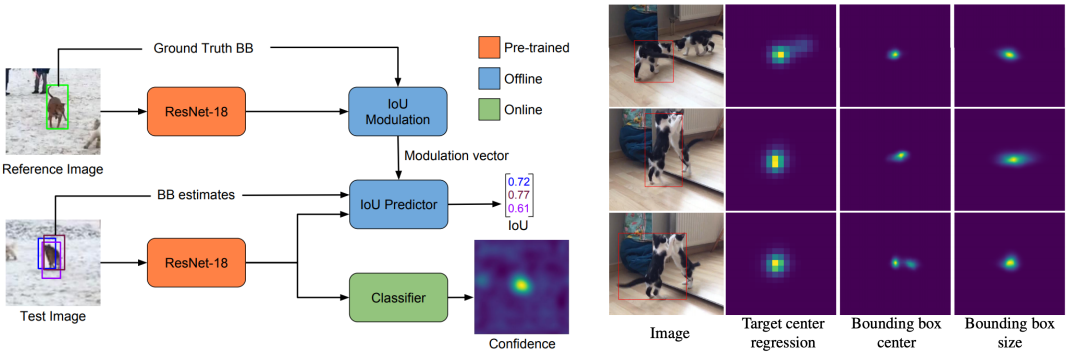

Figure 8: ATOM [30] and PrDimp [33] pipeline.

LTMU The main approach in long-term tracking is using offline trained Siamese nets. Different from that, High-Performance Long-Term Tracking with Meta-Updater [27] solution balances the importance of accommodating the appearance changes in the target with the system's robustness in this long-term context. The proposed framework consists of a verifier, a SiamRPN-based re-detector, the LTMU meta-updater, and a local online tracker that can be replaced with a different one.

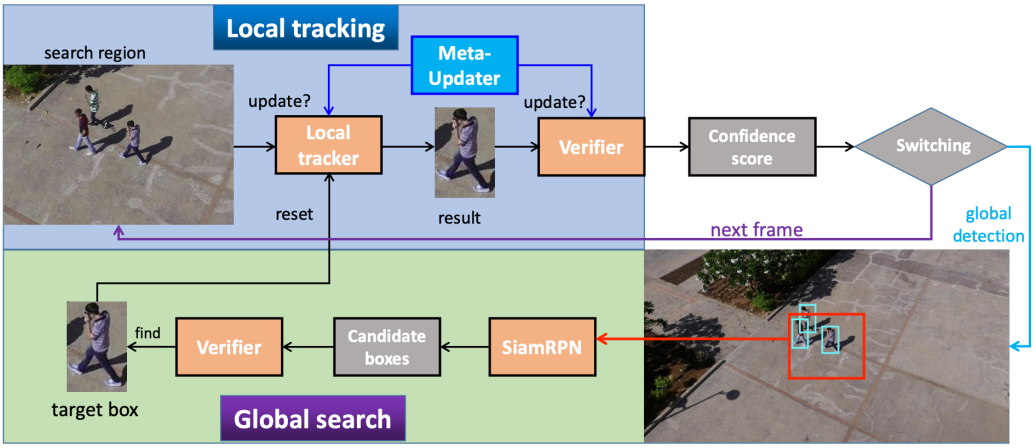

Figure 9: LTMU [27] pipeline.

OCEAN Object-aware Anchor-free Tracking [198] identifies and tackles the problem that the regression network in anchor-based methods is not trained for anchor boxes with low overlap, making it hard to recover when reaching those cases. The proposed solution fixes the imprecise bounding-box predictions and learns an object-aware feature that enhances the overlap and contributes to the target classification task.

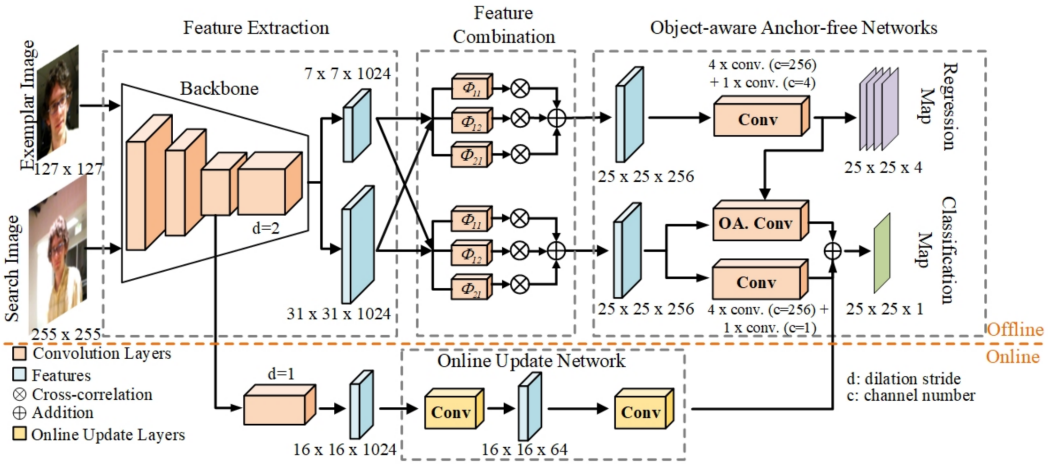

Figure 10: Ocean [198] line.

2.5 OTHER ORIGINAL APPROACHES

I will continue this chapter by describing several interesting tracking approaches and directions that introduce original aspects compared with the rest of the literature on tracking.

GRAPH CONVOLUTIONAL TRACKER Because of the real-time running constraints, a wide variety of siamese trackers base their algorithm only on per frame detection, ignoring the temporal context of the object. As a result, only the initial template is used to match the tracked object in further video frames. To tackle this spatio-temporal history of the target, GCT [50] proposes a graph of target parts (robust to occlusions). Using this context, GCT combines with the current context and learns an adaptive, attention-based graph, which is later used for cross-correlation.

META-TRACKER Meta-learning seems a naturally occurring idea for object tracking, since following the task formulation, the purpose in general tracking is exactly to build a one-shot learner: given a new video, predict the first-frame chosen object trajectory. Nevertheless, only recently has this approach started to gain some traction in the community [128, 166, 27]. Meta-tracker [128] uses a tracking-by-detection solution (MDNet) and learns to distinguish the target objects from background. It enforces this by using label shuffling in training (background with either 0 or 1 label), preventing it from collapsing in memorizing only specific targets appearance from training data and label the rest as background all the time.

VISUAL TRACKING VIA ADVERSARIAL LEARNING VITAL [148] uses adversarial learning to reduce the overfitting on a single frame, one of the most common problems in tracking-by-detection setup. The generator proposes discriminative features in target, while the discriminator classifies the input patch as target or not. This way, the generator, guided by the discriminator, will learn to extract only spatial features that are part of the object (not the background or other distractors). Usually, the problem in the adversarial setup is the numerical stability and convergence of the solution. VITAL proposes an adapted loss that accounts for negative-positive sample imbalance, adding a modulating factor inspired from the focal loss [99].

UNSUPERVISED DEEP TRACKING Using the time axis in the video as a supervision signal was a highly explored direction in vision, recently reaching competitive results. UDT [169] enforcing this with temporal cycles, assuming that a forward trajectory should be similar to the backward one, starting from the last bounding box of the target. Nevertheless, the approach is highly unstable and highly dependant on the training samples generation.

2.6 BENCHMARKS

2.6.1 Classical datasets

OTB-100 Object Tracking Benchmark [179] is one of the first video tracking larger benchmarks, being widely used, becoming largely overfitted over time, but a must-have reference in the experimental protocol. It contains 100 video sequences (updated in 2015 from 50 sequences), with 59k frames. With a di-

small, first, overfitted

verse range of labels for 11 (per frame) attributes for difficult cases (motion blur, scale variation, occlusion, low-resolution, background clutter, out of view, etc.), OTB-100 is a classic benchmark.

VOT16-20 VOT [79] is also one of the classics, coming with a different evaluation protocol, compared with any other dataset. When the tracker misses the target, it is reset, re-initializing it with the ground-truth. Another difference is that the final metrics are computed as a mean over sub-videos of different sizes. For each tested tracker, the framework reports the expected average overlap, accuracy, robustness, and speed. It contains only 60 sequences (with 21k frames) and focuses on short-term videos (and long-term later on), that do not apply particular pre-trained models of appearance. It comes with a semi-automatic ground truth bounding box annotation methodology.

small, different evaluation protocol

UAV123 Videos captured from UAV flying at low-altitude are quite different compared with the ones from other datasets. This dataset [118] contains 123 sequences (and 113k frames) and focuses on long-term aerial tracking, having very small tracking targets.

small, drone filmed

2.6.2 Newer datasets

TRACKINGNET It is the first large-scale dataset for tracking, allowing deep learning models to be trained on a video dataset rather than datasets for object detection in images like before. TrackingNet contains 30K sequences for training and 511 for testing (15 million frames), with labels available through a leaderboard server. The target objects appear in diverse scenarios for 27 classes, mainly covering persons, vehicles, animals, and several other objects. TrackingNet [120] was build by re-purposing YouTube-BoundingBoxes Dataset [135].

largest, leaderboard

GOT-10K Targeting large-scale Generic Object Tracking in the Wild, GOT-10k [68] has 10k sequences (1.5 million frames) of real-world moving objects, with manually labeled bounding boxes. Different from all the others, its split between train and test data provides a one-shot learning scenario, by having a zero-overlapped between train and test target classes. It also has a leaderboard server. The focus in this dataset is on short-term videos.

large, short-term, one-shot, leaderboard

LASOT This is also a Large scale dataset for Single Object Tracking, with 70 balanced classes spread across 1400 videos and 3.5 million manually annotated frames. LaSOT [42] has also labels for 14 challenging cases for tracking. It is mainly oriented towards long-term tracking, with an average video length of 2.5K frames, claiming that the performance of a tracker on this benchmark better reflects its real-world application results.

large, long-term

NFS The Need for Speed dataset [49] is the first one for tracking that proposes a higher frame rate (240 FPS, but it also has a 30 FPS variant). There are just 100 (long) videos (with 380K frames). The frames are also annotated with difficult cases labels (like occlusion, background clutter, fast motion). This dataset is relevant because it allows the tracker to explore and better analyze the case of a real-time application with higher frame rate acquisition, involving denser frames with smoother target movements.

small, high frame rate

3 TRACKING WITH CONSENSUS WITHIN A SOCIETY OF CLASSIFIERS

One of the main challenges in tracking is to adapt to object appearance changes over time, while avoiding drifting to background clutter. We address this challenge by proposing a deep neural network architecture composed of different parts, which functions as a society of tracking parts. The parts work in conjunction according to a certain policy and learn from each other in a robust manner, using co-occurrence constraints that ensure robust inference and learning. From a structural point of view, our network is composed of two main pathways. One pathway is more conservative. It carefully monitors a large set of simple tracker parts learned as linear filters over deep feature activation maps. It assigns different roles to parts, while promoting the reliable ones and removing the inconsistent ones. We learn these filters simultaneously in an efficient way, with a single closed-form formulation for which we propose novel theoretical properties. The second pathway is more progressive. It is learned completely online and thus it is able to better model object appearance changes. In order to adapt in a robust manner, it is learned only on highly confident frames, which are decided using co-occurrences with the first pathway. Thus, our system has the full benefit of two main approaches in tracking. The larger set of simpler filter parts offers robustness, while the full deep network learned online provides adaptability to change. As shown in the experimental section, our approach achieves state of the art performance on the challenging VOT17 benchmark, outperforming the existing published methods both on the general EAO metric as well as in the number of fails by a significant margin.

E. Burceanu, M. Leordeanu, Learning a Robust Society of Tracking Parts using Co-occurrence Constraints, European Conference on Computer Vision - Visual Object Tracking Workshop (ECCVW) 2018

3.1 ZOOMING OUT: TRACKING IN THE CONTEXT OF THE THESIS

We focus in this chapter on building a tracker that is based on multiple and diverse components, that manage to track the target object and its evolution over time from multiple points of view. We briefly point out next the key aspects introduced at the beginning of the thesis, putting this chapter in the context.

- **A. Space-time consistency**: The society of tracking parts evolves over time, by gradually promoting or downgrading each voting part, following its co-occurrence of prediction history with consensus at the previous points in time.
- **B. The power of the consensus**: Combine a pathway composed out of weak tracking parts learned over deep features with a complex end-to-end deep neural net pathway.
- **C. Exploiting multiple intermediate representations**: The object parts are linear classifiers over pre-trained VGG-16 features which can be seen as mid-level representations, while from the second pathway we use the final prediction (a dense map score for the center of the target).
- **D. A limited quantity of supervision**: The only supervision we get is through the pretrained VGG features for object classification in image, so the solution is completely unsupervised from the tracking task point of view.
- **E. Experts: Making use of existing models**: In this chapter, we use as existing knowledge only pre-trained VGG-16 features.

3.2 CONTEXT

Object tracking is one of the first and most fundamental problems that has been addressed in computer vision. While it has attracted the interest of many researchers over several decades of computer vision, it is far from being solved [79, 80, 145, 107, 179]. The task is hard for many reasons. Difficulties could come from severe changes in object appearance, presence of background clutter and occlusions that might take place in the video. The only information

given to the tracker is the bounding box of the object in the first frame. Thus, without knowing in advance the properties of the target object, the tracking algorithm must learn them on the fly. It must adapt correctly and make sure it does not jump toward other objects in the background. That is why the possibility of drifting to the background poses on of the main challenges in tracking.

Our proposed model, at the conceptual level, is composed of a large group of different tracking parts, functioning like a society, each with different roles and powers over the final decisions. They learn from each other using certain co-occurrence rules and are monitored according to their reliability. The way they function together gives them robustness. From a structural point of view, they are all classifiers within a large deep neural network structure, composed of two pathways, namely the FilterParts and the ConvNetPart pathways (see Figure 11). While the first insures robustness through the co-occurrence of a large number of smaller tracker parts, the second pathway insures the ability to adapt to subtle object changes. The ConvNetPart is fully trained online, end-to-end, and uses as ground-truth high confidence tracker responses that are decided together with the whole society of parts. We will refer to the frames of high confident tracker responses as Highly Confident Frames (HCFs). We provide more details in Section 3.4.2. Our idea of using as ground-truth only a small set of high precision points is also related to the recent work on unsupervised object discovery in video [59].

Our proposed approach is based on two key insights. One is the organization of the whole tracker into a large group of different types of classifiers, simpler and more complex, at different scales and with different levels of depth, as part of a larger neural network structure, that make decisions together based on mutual agreements. The second idea is the usage of co-occurrence constraints as basis for ensuring robustness, both for online training of the overall tracker, as well as for frame by frame inference.

RELATION TO PRIOR WORK. Existing trackers in the literature differ in terms of type of target region, appearance model, mathematical formulation and optimization. Objects can be represented by boxes, ellipses [84], superpixels [173] or blobs [54]. The appearance model can be described as one feature set over the region or as an array of features, one for each part of the target [46, 142, 85].

In recent years, trackers based on discriminative correlation filters (DCF), such as MOSSE [11] and KCF [63], achieved the best results on public benchmarks. Newer models like Staple [8], CCOT [32] and ECO [29] provide consistent improvements by adding to the DCF model different components, such

as multi-channel feature maps and robust scale estimation[31]. CCOT, for instance, proposes to learn continuous convolution parameters by optimizing a function that results from transforming the original feature space into a continuous one and applying onto it the continuous convolutions. While learning the parameters continuously, at every frame, provides adaptability to the tracker, overfitting to noise and drifting could pose a threat. In order to reduce overfitting, ECO comes with a generative model over training samples. Nevertheless, most recent tracking approaches still suffer from overfitting to background noise, which causes tracker failure.

A common approach for top trackers in the recent literature is to model object features with deep convolutional networks (CNNs). To address the issue of robustness against background noise in the case of online training of CNNs, the TCNN [121] algorithm, for example, maintains stability of appearance through a tree structure of CNNs. MLDF [79] uses discriminative multi-level deep features between foreground and background together with a Scale Prediction Network. Another approach, MDNET [123] is used as starting point for many CNN trackers. For instance, SSAT [79] uses segmentation to properly fit the bounding box and builds a separate model to detect whether the target in the frame is occluded or not. It uses this decision to consider frames for training the shape segmentation model.

Another line of object tracking research is the development of part-based models. They tend to be more resistant to appearance changes and occlusion. Their multi-part nature gives them robustness against noisy appearance changes in the video. In recent benchmarks however, they did not obtain the top results. For instance, in VOT16 [79] challenge, while the number of part-based trackers, such as DPCF [3], CMT [124], DPT [104], BDF [2], was relatively high (25 %), the best one of the group, SHCT [39], is on the 14th place overall. SHCT [39] is a complex system using a graph structure of the object that models higher order dependencies between object parts, over time. As it is the case with deep convolutional networks, we believe complex systems are prone to overfitting to background noise without a high precision way of selecting their unsupervised online training frames.

Our proposed model combines the best of two worlds. On one hand it uses a powerful deep convolutional network trained on high confidence frames, in order to learn features that better capture and adapt to object appearance changes. On the other hand, it uses the power of a large group of simpler classifiers that are learned, monitored, added and replaced based on co-occurrence constraints. Our approach is validated by the very low failure rate of our tracker,

relative to the competition on the VOT2017 benchmark (our 0.76 failure rate vs. the next best one, 1.13).

OUR MAIN CONTRIBUTIONS: 1) Our first contribution is the design of a tracker as a dual-pathway network, with FilterParts and ConvNetPart pathways working in complementary ways within a robust society of tracking parts. FilterParts is more robust to background noise and uses many different and relatively simple trackers learned on top of deep feature activation maps. ConvNetPart is better capable to learn object appearance and adapt to its changes. It employs a deep convolutional network that is learned end to end during tracking using unsupervised high confidence frames for ground-truth.
2) Our second contribution is that every decision made for learning and inference of the tracker is based on robust co-occurrence constraints. Through co-occurrences over time we learn which FilterParts classifiers are reliable or not. Thus we can change their roles and add new ones. Also, through co-occurrences between the vote maps of the two pathways, we decide which frames to choose for training the ConvNetPart path along the way. Last but not least, through co-occurrences we decide the next object center by creating a combined vote map from all reliable parts.
3) Our third contribution addresses a theoretical point, in Section 3.4.1. We show that the efficient closed-form formulation for learning object parts simultaneously in a one sample vs. all fashion is equivalent to the more traditional, but less efficient, balanced one vs. all formulation.

3.3 INTUITION AND MOTIVATION

Visual tracking is about being able to adapt the current knowledge about an object model to changes that take place continuously in the stream of video. It is also about being stable and robust against background noise during frame by frame inference. A tracking model composed of many parts, with different degrees of complexity, could use the co-occurrences of their responses in order to monitor over time, which parts are reliable and which are not. This would provide **stability**. They could also be used to train the more complex ConvNetPart pathway only on high-confidence frames on which the two pathway responses strongly co-occur in the same region. Thus, they could provide **robust adaptability**. Last but not least, by taking in consideration only where sufficient parts votes co-occur for the object center, we could also achieve **ro-**

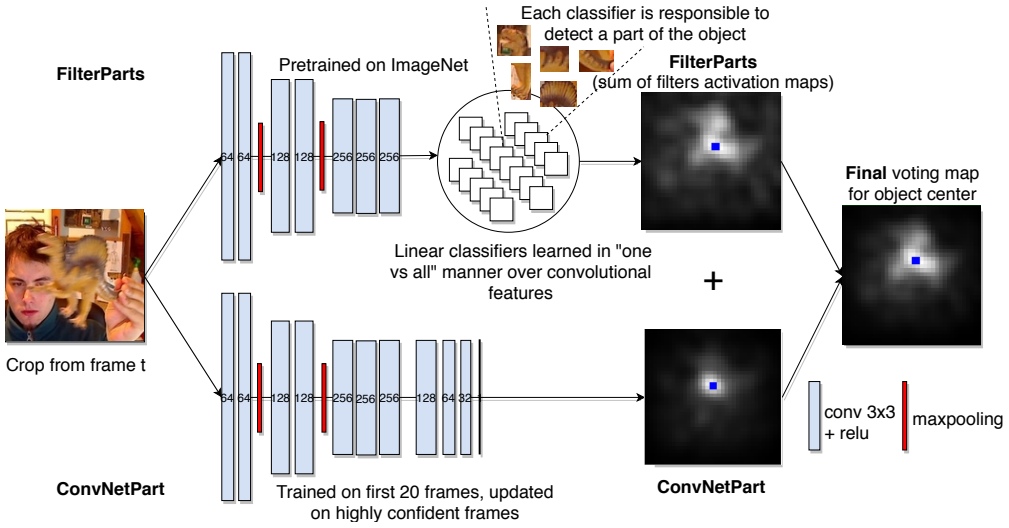

Figure 11: STP overview: The tracker functions as a society of parts. It combines the vote for center maps from all parts over two main pathways, FilterParts and ConvNetPart. The two pathways are learned differently. The FilterParts classifiers once learned are fixed individually but adapt as a group. The ConvNetPart is trained end-to-end with back-propagation over unsupervised tracker outputs from previous highly confident frames (HCFs).

bust frame to frame performance. We discuss each aspect in turn, next:

1) STABILITY THROUGH STEADINESS. We consider a part to be reliable if it has showed independently and frequently enough agreement in voting with the majority of the other parts - a statistically robust measure. A certain part is at the beginning monitored as a **candidate part**, and not used for deciding the next tracker move. It is only after a candidate part's vote for the object center co-occurred frequently enough at the same location with the majority vote, we promote the candidate to become a **reliable part**. From then on its vote will participate in the final vote map. Tracking parts that display consistent reliable behaviour over relatively long periods of time are promoted to the status of **gold members** - they will permanently have the right to vote, they cannot be downgraded and will not be monitored. In similar fashion, for the ConvNetPart, we always keep the tracker output from the first frames (=20) in video during the learning updates of the convolutional net. We further ensure robustness by favoring current tracker prediction to be close to the previous one. We use a tracker location uncertainty mask, centered around the previous

center location.

2) ROBUST ADAPTATION. The tracker is able to continuously adapt by adding candidate parts and removing unreliable ones. It also adapts by learning the ConvNetPart on high confidence frames accumulated over time. For object parts along the FilterParts pathway, gaining reliability, loosing it or becoming a gold member, can happen only over time. It is the temporal buffer, when tracking parts are monitored, which ensures both stability and the capacity to adapt to new conditions in a robust way. In time, the second pathway has access to a larger and larger set of reliable HCFs that are monitored through co-occurrences between the voted tracker centers of the two pathways. By training the net on larger sets of high quality frames we achieve both stability and capacity to adapt to true object appearance changes. As mentioned previously, HCFs used as ground-truth comes from past frames where the center given by the FilterParts alone co-occurred at the same location (within a very small distance) with the one given by the ConvNetPart. In Figure 13 we show why the distance between the two pathways is a good measure for frame confidence - the strong correlation between the distance between the tracker and the ground-truth and the distance between the centers voted along the two pathways is evident. In Figure 12 we also show qualitative results to demonstrate how ConvNetPart and FilterParts could better work together in conjunction, than separately.

3) ROBUST FRAME TO FRAME TRACKING. Each part produces a prediction map for the object center. For the FilterParts pathway, an average vote map is obtained from all reliable parts. That map is then added to the ConvNetPart final vote map, with a strong weight given to the FilterParts pathway. This is the final object center map in which the peak is chosen as the next tracker location. It is thus only through the same strong **co-occurrences** of votes at a single location that we robustly estimate the next move.

3.4 OUR APPROACH

TRACKER STRUCTURE. At the structural level, the Society of Tracking Parts (STP) has two pathways: FilterParts and ConvNetPart pathways (Figure 11).

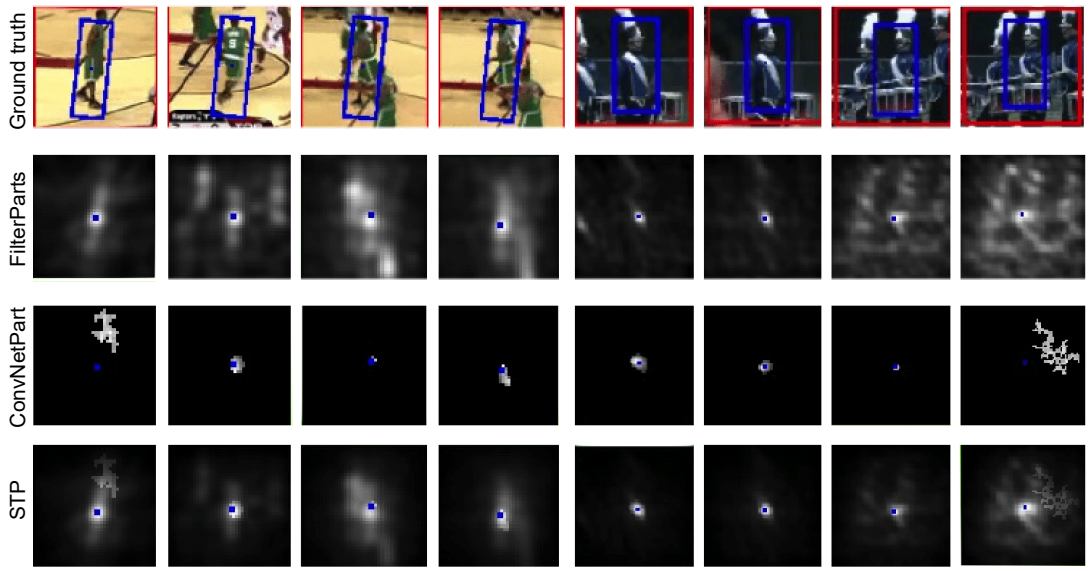

Figure 12: Qualitative comparisons between FilterParts, ConvNetPart and the final (STP) voting maps. Often, in complicated scenarios, the ConvNetPart vote could be of better quality. There are also relatively simple cases where the ConvNetPart activation map look bad, and we need the stability of the FilterParts. The final vote map (STP), provides a more robust maximum. The blue point represent the center of the final vote.

The combination of the two is presented in Alg. 1. The first pathway is formed of smaller object parts that are classifiers represented by linear classifiers over activation maps, from a pre-learned convolutional net. The ConvNetPart pathway is a deep convolutional net, with the same structure as the first pathway up to a given depth. Now we present the actual CNNs structures of the two pathways:

The ConvNetPart is a fully convolutional network, where the first part (common as architecture with FilterParts features extractor) has 7 convolutional layers, with 3x3 filters (each followed by relu) and 2 maxpooling layers (2x2). It is inspired from the VGG architecture [144]. The second part, is composed of 4 convolutional layers with 3x3 filters, having the role to gradually reduce the number of channels and computing the segmentation mask for object center prediction. We could have tested with different, more recent architectures, but in our experiments this architecture was strong enough.

TRACKING BY CO-OCCURRENCES OF PART VOTES: The tracker always chooses as its next move at time t, the place (the center of the bounding box) l_{t+1} where

Algorithm 1 - STP: Society of Tracking Parts

Input: $\text{frame}_{1..m}$ - video frames, bbox_1 - first frame ground-truth bounding box

Result: $\text{bbox}_{2..m}$ - tracking predictions

```
// Initialize the two pathways
filter_parts.init()
convnet_part.init()
while t < m do
    // select search zone - crop around last prediction
    crop_t = crop(frame_t, bbox_{t-1}, margin)
    // track on each pathway
    F_t = filter_parts.track(frame_t, crop_t)
    C_t = convnet_part.track(frame_t, crop_t)
    // combine voting maps
    hcf_frame = is_agreement(F_t, C_t)
    if hcf_frame then
        P_t = (αF_t + (1 − α)C_t) · M_c
    else
        P_t = F_t
    bbox_t = extract_bbox(frame_t, crop_t, P_t)
    // update trackers
    filter_parts.update(frame_t, bbox_t)
    convnet_part.update(frame_t, bbox_t, t, hcf_frame)
    t ← t + 1
```

there is the largest accumulation of votes in P_t, its final object center prediction map. For each filter part i, along the FilterParts pathway, there is an activation map F_{ti}, computed as the response of the classifier c_i corresponding to that part over the search region. The activation maps of filter parts are each shifted with the part displacement from object center and added together to form the overall F_t. Each displacement is fix and it represents the relative position to the center of the object, when each part was selected as candidate. When all filter parts are in strong agreement, all votes from F_t focus around a point. For the second pathway, the object center prediction map C_t is the output of the ConvNetPart network, given the same image crop input as to FilterParts. After smoothing F_t with a small Gaussian filter, it is added to C_t. The final prediction map P_t is then obtained by multiplying pixelwise the linear com-

Algorithm 2 - STP: FilterParts

```
function TRACK(frame_t, crop_t)
    filter_over_crop(frame_t, crop_t, all_parts)
    F_t = vote_parts(all_parts.reliable, all_parts.gold)
    update_stats(all_parts)
    return F_t
procedure UPDATE(frame_t, bbox_t)
    if not update_step then
        return
    // update candidate and reliable parts
    for all part_i in (all_parts.reliable, all_parts.candidate) do
        freq_i ← frequency_close_to_prediction(part_i)
        if freq_i > p_+ then
            promote(part_i)
        else if freq_i < p_− then
            discard(part_i)
    // budgeting
    keep_top_k(all_parts.reliable, all_parts.gold)
    // extract new candidates from current target prediction
    proposed_parts = build_new_classifiers(frame_t, bbox_t)
    candidate_parts = filter_discriminatives(proposed_parts)
```

bination of C_t and F_t, with a center uncertainty mask M_c, around the center in the previous frame. M_c is a circular soft mask, with exponential decay in weights, as the distance from the previous center prediction increases. Thus, $P_t = (\alpha F_t + (1-\alpha)C_t) \cdot M_c$, where $\cdot$ denotes pixelwise multiplication. M_c encourages small center movements at the expense of large, sharp, abrupt ones. The maximum in P_t is chosen as the next center location l_{t+1}. We present the steps for FilterParts pathway in Alg. 2.

3.4.1 Learning along the FilterParts pathway

STP chooses in the FilterParts update phase new parts to add as candidates. They are classifiers, of different sizes and locations, represented as linear filters over activation maps of deep features. To each part it corresponds a patch, within the tracker's main bounding box. Only patch classifiers that are highly discriminative from the rest are selected. One is considered discriminative if

the ratio between the response on its own corresponding patch (the positive patch) and the maximum response over negatives is larger than a threshold t_d. Positive patches are selected from the inside of the bounding box, while (hard) negatives are selected as patches from outside regions with high density of edges. We sample patches from a dense grid (2 pixels stride) of 3 sizes. The small ones will see local appearance and the larger ones will contain some context. A point in grid is covered only by one selected discriminative patch, at one size. The smaller ones have priority and we search the next size for the patch centered in the grid point only if the smaller patch is not discriminative enough. The object box is completely covered when each pixel is covered by any given patch. A simple budgeting mechanism is added, in order to limit the speed impact. When too many parts of a certain patch size become reliable $> N_{max}$, we remove the new reliable ones which are most similar to older parts, based on simple dot product similarity for the corresponding classifiers.

MATHEMATICAL FORMULATION FOR FILTER PARTS CLASSIFIERS. Preliminary version of FilterParts computation was previously presented in [14]. For a given feature type let $\mathbf{d}_i \in \mathbb{R}^{1\times k}$ be the i-th descriptor, with k real elements, corresponding to a patch window at a certain scale and location relative to the object bounding box. In our case, the descriptor $\mathbf{d}_i$ is a vector version of the specific patch concatenated over all activation map channels over the considered layers of depth in the FilterParts pathway. Our formulation is general and does not depend on a specific level of depth - features could as well be simple pixel values of any image channel. Let then $\mathbf{D}$ be the data matrix, formed by putting all descriptors in the image one row below the other. Given n as the total number of descriptors (parts), let $\mathbf{y}_i \in \mathbb{R}^n$ be the label array for the i-th descriptor, with $y_i(i) = 1$ and $y_i(j) = 0$ for $j \neq i$.

We learn the optimal linear classifier $\mathbf{c}_i$ that separates $\mathbf{d}_i$ from the rest of the patches, according to a regularized linear least squares cost, which is both fast and accurate. Classifier $\mathbf{c}_i$ minimizes the following cost ([119] Ch. 7.5):

$$\min \frac{1}{n}\|\mathbf{D}\mathbf{c}_i - \mathbf{y}_i\|^2 + \lambda \mathbf{c}_i^\top \mathbf{c}_i. \tag{3}$$

In classification tasks the number of positives and negatives should be properly balanced, according to their prior distributions and the specific classifier used. Different proportions usually lead to different classifiers. In linear least squares formulations weighting differently the data samples could balance

learning.

LEARNING WITH ONE SAMPLE VERSUS ALL. The idea of training one classifier for a single positively labeled data sample has been successfully used before, for example, in the context of training SVMs [108]. Normally, when using very few positive samples for training a ridge regression classifier, weighting is applied to balance the data. Otherwise the classifier might learn to miss positive samples entirely. Here we show that it is possible, when a single positive sample is used, to obtain the same result with a single positive sample without weighting, as if balancing was applied. We show a novel result, that while the magnitude of the corresponding classifier vector is different for the single positive data sample case, its direction remains unchanged w.r.t. the balanced case.

THEOREM 1. For any positive weight w_i given to the positive i-th sample, when the negative labels considered are 0 and the positive label is 1 and all negatives have the same weight 1, the solution vector to the weighted least squares version of Eq. 3 will have the same direction (it might differ only in magnitude). In other words, it is invariant under L2 normalization.

PROOF. Let $\mathbf{c}_i$ be the solution to Eq. 3. At the optimum the gradient vanishes, thus the solution respects the following equality $(\mathbf{D}^\top\mathbf{D} + \lambda\mathbf{I}_k)\mathbf{c}_i = \mathbf{D}^\top\mathbf{y}_i$. Since $y_i(i) = 1$ and $y_i(j) = 0$ for $j \neq i$, it follows that $(\mathbf{D}^\top\mathbf{D} + \lambda\mathbf{I}_k)\mathbf{c}_i = \mathbf{d}_i$. Since the problem is convex, with a unique optimum, a point that obeys such an equality must be the solution. In the weighted case, a diagonal weight $n \times n$ matrix $\mathbf{W}$ is defined, with different weights on the diagonal $w_j = \mathbf{W}(j,j)$, one for each data sample. In that case, the objective cost optimization in Eq. 3 becomes:

$$\min \frac{1}{n}\|\mathbf{W}^{\frac{1}{2}}(\mathbf{D}\mathbf{c}_i - \mathbf{y}_i)\|^2 + \lambda\mathbf{c}_i^\top\mathbf{c}_i. \tag{4}$$

We consider when all negative samples have weight 1 and the positive one is given w_i. Now we show that for any w_i, if $\mathbf{c}_i$ is an optimum of Eq. 3 then there is a real number q such that $q\mathbf{c}_i$ is the solution of the weighted case. The scalar q exists if it satisfies $(\mathbf{D}^\top\mathbf{D} + \mathbf{d}_i\mathbf{d}_i^\top(w_i - 1) + \lambda\mathbf{I}_k)q\mathbf{c}_i = w_i\mathbf{d}_i$. And, indeed, it can be verified that $q = \frac{w_i}{1+(w_i-1)(\mathbf{d}_i^\top\mathbf{c}_i)}$ satisfies the required equality.

DETAILED PROOF It is easy to obtain the closed form solution for c_i, from linear ridge regression formulation ([119] Ch. 7.5), by minimizing the convex

cost $\frac{1}{n}\|\mathbf{D}\mathbf{c}_i - \mathbf{y}_i\| + \lambda \mathbf{c}_i^\top \mathbf{c}_i$, we get Eq. 5. It results the well known solution by inverting the positive definite matrix $\mathbf{D}^\top\mathbf{D} + \lambda\mathbf{I}_k$.

$$(\mathbf{D}^\top\mathbf{D} + \lambda\mathbf{I}_k)\mathbf{c}_i = \mathbf{D}^\top\mathbf{y}_i \tag{5}$$

In "one vs all" context, we choose $y_i^\top = \begin{bmatrix} 0 & 0 & \dots & 1 & \dots & 0 & 0 \end{bmatrix}$, with 1 only on the i^{th} position. So, the multiplication with y_i selects a column form $\mathbf{D}$: $\mathbf{D}^\top\mathbf{y}_i = \mathbf{d}_i$. Eq. 5 becomes:

$$(\mathbf{D}^\top\mathbf{D} + \lambda\mathbf{I}_k)\mathbf{c}_i = \mathbf{d}_i \tag{6}$$

When building classifiers, the classes should be balanced as numbers of entries. This ensures that the comparison between the activation scores for two different classifiers is valid. In "one vs all", usually the positive class has fewer instances than the negative class. So we needed to use a weighted solution for linear ridge regression [67], in order to build a balanced classifier for our "part of the object vs others/context" classifiers.

We prove that for a specific form of the weights, the weighting can be applied after computing the simple version (closed form linear regression, Eq. 6). This is very important for our algorithm, because for the simple ridge regression we need to compute only one matrix inverse for all classifiers in one step, one matrix that all of them will share: $(\mathbf{D}^\top\mathbf{D} + \lambda\mathbf{I}_k)^{-1}$ from Eq. 6. For the weighted case, the closed form solution (as in [67]) would be different from classifier to classifier:

$$(\mathbf{D}^\top\mathbf{W}_i\mathbf{D} + \lambda\mathbf{I}_k)_{,i} = \mathbf{D}^\top\mathbf{W}_i\mathbf{y}_i \tag{7}$$

The weight matrix for a classifier $\mathbf{W_i}$ has the following form:

$$\mathbf{W_i} = \mathbf{I}_n + \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \dots & & & & & & \\ 0 & 0 & \dots & w_i & \dots & 0 & 0 \\ \dots & & & & & & \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 \end{bmatrix} = \mathbf{I}_n + \mathbf{W}_{sparse_i} \tag{8}$$

with 0s and w_i only on i^{th} position on the diagonal, i being the index of the positive patch in data matrix, $\mathbf{D}$.

Replacing Eq.8 in Eq.7, and observing that $\mathbf{D}^\top \mathbf{W}_{sparse_i} = w_i[0|\mathbf{d}_i|0]$, the right hand side becomes: $\mathbf{D}^\top \mathbf{W}_i \mathbf{y}_i = \mathbf{D}^\top (\mathbf{I}_n + \mathbf{W}_{sparse_i})\mathbf{y}_i = \mathbf{D}^\top \mathbf{y}_i + w_i[0|\mathbf{d}_i|0]\mathbf{y}_i = \mathbf{d}_i + w_i \mathbf{d}_i$. So, for the right term we get:

$$\mathbf{D}^\top \mathbf{W}_i \mathbf{y}_i = (1 + w_i)\mathbf{d}_i \tag{9}$$

By doing the same operations on the left term: $\mathbf{D}^\top \mathbf{W}_i \mathbf{D} = \mathbf{D}^\top (\mathbf{I}_n + \mathbf{W}_{sparse_i})\mathbf{D} = \mathbf{D}^\top \mathbf{D} + w_i[0|\mathbf{d}_i|0]\mathbf{D} = \mathbf{D}^\top \mathbf{D} + w_i \mathbf{d}_i \mathbf{d}_i^\top$, the Eq. 7 can be rewritten:

$$(\mathbf{D}^\top \mathbf{D} + w_i \mathbf{d}_i \mathbf{d}_i^\top + \lambda \mathbf{I}_k)_{,i} = (1 + w_i)\mathbf{d}_i \tag{10}$$

Let $\theta_i = q_i \mathbf{c}_i$, where $\mathbf{c}_i$ is the solution for linear ridge regression (Eq. 6) and $q_i \in \mathbb{R}$. Then Eq. 10 becomes: $(\mathbf{D}^\top \mathbf{D} + w_i \mathbf{d}_i \mathbf{d}_i^\top + \lambda \mathbf{I}_k) q_i \mathbf{c}_i = (1 + w_i)\mathbf{d}_i$. From Eq. 6, by simplifying terms we obtain $q_i \mathbf{d}_i + q_i w_i \mathbf{d}_i \mathbf{d}_i^\top \mathbf{c}_i = (1 + w_i)\mathbf{d}_i$. Then, by multiplying at left with $\frac{\mathbf{d}_i^\top}{\|\mathbf{d}_i\|_2^2}$, we get:

$$q_i + q_i w_i \mathbf{d}_i^\top \mathbf{c}_i = (1 + w_i) \tag{11}$$

So, the solution for q_i is (w_i is $n - 1$, because in "one vs all" classification, all elements in $\mathbf{D}$ are negative samples, except for one, the i^{th}):

$$q_i = \frac{(1 + w_i)}{1 + w_i \mathbf{d}_i^\top \mathbf{c}_i} = \frac{n}{1 + (n-1)\mathbf{d}_i^\top \mathbf{c}_i} \tag{12}$$

Therefore, we proved that if $\mathbf{c}_i$ is the unique solution of linear ridge regression (since $\mathbf{D}^\top \mathbf{D} + \lambda \mathbf{I}_k$ is always invertible, the solution in Eq. 9 is unique), then $q_i \mathbf{c}_i$ (q_i from Eq. 12) is the unique solution of Eq. 6 ($\mathbf{D}^\top \mathbf{D} + w_i \mathbf{d}_i \mathbf{d}_i^\top + \lambda \mathbf{I}_k$ is always invertible, since it is also positive definite).

EFFICIENT MULTI-CLASS FILTER LEARNING. The fact that the classifier vector direction is invariant under different weighting of the positive sample suggests that training with a single positive sample will provide a robust and stable separator. The classifier can be re-scaled to obtain values close to 1 for the positive samples. Theorem 1 also indicates that we could reliably compute filter classifiers for all positive patches in the bounding box at once, by using a single data matrix $\mathbf{D}$. We form the target output matrix $\mathbf{Y}$, with one target labels column $\mathbf{y}_i$ for each corresponding sample $\mathbf{d}_i$. Note that $\mathbf{Y}$ is, in fact, the $n \times n$ $\mathbf{I}_n$ identity matrix. We now write the multi-class case of the ridge regression model and finally obtain the matrix of one versus all classifiers, with one column classifier for each tracking part: $\mathbf{C} = (D^\top D + \lambda \mathbf{I}_k)^{-1} \mathbf{D}^\top$. Note that $\mathbf{C}$ is a regularized pseudo-inverse of $\mathbf{D}$. $\mathbf{D}$ contains one patch descriptor per

line. In our case, the descriptor length is larger than the number of positive and negative samples, so we use the Matrix Inversion Lemma [119](Ch. 14.4.3.2) and compute $\mathbf{C}$ in an equivalent form:

$$\mathbf{C} = \mathbf{D}^\top(\mathbf{D}\mathbf{D}^\top + \lambda\mathbf{I}_n)^{-1}. \tag{13}$$

Now the matrix to be inverted is significantly smaller ($n \times n$ instead of $k \times k$).

MATRIX INVERSION LEMMA Consider a general partitioned matrix $\mathbf{M} = \begin{bmatrix} \mathbf{E} & \mathbf{F} \\ \mathbf{G} & \mathbf{H} \end{bmatrix}$, with $\mathbf{E}$ and $\mathbf{H}$ invertible (Matrix Inversion Lemma [119], Ch. 4.3.4.2). Then the following relation takes place:

$$(\mathbf{E} - \mathbf{F}\mathbf{H}^{-1}\mathbf{G})^{-1}\mathbf{F}\mathbf{H}^{-1} = \mathbf{E}^{-1}\mathbf{F}(\mathbf{H} - \mathbf{G}\mathbf{E}^{-1}\mathbf{F})^{-1} \tag{14}$$

By making the replacement: $\mathbf{E} = \lambda\mathbf{I}_k$, $\mathbf{H} = \mathbf{I}_n$, $\mathbf{F} = \mathbf{D}^\top$, $\mathbf{G} = -\mathbf{D}$ ($\mathbf{E}$ and $\mathbf{H}$ are invertible) and rearranging the terms, we obtain [119] (Ch. 14.4.3.2):

$$(\mathbf{D}^\top\mathbf{D} + \lambda\mathbf{I}_k)^{-1}\mathbf{D}^\top = \mathbf{D}^\top(\mathbf{D}\mathbf{D}^\top + \lambda\mathbf{I}_n)^{-1} \tag{15}$$

We observe that the first term in Eq. 15 is part of the closed form solution for the linear regression (without labels y_i). So, we can replace it with the one easier to compute. Since the bottleneck here is inverting the positive definite matrix $\mathbf{D}^\top\mathbf{D} + \lambda\mathbf{I}_k$ or $\mathbf{D}\mathbf{D}^\top + \lambda\mathbf{I}_n$, we will choose the easiest to invert. And this is the smaller one. In our case, n is the number of patches, and k is the number of features in each patch (equal to the patch area in feature space $\times$ number of channels, which is 256). A roughly approximation for n is 500 and approximations for k are $6400 \approx 5 \times 5 \times 256$, $74000 \approx 17 \times 17 \times 256$ and bigger for patches of bounding box size.

The second solution for computing the classifier is inverting a matrix two orders of magnitude smaller (as number of elements) than the first solution. So we choose the second part of Eq. 15 for the closed form solution.

RELIABILITY STATES. The reliability of a filter part i is estimated as the frequency f_i at which the maximum activation of a given part is in the neighborhood of the maximum in the final activation P_t where the next tracker center l_{t+1} is chosen. If a part is selected for the first time, it is considered a candidate part. Every U frames, the tracker measures the reliability of a given part, and promotes parts with a reliability larger than a threshold $f_i > p_+$, from candidate state (C) to reliable state (R) and from reliable (R) to gold (G). Parts

Algorithm 3 - STP: ConvNetPart

```
1: function TRACK(frame_t, crop_t)
2:     C_t = cnn.forward(frame_t, crop_t)
3:     return C_t
4: procedure UPDATE(frame_t, bbox_t, t, hcf_frame)
5:     if t < FIRST_FRAMES then
6:         gold_trainset.add(frame_t, bbox_t)
7:     else if hcf_frame then
8:         trainset.add(frame_t, bbox_t)
9:     if not update_step then
10:        return
       // budgeting
11:    trainset.keep_max_samples(N_max)
       // update model
12:    finetune_cnn(trainset)
```

that do not pass the test $f_i \leqslant p_-$ are removed, except for gold ones which are permanent.

BOUNDING BOX ESTIMATION PER FRAME We compute the bounding boxes for each frame in a simple manner. We considered all parts from the FilterParts pathway which voted and agreed on the tracker center for the current frame. We used their activation maps (original locations, not displaced relative to the object center) in order to estimate an affine transformation of the object and used the transformed bounding box as our tracker prediction.

3.4.2 Learning along the ConvNetPart Pathway

The end output of the ConvNetPart pathway is an object center prediction map, of the same size as the one produced along the FilterParts pathway. Different from FilterParts, the second pathway has a deeper architecture and a stronger representation power, being trained end-to-end with back-propagation along the video sequence. First, we train this net for the first 20 frames, using as ground-truth the FilterParts center prediction (expected to be highly accurate). Afterwards, the ConvNetPart is considered to be reliable part and it will contribute, through its center prediction, to the final tracker prediction.

From then on, the ConvNetPart will be fine-tuned using as ground-truth the final tracker predictions on highly confident frames (HCFs). This will ensure that we keep the object appearance up to date, and we won't drift in cases of local occlusion or distractors. We present the steps from the ConvNetPart pathway in Alg. 3. Results from Table 4 supports our decision.

SELECTING TRAINING SAMPLES FROM HIGHLY CONFIDENT FRAMES. We call HCF (Highly Confident Frame) a frame on which the distance between FilterParts and ConvNetPart votes for object center prediction is very small. When the two pathways vote almost on the same center location, we have high confidence that the vote is correct. In order to balance efficiently the number of updates with keeping track of object appearance changes, we do the following. First, we accumulate frames of high confidence and second, at regular intervals, we fine tune the network using the accumulated frames. The assumption we made is that on HCFs, our tracker is closer to ground-truth than in the other frames. This is confirmed in Figure 13. 11% of all frames are HCFs. More extensive tests for validating HCF usefulness are described in Section 3.5.

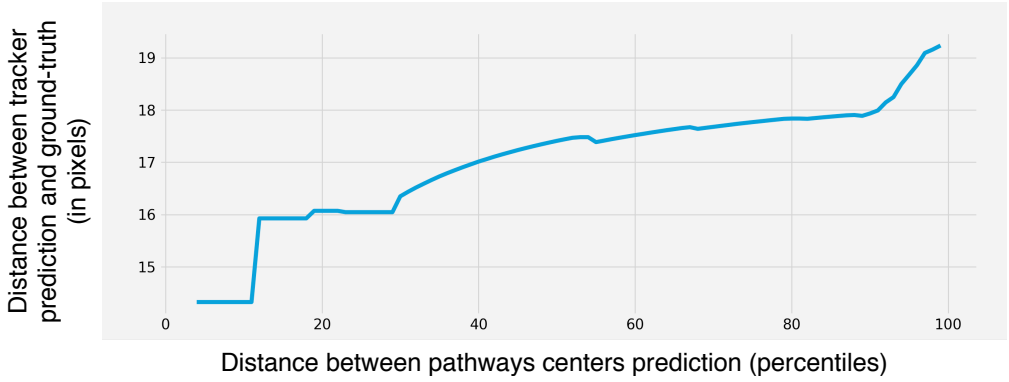

Figure 13: The plot shows the expected distance to ground-truth for a given distance between the centers predicted by the two pathways. As seen, the correlation is strong and it is therefore used for selecting in an unsupervised way high confidence frames. We choose HCFs from the first 11% percentile. Experiments run on VOT16 dataset.

TECHNICAL DETAILS FOR TRAINING THE CONVNETPART. For each training frame, we use as input an image crop around the object (Figure 14). The ground-truth is given as a segmentation map of the same size, with a circle stamp in the center. We increase robustness and generalization by randomly

shifting the image along with its ground-truth - thus we also augment the data by providing two such randomly shifted pairs, per frame. We use the Adam optimizer, with learning rate lr, from Pytorch [97], at first for k epochs on the first N(=20) frames, then on k epochs on each update, after each U frames. In the update step, we always use as samples the last N HCFs and the first N frames - thus we combine the new changes with the initial appearance. The training loss was a simple MSE $= \frac{\sum(x_i - y_i)^2}{n}$. Note that we did not experiment with many architectures and loss functions, which might have further improve performance.

PARAMETERS. We use the following parameters values in all our experiments from Section 3.5: $\alpha = 0.6$, $U = 10$ frames, $t_d = 1.4$, $p_+ = 0.2$, $p_- = 0.1$, $k = 10$ epochs, $N = 20$ frames, $lr = 1e-5$ and $N_{max} = 200$ parts for each scale size.

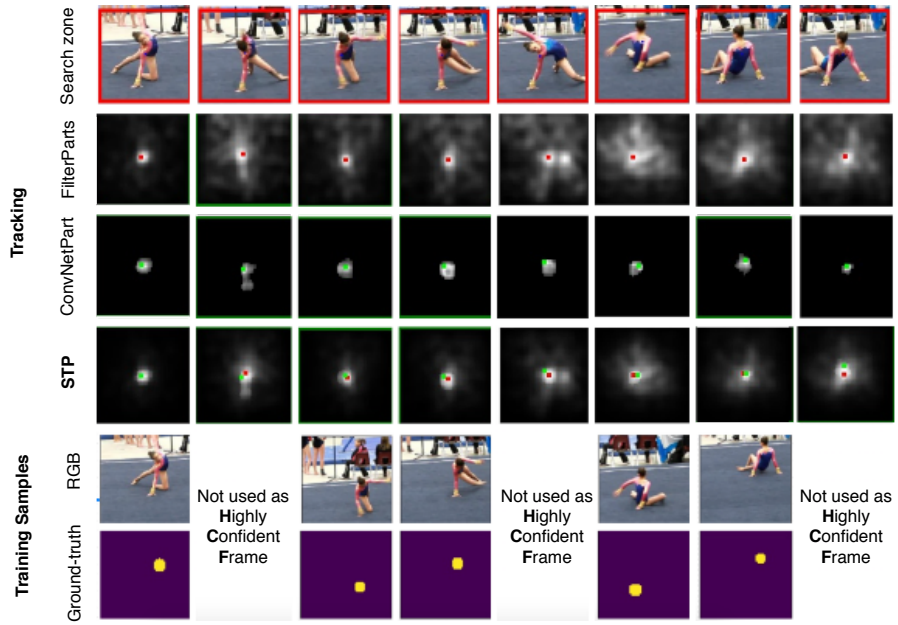

Figure 14: The voting maps for FilterParts, ConvNetPart and the final (STP), respectively. We also show the qualitative view of training samples selection for ConvNetPart. Frame is not Highly Confident if pathways votes centers are distanced. Best seen in colors.

3.5 EXPERIMENTAL ANALYSIS

RESULTS ON VOT17 AND VOT16 BENCHMARKS. We tested our tracker on the top visual object tracking benchmarks, VOT17 [80] and VOT16 [79]. VOT16 contains 60 video sequences, containing many difficult cases of occlusion, illumination change, motion change, size change and camera motion. The VOT17 dataset replaces sequences from VOT16 that were solved by most trackers with new and more difficult ones. For computing the final EAO evaluation score, VOT setup is re-initializing the tracker when it completely misses the target.

In Table 1 we present the results after running our tracker through the VOT toolkit. We compared our method against top published tracking methods: ECO [29], CCOT [32], CFWCR [62], Staple [8], ASMS [165], EBT [202], CCCT [19], CSRDCF [106], MCPF [197], ANT [18], some with reported results on both benchmarks. Our STP outperforms the current state of the art methods on VOT17, and is in the top three on VOT16. Note that we used the exact same set of parameters on all videos from both VOT17 and VOT16. What distinguishes our tracker the most from the rest is the much lower failure rate (0.76 vs. second best 1.13, on VOT17). We think this is due to the robustness gained by the use of co-occurrence constraints in all aspects of learning and inference, and the dual-pathway structure, with each pathway having complementary advantages.

Next we show how each design choice influenced the strong performance of our tracker.

COMBINING THE FILTERPARTS AND CONVNETPART PATHWAYS. In Table 2 we test the effect of combining the two pathways on the overall tracker. Each pathway is let by itself to guide the tracker. In the "FilterParts only" line, we have results where the first pathway becomes the tracker, with no influence from ConvNetPart ($\alpha = 1$). On the second we show the opposite case, when the tracker is influenced only by ConvNetPart ($\alpha = 0$). In that case the ConvNetPart is trained on the first 20 frames, then continuously updated on its own output, with no influence from the FilterParts pathway.

In general, the FilterParts pathway is more robust and resistant to drifting because it incorporates new information slower, after validating the candidates in time. It is also based on stronger pre-trained features on ImageNet [34]. It is more stable (lower failure rate) but less capable of learning about object appearance (lower accuracy, as IOU w.r.t ground-truth). The ConvNetPart pathway is

Tracker \ Dataset	VOT17 [80]			VOT16 [79]		
	EAO	R	A	EAO	R	A
STP (ours)	**0.309**	**0.765**	0.44	0.361	**0.47**	0.48
CFWCR [62]	0.303	1.2	0.48	**0.39**	0.81	**0.58**
ECO [29]	0.28	1.13	0.48	0.374	0.72	0.54
CCOT [32]	0.267	1.31	0.49	0.331	0.85	0.52
Staple [8]	0.169	2.5	**0.53**	0.295	1.35	0.54
ASMS [165]	0.169	2.23	0.494	0.212	1.925	0.503
CCCT [19]	-	-	-	0.223	1.83	0.442
EBT [202]	-	-	-	0.291	0.9	0.44
CSRDCF [106]	0.256	1.368	0.491	-	-	-
MCPF [197]	0.248	1.548	0.510	-	-	-
ANT [18]	0.168	2.16	0.464	-	-	-

Table 1: Top published trackers in terms of Expected Average Overlap (EAO), Robustness or Failure rate (R) and Accuracy (A), on VOT17 [80] and VOT16 [79] benchmarks. With red is marked the first result, blue is for the second and green for the third. Our tracker achieves state of the art performances on VOT17 [80] in terms of EAO and the 3^{rd} score on VOT16 [79]. Our failure rate is the best by large margins on both datasets (42% and 67%). Our overlap score is lower as we did not explicitly learn object shape or mask, but focus instead on its center. Note that we obtained the results with the exact same tracker and parameters for both VOT17 and VOT16. We will make our code available.

deeper and more powerful, but as it is continuously trained on its own tracker output it is prone to overfitting to background noise, resulting in many failures.

When using both components, the two pathways work in conjunction and learn from each other using their outputs' co-occurrence constraints. The deeper pathway (ConvNetPart) is learning from the less flexible but more robust pathway (FilterParts). The numbers confirm our intuition and show that the two paths work in complementary, each bringing important value to the final tracker. The boost in performance after combining them is truly significant.

USING DIFFERENT PART ROLES IN FILTERPARTS PATHWAY. We also tested with the case when all filters have one single role. Instead of considering

Version \ Dataset	VOT17			VOT16		
	EAO	R	A	EAO	R	A
FilterParts only	0.25	0.99	0.42	0.306	0.80	0.44
ConvNetPart only	0.205	2.09	0.43	0.265	1.53	0.46
Combined	**0.309**	**0.765**	**0.44**	**0.361**	**0.47**	**0.48**

Table 2: In "FilterParts only" experiment, the second pathway is not used at all. In "ConvNetPart only" experiment, we use the FilterParts pathway only for the first 20 frames, to initialize the network, and not use it afterwards. In the absence of high confidence frames selection, the ConvNetPart is trained on each frame, using its own predictions as ground-truth.

candidates, reliable and gold parts, which ensure stability over time, now all parts added over the sequence have the right to vote at any time. In Table 3 we see that the impact of multiple roles for filter parts, depending on their validation in time is high, bringing a 5% increase in terms of EAO, comparing to the basic one role for all version.

Version \ Dataset	VOT17			VOT16		
	EAO	R	A	EAO	R	A
One role	0.262	0.99	0.44	0.31	0.715	0.47
All roles	**0.309**	**0.765**	**0.44**	**0.361**	**0.47**	**0.48**

Table 3: Impact of different part roles used in FilterParts pathway. Considering roles based on parts credibility over time (candidate, reliable, gold), which is measured using spatial and temporal co-occurrences, is of great benefit to the tracker. It brings an advantage of 5% in EAO over the vanilla, "one role for all" case.

LEARNING WITH HIGHLY CONFIDENT FRAMES ON CONVNETPART PATHWAY. In order to better appreciate the value of HCFs in training the ConvNetPart, we have tested it against the cases of training on all frames (all frames are good for training) and that of training only on the first 20 frames (no frame is good, except for the first 20 when the ConvNetPart is initialized). As we can see in Table 4, the "Full continuous update" regime on all frames is worst or at most similar in performance with "No update" at all. This shows that the model can overfit very quickly, immediately resulting in drifting (high failure rate). The idea to learn only from Highly Confident Frames is of solid value, bringing

a 2% improvement in the final metric EAO, and a large cut off in failure rate. Even when we randomly select frames to be HCFs, of the same number as in the case of the true HCF measure, we again obtained the same drop of 2% in performance. These results, along with the statistical correlation between HCF and the ground-truth presented previously in Figure 13 validate experimentally the value of considering only a smaller set of high precision frames for training, even when that set might be just a small portion of all high quality frames.

Dataset / Version	VOT17			VOT16		
	EAO	R	A	EAO	R	A
No update	0.28	0.95	0.43	0.34	0.7	**0.48**
Full update	0.284	0.92	**0.44**	0.327	0.66	0.46
HCFs update	**0.309**	**0.765**	**0.44**	**0.361**	**0.47**	**0.48**

Table 4: Comparison in performance on VOT17 and VOT16, between updating the ConvNetPart only on Highly Confident Frames (HCF update), not updating it at all (No update), or updating it on every frame (Full update). We mention that in all our experiments we used the top 11% past frames, in confidence score, to perform training at a given time.

SPEED. The speed of FilterParts pathway is on average 30 fps on GTX TITAN X, for a maximum of 600 filter parts. The ConvNetPart comes with a significant time penalty only during learning updates. The tracker update is happening once on 10 frames, but the ConvNetPart is updating only when there are new HCFs in the queue. 11% of the frames are considered HCFs and they tend to be grouped near portions of the video where the confidence is very high. A full update takes 4 seconds and on average happens in 5% of the frames. So the STP average speed is of 4 fps on GPU.

3.6 CONCLUSION

We have proposed a deep neural network system for object tracking that functions as a society of tracking parts. Our tracker has two main deep pathways, one that is less flexible but more robust, and the second that is less robust

but more capable of adapting to complex changes in object appearance. Each part uses co-occurrences constraints in order to keep its robustness high over time, while allowing some degree of adaptability. The two pathways are also combined in a robust manner, by joining their vote maps and picking the locations where their votes co-occurred the most. From a technical point of view, the novelty aspects of our system include the way the classifiers in the FilterParts pathway are learned and ascribed different roles, depending on their degree of reliability. These roles relate the idea of a society, where some parts are candidates that are being monitored, others are reliable voters, where those who proved their reliability long enough become gold members. Another novelty aspect which brings an important value in practice, is the way we train the ConvNetPart on high confidence frames only, by selecting for training only those frames where the two different and complementary pathways, agree. We also provide a novel theoretical result, to the best of our knowledge, we proves that the efficient one sample vs. all strategy employed for learning the classifiers in the FilterParts path, is stable and gives basically the same result as in the balanced case. In experiments we provide solid validation of our design choices and show state of the art performance on VOT17 and top three on VOT16, while staying on top on both in terms of failure rate, by a significant margin.

Next in the thesis I will analyze how segmentation can enhance the performance of the tracking task. First we will approach only the video segmentation task in the next chapter, and next we will combine it with tracking to see how tracking can benefit from having a more refined intermediary representation, like the object segmentation mask.

4 SPACE-TIME SPECTRAL SEGMENTATION TOWARDS CONSENSUS

We pose video object segmentation as spectral graph clustering in space and time, with one graph node for each pixel and edges forming local space-time neighborhoods. We claim that the strongest cluster in the video graph represents the salient object. We compute this cluster using a novel and fast 3D filtering technique that finds the spectral solution as the principal eigenvector of the graph's adjacency matrix. We also show how to learn efficiently spectral clustering over multiple input feature channels. The method reduces to a set of specific 3D convolutions in the space-time feature channels volume, which is equivalent to computing the principal eigenvector without building the adjacency matrix explicitly. This key property allows us to have fast parallel implementation on GPU, orders of magnitude faster than classical approaches for computing the eigenvector. Our motivation for a spectral space-time clustering approach, unique in video semantic segmentation literature, is that such clustering is dedicated to preserving object consistency over time, which we evaluate using our novel segmentation consistency measure. In extensive experiments we show significant improvements over top methods, as well as over powerful ensembles that combine such methods, on the challenging DAVIS-2016 and SegTrackv2 datasets.

4.1 ZOOMING OUT: SEGMENTATION IN THE CONTEXT OF THE THESIS

We noticed in the previous chapter on tracking the quantity of noise that step by step leads the tracker into drifting. Thus we emphasize the need of having a more refined representation, like the segmentation of the object. So, in this chapter, we approach the object segmentation task in video. More, we

E. Burceanu, M. Leordeanu, A 3D Convolutional Approach to Spectral Object Segmentation in Space and Time, The International Joint Conferences on Artificial Intelligence (IJCAI) 2020

take advantage here of having the object contour as input and refine it based on space and time natural consistencies discovered through clustering over the video volume of pixels. We analyze next this chapter, following the key aspects introduced at the beginning of the thesis:

- **A. Space-time consistency**: This approach formulates the segmentation problem in video as clustering in the video volume, where neighbours are both spatial and temporal. Our spectral solution over this 3D volume naturally exploits the connections between dimensions.
- **B. The power of the consensus**: In the first part of this work, the consensus can be seen at the level of neighbourhoods, where each pixel is iteratively updated using its space-time neighbours. The second part combines many top segmentation methods, each as a different channel in our iterative spectral approach.
- **C. Exploiting multiple intermediate representations**: -
- **D. A limited quantity of supervision**: The basic SFSeg algorithm is purely unsupervised. In the second part, we use supervision to learn how to combine multiple channels in the input.
- **E. Experts: Making use of existing models**: We use expert models as input in SFSeg++, our extended algorithm, to better analyze the potential increase in performance of our approach when compared with other ensemble methods.

4.2 CONTEXT

Elements from a video are interconnected in space and time and have an intrinsic graph structure (Fig. 15). Most existing approaches use higher-level components, such as objects, super-pixels or features, at a significantly lower resolution. Considering the graph structure in space and time, explicitly at the dense pixel-level, is an extremely expensive problem.

Our proposed solution to video object segmentation is based on transforming an expensive eigenvalue problem inspired by spectral clustering, into 3D convolutions on the space-time volume, which makes our algorithm fast, while keeping the properties of spectral clustering. We first refine the output of a single existing method (**SFSeg**), next we show how to learn the space-time clustering in the context of multiple feature input channels (**SFSeg++**). This allows

learning of powerful ensembles in combination with our spectral clustering approach, to obtain state-of-the-art performance on challenging datasets by a significant margin.

Most state-of-the-art algorithms for this task do not use the time constraint, and when they do, they take little advantage of it. Time plays a fundamental factor in how objects move and change in the world, but computer vision does not yet exploit it sufficiently. Consequently, the segmentation outputs of current state-of-the-art methods is not always consistent over time. Our work comes to address precisely this aspect and our contribution is demonstrated through solid experiments on DAVIS-2016 and SegTrackv2 datasets, on which we improve over such state-of-the-art solutions.

We first demonstrate theoretically and through extensive experiments that the eigenvector of the space-time graph's adjacency matrix is a good solution for salient object segmentation and can be computed fast with a special set of 3D convolutions. We prove theoretically and in practice that SFSeg reaches the same solution as standard routines for eigenvector computation. We also show in experiments that the values in the final eigenvector, with one element per video pixel, confirm the spectral clustering assumption and provide an improved soft-segmentation of the main object.

Next, we present the full SFSeg++, an augmented version of SFSeg with learning capability. We show how to learn to combine multiple input channels in conjunction with the 3D spectral filtering, to significantly improve over the initial SFSeg version and also over different types of ensembles that combine the output of existing state-of-the-art methods.

One of the key properties of our filtering-based optimization is that it converges to the eigenvector of the graph of pixels in space and time. The segmentation map obtained is spatio-temporally consistent, with a smooth and coherent transition between frames. That is the main reason why the proposed method is best suited for improving or refining the output of any other method, since most existing algorithms do not take full advantage of the space-time consistency. Therefore, when our method is applied on top of the output of another, the noise coming from other objects is removed and missing parts of the object are added back. Through multiple iterations, the relevant information is propagated step by step to farther away neighbourhoods in space and time, acting like a diffusion. As experiments show, the improvement brought by our algorithm is consistent and reliable - it takes place almost every time and with the same set of hyper-parameters.

The main contributions of this work are the following:

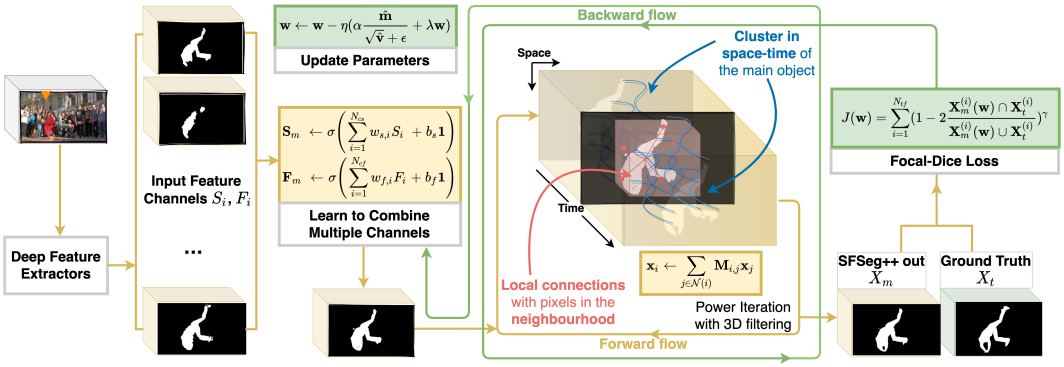

Figure 15: SFSeg++: Our method learns to combine multiple input feature channels (*e.g.* learnable ensemble of other methods' outputs, as separate channels) and compute the principle eigenvector of the space-time graph with a special set of 3D convolutions very fast. The combination weights for the different channels are learned with gradient descent, by propagating the gradient of the loss through the full spectral algorithm. In experiments we start from the output of top published segmentation methods and use their combined output volume, in space and time, as graph nodes for our spectral approach, which then computes the final segmentation frame by frame. Please note that this method is general and could learn to combine any type of input feature channels, as tests show. **Also note that SFSeg++ permits full end-to-end learning**, in the case when the deep networks and systems that provide the input feature channels are fully differentiable.

1. We formulate the instance segmentation problem in video as an eigenvalue problem on the adjacency matrix of the pixels' graph in space and time.

2. We provide a fast optimization algorithm with a special set of 3D filtering operations, which computes the required eigenvector (representing the desired segmentation) without explicitly creating or using the huge graph's adjacency matrix, in which each pixel in space and time has an associated node.

3. Our SFSeg++ algorithm becomes fully learnable over multiple input channels. In experiments we show that SFSeg++, learning over all segmentation methods available for training on DAVIS 2016, brings in complementary information and significantly improves over the input. SFSeg++

outperforms on the current single best state-of-the art by 4.2% and the ensemble of top 15 published methods by 3.1%.

4. We introduce TCONT, a novel temporal consistency metric for segmentation in video, which looks at the temporal continuity of the predictions in video and their stability with respect to the motion field of the object (as estimated by optical flow). The experiments show that our SFSeg++ algorithm is significantly more temporally consistent with respect to ground-truth, when compared to single or ensemble state-of-the-art approaches.

4.3 RELATION TO PRIOR WORK

Most state-of-the-art methods for video object segmentation use **CNNs architectures**, pre-trained for object segmentation on other large image datasets. They have a strong image-based backbone and are not designed from scratch with both space and time dimensions in mind. Many solutions [76, 157, 22] adapt image segmentation methods by adding an additional branch to the architecture for incorporating the time axis: motion branch (previous frames or optical flow) or previous masks branch (for mask propagation). Other methods are based on one-shot learning strategies and fine-tune the model on the first video frame, followed by some post-processing refinement [163, 109]. Approaches derived from OSVOS [110] do not take the time axis into account.

IMPROVING ALONG THE TEMPORAL DIMENSION. Our method comes to better address the natural space-time relationship, which is why it is always effective when combined with frame-based segmentation algorithms. The need for video consistency was previously identified and approached. A temporal stability metric was proposed in DAVIS-2016 [130] but in less than a year it was withdrawn [133], since the metric was hardly influenced by occlusion, leading to irrelevant comparisons. Temporal stability was based on the Dynamic Time Warping problem, looking to match, between two frames, pixels from the contour of the object, such that the distance between Shape Context Descriptor from the two shapes is minimized. In contrast, our TCONT metric for temporal consistency is based on checking the alignment of nearby predictions using direct and reverse optical flow, and it is not affected by occlusions. Experiments in Sec. 4.6.4 prove that our approach brings a complementary value along the time axis, in solving object segmentation in video while improving the temporal consistency of the result.

RELATION TO DIFFERENT GRAPH REPRESENTATIONS. Graph methods are suitable for segmentation and can have different representations, where the **nodes** can be pixels, super-pixels, voxels, image or video regions [70]. While there are works on directed graphs [160, 189], edges are usually undirected, modeled by symmetric similarity functions. The choice of the representation influences both accuracy and runtime. Specifically, pixel-level representations are computationally extremely expensive, making the problem intractable for high resolution videos. Our fast solution implicitly uses a pixel-level graph representation: we make a first-order Taylor approximation of the Gaussian kernel (usually used for pairwise affinities) and rewrite it as a sequence of 3D convolutions in the video directly. Thus, we get the desired outcome without explicitly working with the graph. We describe the technical novelties in detail in Sec. 4.4.

RELATION TO SPECTRAL CLUSTERING. Computing eigenvectors of matrices extracted from data is a classic approach for clustering. There are several choices in the literature for choosing those matrices, the most popular being the Laplacian matrix [125], normalized [141] or unnormalized. Other methods use the random walk matrix [113, 70] or directly the unnormalized adjacency matrix [90]. Most methods are based on finding the eigenvectors corresponding to the smallest eigenvalues, while others, including ours, require the leading eigenvectors. **Graph Cuts** are a popular class of spectral clustering algorithms, with many variants, like normalized [141], average [140], min-max [35], mean cut [174] and topological cut [192].

RELATION TO CRFS. Discriminative graphical models such Conditional Random Fields [86] and Discriminative Random Fields [83] are often applied over the segmentation of images and videos (denseCRF [82]). Different from the more classical Markov Random Fields (MRFs) [178], which are generative models, CRFs are more effective as they incorporate the observed data both at the level of nodes as well as edges. Different from our approach, CRFs have a strict probabilistic interpretation and use inference algorithms (*e.g.* belief propagation, iterative conditional modes, Gibbs sampling, graph-cut) that are significantly more expensive than the simpler eigenvector power iteration that we use for optimizing our non-probabilistic objective score. In experiments we compare and also combine our method with denseCRF [82] and show that the two bring complementary value to the final solution.

RELATION TO IMAGE SEGMENTATION. Graph cuts, which represent a well-known class of spectral clustering algorithms, have been widely used in image segmentation [180, 141]. They are expensive in practice, as they require the computation of eigenvectors of smallest eigenvalues for very large Laplacian matrices. Fast graph-based algorithm for image segmentation exist, such as [44], which is linear in the number of edges and it is based on an heuristic for building the minimum spanning tree. It is still used as staring point by current methods. Another approach [134] is to learn image regions with spectral graph partitioning and formulate segmentation as a convex optimization problem.

RELATION TO VIDEO SEGMENTATION. Many video segmentation methods adapt existing image segmentation approaches. In [188] a parametric graph partitioning model over superpixels is proposed. Hierarchical graph-based segmentation over RGBD video sequences [65] also groups pixels into regions. The problem is solved using bipartite graph matching and minimizing the spanning tree. In [96], an efficient graph cut method is applied on a subset of pixels. To our best knowledge, all of the efficient methods group pixels into superpixels, regions from a grid or object proposals [41, 131, 111, 37] to handle the computational and memory burden. However, the hard initial grouping of pixels comes with a risk and could carry errors into the final solution, as it misses details available only at the original pixel resolution.

OUR WORK is most related to [90, 113]. The solution is the leading eigenvector of the adjacency matrix, computed fast and stable with power iteration as explained in Sec. 4.4. Note that using the unnormalized adjacency matrix in combination with power iteration is the least expensive spectral approach possible and the only one that can be factored into simple and fast 3D convolutions. This possibility gives our algorithm efficiency and speed (Sec. 4.5).

4.4 OUR APPROACH

We formulate salient object segmentation in video as a graph partitioning problem (foreground vs background), where the graph is both spatial and temporal. Each node i represents a pixel in the space-time volume, which has $N = N_f \times H \times W$ pixels. N_f is the number of frames and (H, W) the frame size. Each edge captures the similarity between two pixels and is defined by the pairwise function $\mathbf{M}_{i,j}$. We require the pairwise connections between pix-

els i and j, in space and time, to be symmetric and non-negative, defining a $N \times N$ adjacency matrix $\mathbf{M}$. We take into account only the local connections in space-time, so $\mathbf{M}$ is sparse.

Let $\mathbf{s}$ and $\mathbf{f}$ be feature vectors of size $N \times 1$ with a feature value for each node. They will be used in defining the similarity function $\mathbf{M}_{ij}$ (Eq. 16). For now we consider the simplest case when $(\mathbf{s}_i, \mathbf{f}_i)$ represent single-channel features (*e.g.* they could be soft masks, grey level values, edge or motion cues, or any pre-trained features). Later on we show how we can easily adapt the formulation to the multi-channel feature case. We define the edge similarity $\mathbf{M_{i,j}}$ using a Gaussian kernel:

$$\begin{aligned} \mathbf{M}_{i,j} &= \mathbf{s}_i^p \mathbf{s}_j^p e^{-\alpha(\mathbf{f}_i - \mathbf{f}_j)^2 - \beta \mathbf{dist}_{i,j}^2} \\ &= \mathbf{s}_i^p \mathbf{s}_j^p e^{-\alpha(\mathbf{f}_i - \mathbf{f}_j)^2} \mathbf{G}_{i,j} \end{aligned} \tag{16}$$

$$\mathbf{M}_{i,j} \approx \underbrace{\mathbf{s}_i^p \mathbf{s}_j^p}_{\text{unary terms}} \quad \underbrace{[1 - \alpha(\mathbf{f}_i - \mathbf{f}_j)^2]\mathbf{G}_{i,j}}_{\text{pairwise terms}}. \tag{17}$$

In graph methods, it is common to use two types of terms for representing the model over the graph. Unary terms are about individual node properties, while pairwise terms describe relations between pairs of nodes. In our case, $\mathbf{s}_i$, $\mathbf{s}_j$ describe individual node properties, whereas $\mathbf{f}_i$, $\mathbf{f}_j$ are used to define the pairwise similarity kernel between the two nodes. Note that in Eq. 17 we approximate the Gaussian kernel with its first-order Taylor expansion. The approximation is crucial in making the filtering approach possible, as shown next. Hyper-parameters p and α control the importance of those terms.

To partition the space-time graph of video pixels, we want to find the strongest cluster in this graph. We first represent a segmentation solution (i.e., cluster in the space-time graph) with an indicator vector $\mathbf{x}$, that has one element for each node in the 3D space-time volume, such that $x_i = 1$ if node (pixel) i is in the video segmentation cluster (foreground) and $x_i = 0$ otherwise (background). We define the clustering score to be the sum over all pairwise similarity terms $\mathbf{M}_{ij}$ between the nodes inside the cluster. The higher this score, the stronger the sum of connections and the cluster. The segmentation score can be written compactly in matrix form as $S(\mathbf{x}) = \mathbf{x}^\top \mathbf{M}\mathbf{x}$. Similar to other spectral approaches in graph matching [90], we find the segmentation solution $\mathbf{x}_s$ that maximizes $S(\mathbf{x})$ under the relaxed constraints $\|\mathbf{x}\|_2 = 1$. Fixing the L2 norm of $\mathbf{x}$ is needed since only relative soft segmentation values matter. Thus, the optimization problem become one of maximizing the Raleigh quotient:

$$\mathbf{x}_s = \underset{\mathbf{x}}{\operatorname{argmax}}(\mathbf{x}^\top \mathbf{M}\mathbf{x}/\mathbf{x}^\top \mathbf{x}). \tag{18}$$

The global optimum solution is the principal eigenvector of $\mathbf{M}$. $\mathbf{M}$ is symmetric and has non-negative values, so the solution will also have non-negative elements, by Perron-Frobenius theorem [48]. The final segmentation could be simply obtained by thresholding. However, matrix $\mathbf{M}$, even for a small video has 20 million nodes (50 frames of 480×854), making the problem of finding the leading eigenvector with standard procedures intractable (Sec 4.5.2).

Next we show how to take advantage of the first-order expansion of the pairwise terms defining $\mathbf{M}$ and break power iteration into several very fast 3D convolutions in space and time, directly on the feature maps, without explicitly using the huge adjacency matrix of the video. Our method receives as input pixel level feature maps and returns a final segmentation, as the solution $\mathbf{x}_s$ to Eq. 18.

4.4.1 Power iteration with pixel-wise iterations

We apply power iteration algorithm to compute the eigenvector. At iteration $k+1$, we have Eq. 19:

$$\mathbf{x}_i^{k+1} \leftarrow \sum_{j \in \mathcal{N}(i)} \mathbf{M}_{i,j} \mathbf{x}_j^k, \tag{19}$$

where, after each iteration, the solution is normalized to unit norm and $\mathcal{N}(i)$ is the set of neighbors pixels with i, in space and time. Expanding $\mathbf{M}_{i,j}$ (Eq. 17), Eq. 19 becomes:

$$\mathbf{x}_i^{k+1} \leftarrow \alpha \mathbf{s}_i^p \sum_{j \in \mathcal{N}(i)} \mathbf{s}_j^p [\alpha^{-1} - \mathbf{f}_i^2 - \mathbf{f}_j^2 + 2\mathbf{f}_i \mathbf{f}_j] \mathbf{G}_{i,j} \mathbf{x}_j^k, \tag{20}$$

$$\begin{aligned} \mathbf{x}_i^{k+1} \leftarrow \alpha \mathbf{s}_i^p (\alpha^{-1} - \mathbf{f}_i^2) \sum_{j \in \mathcal{N}(i)} \mathbf{s}_j^p \mathbf{G}_{i,j} \mathbf{x}_j^k - \\ \alpha \mathbf{s}_i^p \sum_{j \in \mathcal{N}(i)} \mathbf{s}_j^p \mathbf{f}_j^2 \mathbf{G}_{i,j} \mathbf{x}_j^k \\ 2\alpha \mathbf{s}_i^p \mathbf{f}_i \sum_{j \in \mathcal{N}(i)} \mathbf{s}_j^p \mathbf{f}_j \mathbf{G}_{i,j} \mathbf{x}_j^k. \end{aligned} \tag{21}$$

4.4.2 Power iteration using 3D convolutions

In Eq. 21 we observe that the links between the nodes are local ($\mathbf{M}$ is sparse) and we can replace the sums over neighbours with local 3D convolutions in space and time. Thus, we rewrite Eq. 21 as a sum of convolutions in 3D:

$$\begin{aligned}\mathbf{X}^{(crt)} \leftarrow \mathbf{S}^p \cdot (\alpha^{-1}\mathbf{1} - \mathbf{F}^2) \cdot G_{3D} * (\mathbf{S}^p \cdot \mathbf{X}^k) - \\ \mathbf{S}^p \cdot G_{3D} * (\mathbf{F}^2 \cdot \mathbf{S}^p \cdot \mathbf{X}^k) + \\ 2\mathbf{S}^p \cdot \mathbf{F} \cdot G_{3D} * (\mathbf{F} \cdot \mathbf{S}^p \cdot \mathbf{X}^k),\end{aligned} \tag{22}$$

$$\mathbf{X}^{k+1} \leftarrow \mathbf{X}^{(crt)} / \|\mathbf{X}^{(crt)}\|_2, \tag{23}$$

where $*$ is a convolution over a 3D space-time volume with a 3D Gaussian filter (G_{3D}), $\cdot$ is an element-wise multiplication, 3D matrices $\mathbf{X}^k, \mathbf{S}, \mathbf{F}$ have the original video shape ($N_f \times H \times W$) and $\mathbf{1}$ is a 3D matrix with all values 1. We transformed the standard form of power iteration in Eq. 19 in several very fast matrix operations: 3 convolutions and 13 element-wise matrix operations (multiplications and additions), which are local operations that can be parallelized.

4.4.3 Multi-channel SFSeg++

Our approach in Eq. 22 can easily accommodate multiple feature channels. We extend the initial SFSeg formulation such that it can learn how to combine the input feature maps for improving its final result into multi-channel versions of the unary and pairwise maps: $\mathbf{S}_m$ and $\mathbf{F}_m$, respectively, as described in the next Eq. 24.

$$\begin{aligned}\mathbf{S}_m \leftarrow \sigma(\sum_{i=1}^{N_{cs}} w_{s,i}\mathbf{S}_i + b_s\mathbf{1}), \\ \mathbf{F}_m \leftarrow \sigma(\sum_{i=1}^{N_{cf}} w_{f,i}\mathbf{F}_i + b_f\mathbf{1}),\end{aligned} \tag{24}$$

where σ is the sigmoid function $\sigma(x) = 1/(1+e^x)$, N_{cs} and N_{cf} are the number of unary and pairwise input feature channels, respectively, $\mathbf{1}$ is an all-one matrix for the bias terms and $w_{s,i}, w_{f,i}, b_s, b_f$ are their corresponding learned weights.

The final power iteration algorithm with 3D filtering for the multi-channel case is described in the next equations: Eq. 25 and Eq. 26, which follow immediately from the single-channel Eq. 22 and Eq. 23:

$$\mathbf{X}_m^{(crt)} \leftarrow \mathbf{S}_m^p \cdot (\alpha^{-1}\mathbf{1} - \mathbf{F}_m^2) \cdot G_{3D} * (\mathbf{S}_m^p \cdot \mathbf{X}_m^k) - \mathbf{S}_m^p \cdot G_{3D} * (\mathbf{F}_m^2 \cdot \mathbf{S}_m^p \cdot \mathbf{X}_m^k) + 2\mathbf{S}_m^p \cdot \mathbf{F}_m \cdot G_{3D} * (\mathbf{F}_m \cdot \mathbf{S}_m^p \cdot \mathbf{X}_m^k), \tag{25}$$

$$\mathbf{X}_m^{k+1} \leftarrow \mathbf{X}_m^{(crt)} / \|\mathbf{X}_m^{(crt)}\|_2. \tag{26}$$

LEARNING OVER MULTI-CHANNEL SFSEG++ We learn $w_{s,i}, w_{f,i}, b_s, b_f$ parameters by minimizing the Focal-Dice loss [170] over the training video frames, proved to be suitable for segmentation tasks (Eq. 27):

$$J(\mathbf{w}_t) = \sum_{i=1}^{N_{tf}} (1 - 2\frac{\mathbf{X}_m^{(i)}(\mathbf{w}_t) \cap \mathbf{X}_t^{(i)}}{\mathbf{X}_m^{(i)}(\mathbf{w}_t) \cup \mathbf{X}_t^{(i)}})^\gamma, \tag{27}$$

where N_{tf} is the number of training frames and X_t is the ground-truth video segmentation.

We optimize the loss using a state-of-the-art gradient descent based optimizer (AdamW [25]):

$$\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} - \eta_t(\alpha \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon} + \lambda \mathbf{w}_{t-1}), \tag{28}$$

where η_t is the scaling factor and $\nabla_{\mathbf{w}} J(\mathbf{w}_{t-1})$ is used for updating first and second momentum vectors: $\hat{\mathbf{m}}_t$ and $\hat{\mathbf{v}}_t$ respectively, at optimization step t.

We make SFSeg++ trainable by using a soft-binarization between iterations. Consequently, it can learn end-to-end, from the original input frames all the way to final output, in the case when the input feature channels are also learnable end-to-end.

4.4.4 TCONT: Temporal consistency metric

We introduce TCONT, a new metric that measures the consistency in time of a video segmentation input. Different from other metrics in semantic segmentation, this is specifically thought for video segmentation and focuses on the key distinction between a video seen as a whole and a video seen as just a set of frames. We first observe that a certain segmentation method could perform well at the level of individual frames, without being able to provide the desired consistency and coherency of the object shape over time. This aspect is important for video object segmentation methods and should constitute

a complementary dimension for measuring performance, different from the standard, per frame, IoU. Ideally, one would want a segmentation procedure to be both accurate per frame as well as consistent over time. As we see in experiments, while the two aspects are correlated, they are not the same.

Here we introduce the TCONT metric, which rebuilds each frame in the segmentation video by taking into account its shape continuity from the left and right frames, via optical flow ($\overrightarrow{\mathbf{OF}_{\text{dir}}}, \overleftarrow{\mathbf{OF}_{\text{rev}}}$). Ideally, if we warp the object masks from the left and right frames into the current frame, using optical flow, we would want to obtain the same shape, also identical to the current shape in the middle. In order to evaluate this desired property, we transform the segmentations of the previous and next frames into the current one, using optical flow, and then add all three segmentations, past, present and future, with equal proportions. This ensemble of three segmentations, which produce a combined mask, as the average of the three can now be evaluated on its own, with respect to ground-truth or with respect to the initial current, per frame segmentation.

Intuitively, a large TCONT agreement score means that the frames are contiguous and movement-consistent, such that the segmentation correctly changes over time according to the movements of the object in the video, as expressed by optical flow (without wrongly having large parts of the segmentation flickering in noisy ways, in time). If such time consistency exists and the initial masks are close to ground-truth in similar ways, it is expected that the average mask of the three will maintain its high IoU measure with respect to ground-truth. However, if the segmentations at the three moments in time are not consistent temporally with respect to the object motion field, as estimated by the optical flow, then we could expect the average of the three to produce a degraded, blurry, less accurate mask.

Mathematically we define the TCONT metric as follows:

$$\mathbf{x}_{\text{tcont}} = \frac{1}{3}(\overrightarrow{\mathbf{OF}_{\text{dir}}}(\mathbf{x}_{1:n-1}) + \mathbf{x}_{2:n-1} + \overleftarrow{\mathbf{OF}_{\text{rev}}}(\mathbf{x}_{2:n})) \tag{29}$$

$$\text{TCONT}_{\text{gt}} = \text{IoU}(\mathbf{x}_{\text{tcont}}, \mathbf{gt}_{2:n-1}), \tag{30}$$

where $\mathbf{x}$ is the soft video segmentation, $\mathbf{gt}$ is the ground-truth segmentation to which we relate, n is the number of frames, IoU is the IoU metric computed frame by frame, using a certain threshold, and $\overrightarrow{\mathbf{OF}_{\text{dir}}}, \overleftarrow{\mathbf{OF}_{\text{rev}}}$ represents the direct and reverse optical flow respectively. In Sec. 4.6.4 we validate experimentally that SFSeg++ algorithm significantly improves the temporal consistency of the initial input segmentation, while also improving the overall IoU

performance. This result is confirmed in most of the experimental setups in Sec. 4.6.

4.5 ALGORITHM

Next we present our algorithm, which applies to both single or multiple input channels. SFSeg++ (Alg. 4) starts by combining the input channels $\mathbf{S}_i$, $\mathbf{F}_i$, which could be of any kind: lower-level (optical flow, edges, gray-level values) or higher-level pre-trained semantic features (deep features or initial soft/hard segmentation maps) into the new feature maps $\mathbf{S}_m$ and $\mathbf{F}_m$. In the multi-channel SFSeg++ experiments we use as input channels the segmentation output of 15 top methods in the literature.

Next we initialize the solution $\mathbf{X}_m$ either with an uniform vector or with the segmentation provided by $\mathbf{S}_m$. At each iteration we first select a time frame around the current one. In Step 2 we multiply the corresponding matrices, apply the convolutions, compose the results and obtain the new segmentation mask for pixels in current frame, using the space-time operations (as in Eq. 22). At evaluation time, the solution needs to be binary, so after each iteration, we project the solution in a more discrete space (see Sec. 4.5.1).

4.5.1 Soft-binarization: from continuous to discrete

The spectral solution over the power iterations is continuous. However, at the very end, we need a binary, hard segmentation map for the object of interest. While the relaxed, continuous solution is globally optimal w.r.t Eq. 18, there is no guarantee that a simple, binary thresholding of the final continuous solution will retain optimality. In fact, as previously observed in the graph matching literature such optimality is often lost by thresholding, so keeping the continuous solution as close as possible to the initial discrete domain comes with a better final performance [91], even though the global optimum in the spectral space is affected. Therefore, we took a similar approach, and after the continuous power iteration is over, we continue with a set of iterations after each of which the solution is soft-binarized, with a sigmoid function (controlled by a parameter) that gradually brings the solution to the discrete domain. Thus, after each soft-binariaztion iteration, the solution is projected onto an almost discrete space. So, after the very last iteration, we apply a hard

Algorithm 4 SFSeg++: We linearly combine input channels into one single-channel, using a set of weights (that we learn), then pass it through a pixel-wise sigmoid function. At each iteration we pass through the whole video and compute the updated soft-segmentation $\mathbf{X}_m$. In the first step, we select a time window around current frame $[i-t, i+t]$ ($t = 6$ in experiments). Next we compute the eigenvector using the proposed 3D convolutions. At the end of the continuous power iteration (for N_{cont} iterations), we start soft-binarizing the solution at the end of each iteration until all N_{iter} iterations are completed (see Sec. 4.5.1 for details).

$\mathbf{S}_i, \mathbf{F}_i$ - **Input**: unary and pairwise video feature maps
N_f - **Input**: number of video frames
N_{iter} - **Input**: total number of iterations
N_{cont} - **Input**: number of continuous space iterations
$\mathbf{X}_m$ - **Output**: salient object segmentation in video

▷ **Initialization.** Note that for single-channel case, we directly assign maps to $\mathbf{S}_m$ and $\mathbf{F}_m$, without the sigmoid:

$\mathbf{S}_m \leftarrow \sigma(\sum_{i=1}^{N_{cs}} w_{s,i}\mathbf{S}_i + b_s\mathbf{1})$
$\mathbf{F}_m \leftarrow \sigma(\sum_{i=1}^{N_{cf}} w_{f,i}\mathbf{F}_i + b_f\mathbf{1})$
$\mathbf{X}_m \leftarrow \mathbf{S}_m$

for iter **in** $[1..N_{iter}]$ **do**
  for i **in** $[(t+1)..(N_f - t)]$ **do**
    ▷ **STEP 1:** Temporal window around frame i:
      $\mathbf{X}_{(t)}, \mathbf{S}_{(t)}, \mathbf{F}_{(t)} \leftarrow T_{i,t}(\mathbf{X}_m, \mathbf{S}_m, \mathbf{F}_m)[i-t : i+t]$

    ▷ **STEP 2:** Compute new segmentation:
      $\mathbf{T1} \leftarrow (\alpha^{-1}\mathbf{1} - \mathbf{F}^2_{(t)}) \cdot G_{3D} * (\mathbf{S}^p_{(t)} \cdot \mathbf{X}_{(t)})$
      $\mathbf{T2} \leftarrow -G_{3D} * (\mathbf{F}^2_{(t)} \cdot \mathbf{S}^p_{(t)} \cdot \mathbf{X}_{(t)})$
      $\mathbf{T3} \leftarrow 2\mathbf{F}_{(t)} \cdot G_{3D} * (\mathbf{F}_{(t)} \cdot \mathbf{S}^p_{(t)} \cdot \mathbf{X}_{(t)})$
      $\mathbf{X}_{new}[i] \leftarrow \mathbf{S}^p_{(t)} \cdot (\mathbf{T1} + \mathbf{T2} + \mathbf{T3})$
  $\mathbf{X}_m \leftarrow \text{normalize}(\mathbf{X}_{new})$

  ▷ **STEP 3:** Soft-binarization (using a sigmoid):
  if iter $> N_{cont}$ **then**
    $\mathbf{X}_m \leftarrow \sigma(\mathbf{X}_m)$

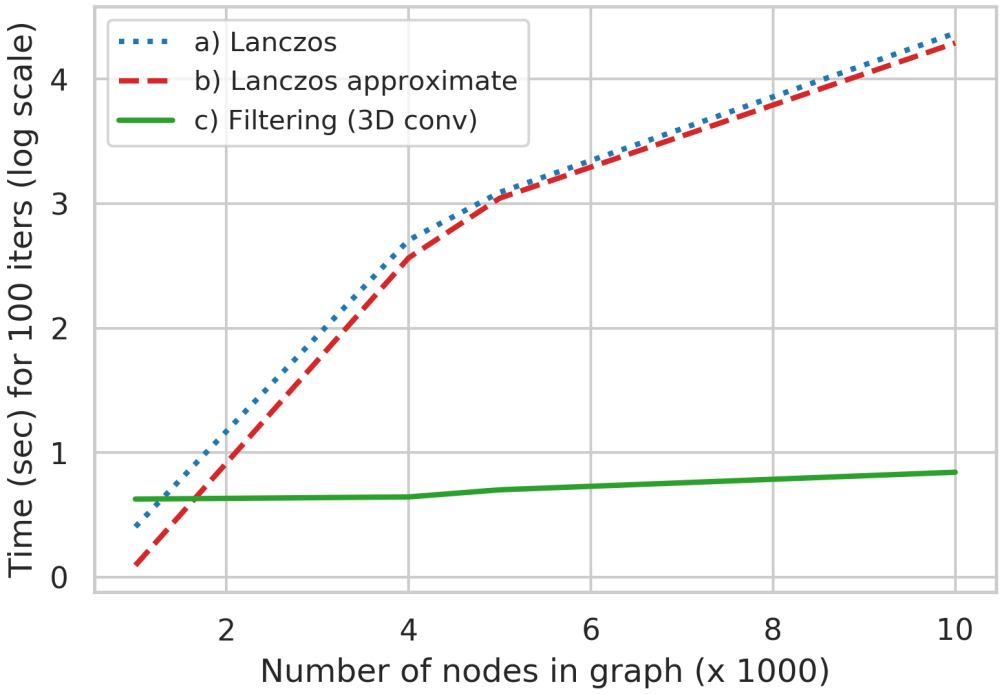

Figure 16: Total runtime in logarithmic scale for 100 iterations, including the time for building the adjacency matrix for power iteration. The filtering formulation scales with the number of nodes, in contrast to power iteration, having an exponentially better time.

threshold on a solution that is much closer to the desired discrete space than the continuous spectral solution.

4.5.2 Computational complexity

We compare the standard power iteration eigenvector computation with our filtering formulation, both from qualitative and quantitative points of view. In terms of the quantitative comparative analysis we will look at both accuracy (how close the filtering is to the exact solution) and speedup (how fast the 3D filtering approach is to the classic Lanczos method for computing the principal eigenvector of a matrix).

Lanczos [87] method for sparse matrices has $\mathcal{O}(kN_fN_pN_i)$ complexity for computing the leading eigenvector, where k is the number of neighbours for each node, N_f the number of frames in video, N_p the number of pixels per frame and N_i the number of iterations. Our full iteration algorithm has also

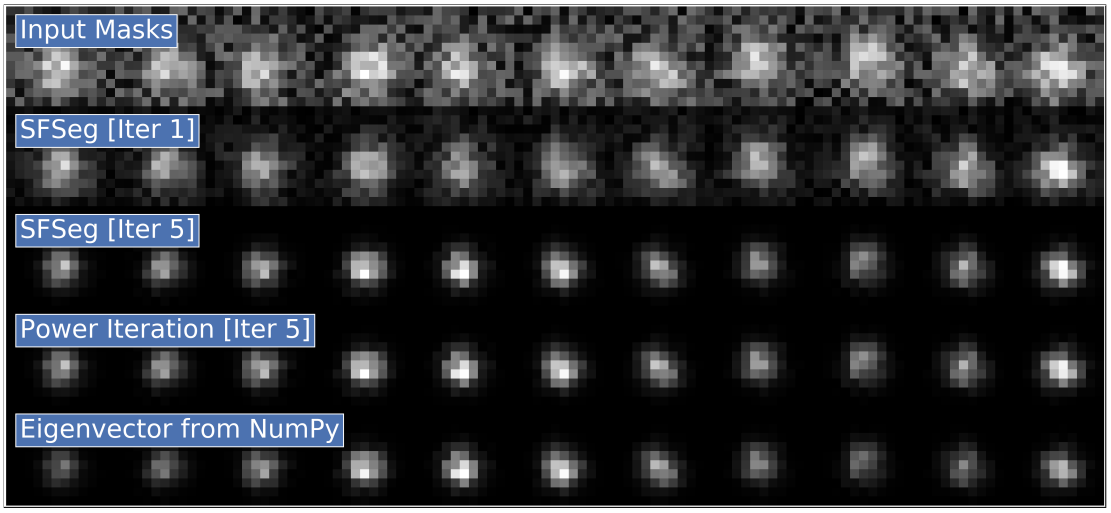

Figure 17: A toy example for qualitative comparisons with soft masks for an eleven frames video. Starting with a very noisy input segmentation mask and showing: SFSeg segmentations after 1 and 5 iterations; Power Iteration after 5 iterations; the real principal eigenvector. The results are almost identical, proving that SFSeg is a good approximation. More, for all other methods we compare with, this is tractable only on toy examples.

$\mathcal{O}(kN_fN_pN_i)$ complexity, but with highly parallelizable operations, comparing to Lanczos. The Gaussian filters are separable, so the 3D convolutions can be broken into a sequence of three vector-wise convolutions, reducing the complexity $\mathcal{O}(k)$ for filtering to $3\mathcal{O}(k^{\frac{1}{3}})$: 3*7*7=147 vs 3+7+7=17 for a 3x7x7 kernel.

We compare the actual runtimes of three solutions: **a)** Lanczos for computing the principal eigenvector of the adjacency matrix built using the initial Eq. 16 **b)** Lanczos for computing the principal eigenvector of the approximate adjacency matrix based on Eq. 17 **c)** our 3D convolutions approach. For a small graph of 4000 nodes (a video with 10 frames of 20×20 pixels), **a)** and **b)** have 0.15 sec/iter and **c)** our 3D filtering formulation has 0.02 sec/iter (Fig. 16), which is almost an order of magnitude faster. As the number of nodes grows our 3D filtering approach quickly becomes many orders of magnitude faster. The proposed method scales better and has a huge advantage when working with videos with millions of nodes. As stated before, the key reasons for the large speedup is that we do not explicitly build the adjacency matrix and filtering is parallelized on GPU.

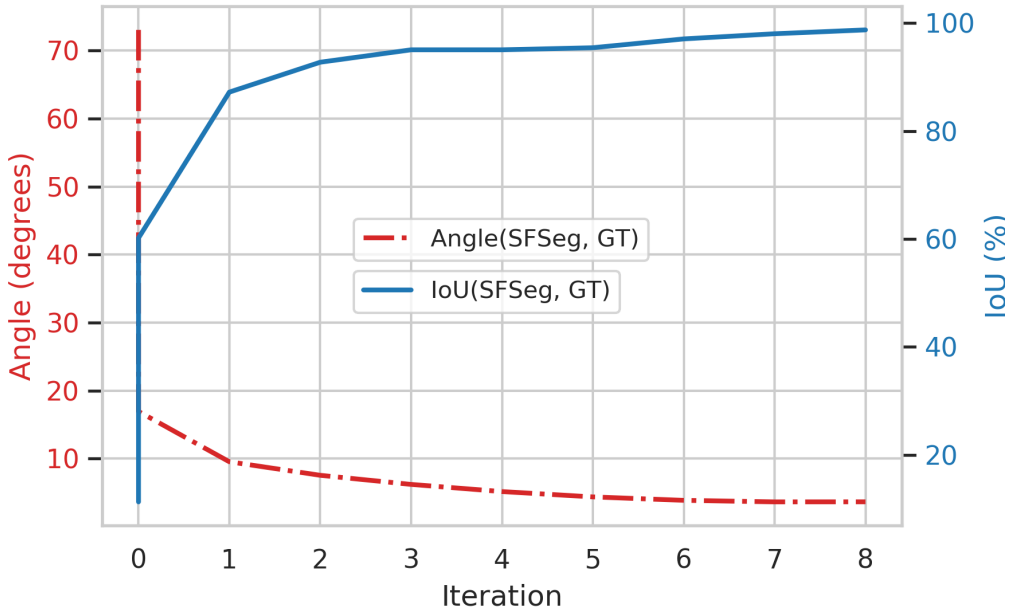

Figure 18: The angle and the IoU between the exact eigenvector (computer with Lanczos method) and our SFSeg solution. The evolution of those metrics is monitored over multiple SFSeg iterations.

4.5.3 Qualitative analysis

We perform tests on synthetic data, in order to study the differences between the original spectral solution using the exponential pairwise scores (Eq. 16) and the one obtained after our first-order Taylor approximation trick (Eq. 17). In Fig. 17 we see qualitative comparisons between the solutions obtained by three implementations: 1) SFSeg, 2) classical power iteration and 3) eigenvector computed with Lanczos method in NumPy [60], with original pairwise scores. The outputs are almost identical. In this synthetic experiment, the input is very noisy, but all spectral solutions manage to reconstruct the initial segmentation.

4.5.4 Quantitative analysis

We analyze the numerical differences between the original eigenvector and our approximation (SFSeg). We plot the angle (in degrees) and the IoU (Jaccard) between SFSeg (first-order approximation of pairwise functions, optimized with 3D convolutions) and the original eigenvector (exponential pairwise functions in the adjacency matrix), over multiple SFSeg iterations in Fig. 18.

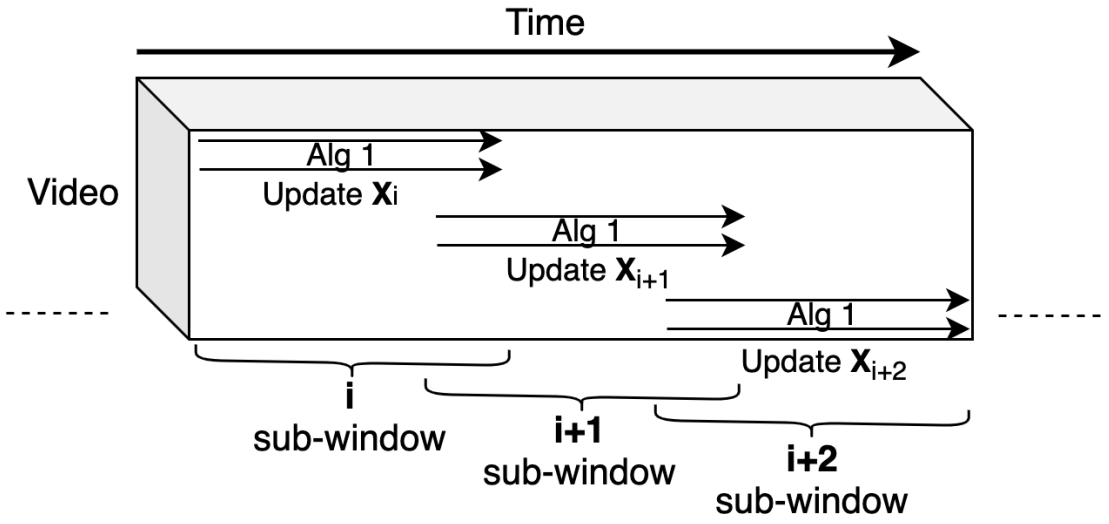

Figure 19: When the video is a contiguous stream or when it is very large, instead of applying our spectral method on the full video, we can apply fewer iterations on smaller video sub-windows, with similar effect. To speed up convergence, we initialize the current solution with the final solution over the previous sub-window (for the frames that overlap).

Note that in these experiments we intentionally start from a far away solution (70 degrees difference between the SFSeg initial segmentation vector and the original eigenvector) to better show that SFSeg indeed converges to practically the same eigenvector, even when starting with a weak initialization. Such comparisons can be performed only on synthetic data with relatively small videos, for which the computation of the adjacency matrix needed for the original eigenvector is tractable. The results clearly show that SFSeg, with first order approximations of the pairwise functions on edges and optimization based on 3D filters, reaches the same theoretical solution, while being orders of magnitude faster.

4.5.5 Online vs Offline processing

A full SFSeg++ iteration consists of passing through the entire video. In this form (Alg. 4), we can pass to the next iteration of the algorithm only after we go through the full video, making it an offline algorithm. That is because we need to pass (filter in 3D) through the whole video multiple times until we reach convergence.

With every iteration, local information is propagated one step further in both space and time. In practice, the method converges after several iterations. Therefore, far-away information is not that useful for determining the segmentation at a given frame. The main reason for that is that objects move and change their shape from one frame to the next, making far connections less relevant.

Based on this observation, we could expect that in practice we only need **partial iterations**, to apply the iterations on smaller sub-volumes of video (see Fig. 19). This also allows us to use SFSeg++ as an **online segmentation method**. We analyzed the convergence of the solution when instead of performing all iterations on the full video, we break it into a sequence of smaller number of iterations, in local sub-windows around a moving center frame. We observed almost identical results with the full iterations version, but at a lower computational complexity (see next paragraph). The speedup factor is identical to the size of the sub-window relative to the full video, for an equal number of iterations. Note however that all our speed comparisons to the Lanczos method are using iterations over the full video. Also all reported results are obtained with the original full video iterations method. As stated here, the two versions, with partial and full iterations, achieve the same results for all practical purposes.

NUMERICAL CONSIDERATIONS. The complexity of filtering in a sub-window is $\mathcal{O}(kqN_pN_i)$, where q is the number of frames in a sub-window. For the online version, q represents the number of frames in the past that we need to consider from the current incoming frame. It can be shown that sub-window filtering could reduce in practice the complexity of filtering over the full offline video by up to $\frac{N_f}{q}$ times. Fig. 19 can help visualize why this is the case.

4.6 EXPERIMENTAL ANALYSIS

4.6.1 Single-channel SFSeg

ON DAVIS-2016 DAVIS-2016 [130] is a densely annotated video object segmentation dataset. It contains 50 high-resolution video sequences (30 train/20 valid), with a total of 3455 annotated frames of real-world scenes. The benchmark comes with two tasks: the unsupervised one, where the solutions do not have access to the first frame of the video and the semi-supervised one, where the methods use the ground-truth from the first frame.

	Input Method	Input Score (J)	**SFSeg** over Input (J)	Improved Videos (%)
Semi Supervised	OnAVOS [163]	86.1	**86.3** (+0.2)	65
	OSVOS-S [109]	85.6	**86.0** (+0.4)	90
	PReMVOS [103]	84.9	**88.2** (+3.3)	90
	FAVOS [21]	82.4	**83.0** (+0.6)	95
	OSMN [185]	73.9	**75.9** (+2.0)	95
Un Supervised	COSNet [102]	80.5	**80.9** (+0.4)	65
	MotAdapt [143]	77.2	**77.5** (+0.3)	65
	PDB [147]	77.2	**77.4** (+0.2)	60
	ARP [77]	76.2	**77.7** (+1.5)	90
	LVO [157]	75.9	**78.8** (+2.9)	90
	FSEG [72]	70.7	**72.3** (+1.6)	95
	NLC [41]	55.1	**55.6** (+0.5)	65
	Average Boost		+1.1%	80%

Table 5: Single-channel SFSeg on DAVIS-2016 tasks. SFSeg has the same hyper-parameters per task. We also included results for other competitive (non-SOTA) inputs. 2^{nd} column: Jaccard score for the input method; 3^{rd} column: score after applying SFSeg over the input method; 4^{th} column: the percentage of videos when the performance is improved after using SFSeg. The average SFSeg boost is 1.1% in Jaccard score. On average SFSeg raises performance for 80% of videos.

We test the single-channel version, SFSeg, with input from pre-computed segmentations of the video produced by top methods from DAVIS-2016, on both tasks. For the features maps, we initialized $\mathbf{S}$ with the pre-computed input segmentation values. For $\mathbf{F}$, we used two channels: the magnitude for the direct optical flow and for the reverse optical flow. For optical flow we used [136] implementation of Flownet2 [69]. We set: $N_i = 5$; $\alpha = 1$ and $p = 0.1$ for unsupervised task and $p = 0.2$ for the semi-supervised one. The algorithm is implemented as in Alg. 4. For the temporal consistency metric we used the same optical flow implementation.

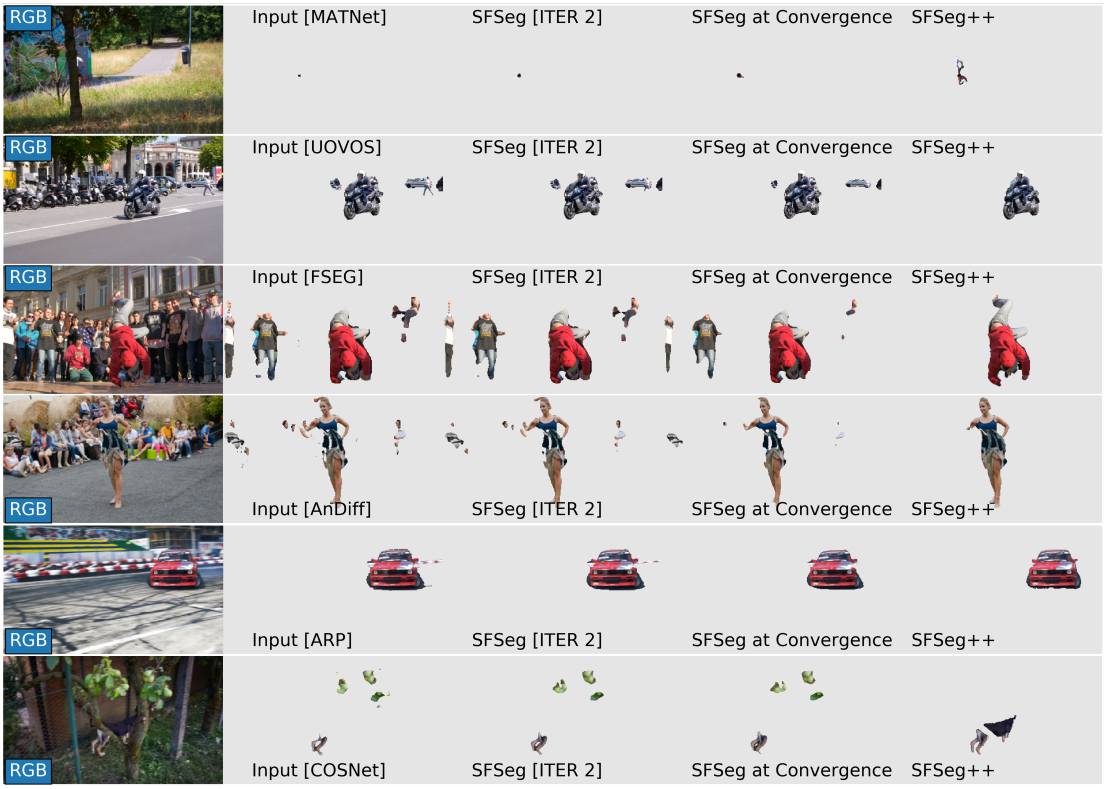

Figure 20: We present the evolution of the single-channel SFSeg towards convergence, over several iterations, starting from the output of different methods, as state in square parenthesis. Using the input segmentation mask (column 2) from top methods: [201, 203, 72, 186, 77, 102], we show the intermediate value of the mask at second iteration (column 3) and on convergence (iteration 5, column 4). On the last column, we also show for comparison, the output of the learned multi-channel SFSeg++ (with 15 input channels, from 15 different methods). Note the vastly superior results of SFSeg++, which learns to combine many channels, learning both to recover occluded objects (1st and last line), but also to remove distractors (lines 2-5).

In Tab. 5 we show the results of our method, SFSeg, when combined with top methods on DAVIS-2016, semi-supervised and unsupervised tasks. For a better understanding of the results, we also show the effect of applying SFSeg over other competitive, non-SOTA methods. We noted that the improvement is not related with the quality measure of the input: in some cases the improvement is stronger when input comes from stronger methods. Nevertheless, we

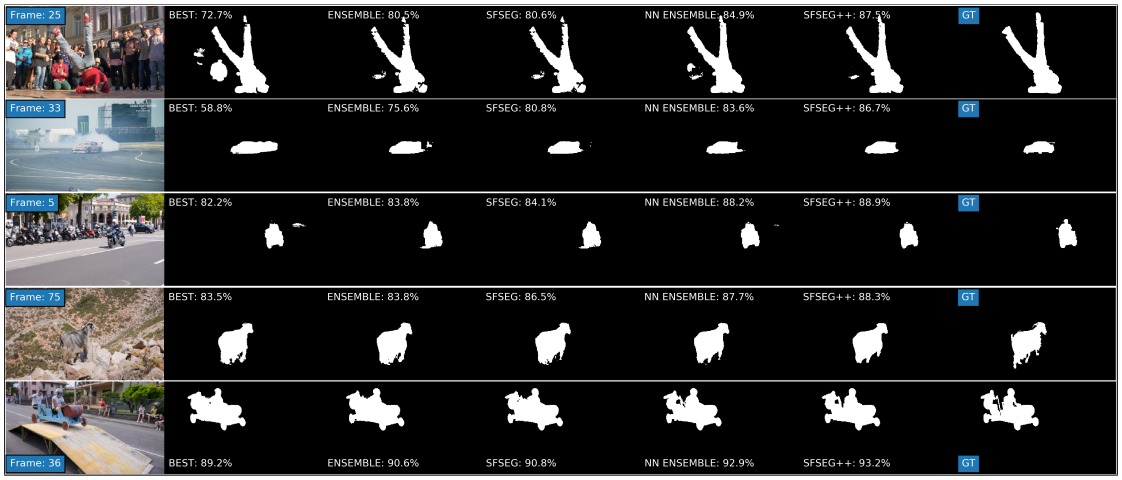

Figure 21: Qualitative output for several videos in DAVIS-2016, from left to right: the best method (which is also part of the ensemble), the ensemble output (arithmetic mean over 15 top methods), SFSeg applied over the ensemble output, the learned ensemble output with neural nets and finally, our SFSeg++. On the last column we also show the ground-truth (GT) for comparison. Both SFSeg and SFSeg++ improve significantly (especially the multi-channel learnable SFSeg++) over their input channels.

consistently improve over the input method, whose segmentation mask we use to initialize the segmentation $\mathbf{X}_0$ (Eq. 22).

In Fig. 20 we show the iterative effect of SFSeg. Each example starts from the initial RGB frame and its initial segmentation (as produced by top DAVIS-2016 methods), and presents the segmentation at an intermediate iteration and the final one, when SFSeg reaches convergence. For comparison, we also show the output of the learnable multi-channel SFSeg++. Note how SFSeg++ learns to get the best of the combination of multiple inputs, each with significantly lower performance. Also note how the iterative 3D spectral filtering process of SFSeg improves from one iteration to the next.

In Fig. 21 we show comparative qualitative examples between the output of the best method in the ensemble (MATNet [201]), the arithmetic mean, SFSeg over this mean, the learned ensemble and SFSeg++.

ON SEGTRACKV2 SegTrackv2 [95] is a video object segmentation dataset, containing 14 videos, with multiple objects per frame. The purpose for video object segmentation task is to find the segmentation for all the objects in the frame, either by using the first frame or not. We use our standalone method, SFSeg,

Method	Score (J)
LVO [157]	57.3
FSEG [72]	61.4
OSVOS [110]	65.4
NLC [41]	67.2
MaskTrack [76]	70.3
BB + **SFSeg** + denseCRF (ours)	**72.7**

Table 6: Comparative results on SegTrackv2. Our standalone solution, Backbone + SFSeg + denseCRF, obtains the best results among the other top methods in the literature.

applied over the soft output of a competitive **Backbone (BB)**: UNet [138] over ResNet34 [61] pretrained features, fine-tuned 40 epochs on salient object segmentation in images on DUTS dataset [168], using RectifiedAdam as optimizer. In Tab. 6 we show comparative results of our standalone method and other top solutions on the SegTrackv2 dataset.

4.6.2 Single-channel SFSeg vs denseCRF

We compare SFSeg with denseCRF [82], which is one of the most used refinement methods in video object segmentation [147]. When applied over the same backbone presented above, we observe that SFSeg brings a stronger improvement than denseCRF on both DAVIS-2016 and SegTrackv2 (Tab. 7). Moreover, the two are complementary: in combination, the performance is boosted by the largest margin.

4.6.3 Learned Multi-channel SFSeg++

Before we present the actual tests performed with SFSeg++, we provide more details about the exact experimental setup. In the comparative experiments, SFSeg++ uses, as input feature channels, the outputs of the top 15 methods, as published on DAVIS-2016 website, for the task where no ground-truth is given in the first frame of the test videos (officially labeled the "Unsupervised task"): MATNet [201], AnDiff [186], COSNet [102], ARP [77], UOVOS [203], FSeg [72],

Method	DAVIS (J)	SegTrack v2 (J)
BB	67.2	72
BB + denseCRF	68.1	72
BB + SFSeg	68.7	72.1
BB + SFSeg + denseCRF	**69.2**	**72.7**

Table 7: Refinement Comparison. We compare SFSeg with denseCRF when applied to a competitive end-to-end Backbone (as detailed in Sec. 4.6.1). While SFSeg outperforms denseCRF when used individually, the two methods prove to be not only different, but also complementary, since combining them boosts the Jaccard score.

LMP [156], TIS [56], ELM [88], FST [127], CUT [74], NLC [41], MSG [126], KEY [89], CVOS [153], TRC [47]. The experiments which test the ability of our learning approach to deal with noisy input channels, modify these initial channels according to the different types of noise used, as explained in the next Section 4.6.3. Also, in all experiments we train all models in a supervised fashion only on the DAVIS-2016 training split. Results are, of course, reported on DAVIS-2016 validation set. Thus we follow the exact official protocol of DAVIS-2016 challenge. For efficiency and compactness, we also observed that in the case of SFSeg++ we can use the same 15 initial channels to construct both unary and pairwise maps. In other words $\mathbf{S}_i = \mathbf{F}_i$ and their learned weights are also shared: $w_{s,i} = w_{f,i}$.

APPLYING MULTI-CHANNEL SFSEG++ OVER NOISY INPUT To better understand the effectiveness of learning over multiple input channels we first analyse our method's robustness to noise. Thus, we added different types of noise to the input channels. First, we keep only the top 3 original input methods, replacing the rest 12 methods with noise. Second, we only disturb the rest 12 methods with noise. Third, we used a combination of the two: 5 methods disturbed and 7 replaced with noise.

We tested with three types of noise: **a) Uniform (U)** - This introduces a low to moderate level of noise. We add uniform noise in $[0, 0.1]$, uniformly over the whole input map (which has values between 0 and 1 and normalize the result, **b) Salt&Pepper (SP)** - This noise is more invasive for the original maps. We

randomly replace pixels with 0 or 1, with 20% probability and **c) Black and White randomly placed Rectangles (BWR)** - This is an aggressive and more structured type of noise, to simulate the effect of an occlusion or per frame distractor. We add one rectangle per frame, uniformly sampling its position (inside 100px margins) of randomly sized weight and height, filling between 2% to 5% of the total frame size. We present the results and comparisons in Tab. 8, where for the basic Ensemble we use the average output of the 15 different methods and for the Best Learned Ensemble we selected among the different Deep Neural Nets architectures that learn over the same 15 input channels, over a certain temporal window (see Sec. 4.6.3 for more details).

In Fig. 22 we show the actual weights learned for the different cases of added noise or for the case of using the original input channels. Note that the input methods (channels) are presented in the descending order of their individual performance. We can immediately notice that SFSeg++ learns to select the most relevant channels (larger weights) and ignore the more noisy or weaker ones (lower weights). For example, when no noise is added (Fig. 22 - Top 15), the weights are correlated with performance, thus channels with better performance (on the left side of the plot) tend to have larger weights than the ones with weaker performance (towards the right side of the plot). When noise is added to some channels (labeled as "Noisy" in Fig. 22) or some channels are completely replaced with noise (labeled as "Noise" in Fig. 22), the top 3 methods (without noise) receive very strong weights, while the remaining ones much weaker weights. It is also interesting that for the same type and amount of noise being added, the middle 5 channels have larger weights than the last 7 channels, as they have better individual performance (case 2). Also, when noise is added to some channels while others are replaced by noise (case 3), the noisy ones receive significantly stronger weights than the ones that are complete noise, with close-to-zero weights. The visual analysis of the weights reveals that learning is indeed meaningful: it automatically discovers which channels are relevant and which ones are not and weighs them accordingly.

Key observations: When interpreting the learned weights over the input channels, we observed that the channels having noise added receive very low weights and become mostly ignored. The selection of channels taking place during learning focuses more on top methods or channels without noise. This shows the efficiency and robustness of our learning. Also note that while the single-channel SFSeg is able to improve significantly over poor inputs (SP), SFSeg++ truly distances itself from even the best learned ensembles. SFSeg++ achieves best performance in all cases, highlighting that our algorithm learns to take

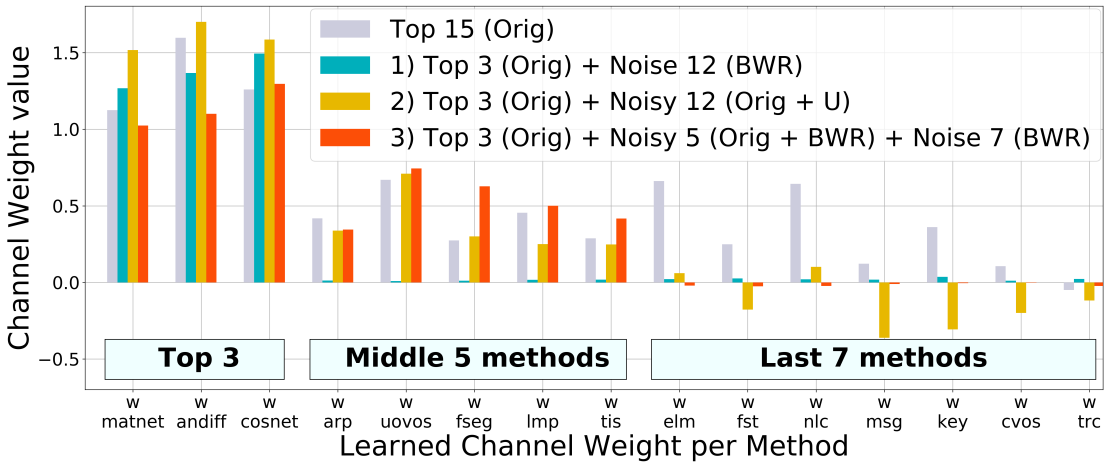

Figure 22: Learned Channels Weights: We display the learned weights values for SFSeg++ learning over unmodified input channels from top 15 methods as well as the weights for the different cases of added noise (as explained in Sec. 4.6.3). The input methods are shown from left to right, in descending order of their individual performance. We immediately notice that the learned weights are larger for channels from methods with superior performance (towards the left: **top 3 vs the remaining 12** or **middle 5 vs last 7**) or channels not corrupted by noise (**top 3 vs the remaining 12**). This indicates that learning is meaningful: it is an efficient and automatic way to select the relevant input channels.

advantage of relevant details in input maps better than any competitor - due to the combination of learning and clustering over the temporal dimension.

COMPARING SFSEG++ WITH LEARNED NEURAL NET ENSEMBLES To better understand the value of the space-time clustering approach combined with learning as opposed to simply learn to combine multiple input channels, we compared against some powerful deep neural networks ensembles, which take as input the same channels as SFSeg++. We varied the number of parameters and architectures of the deep networks, to make sure the comparisons do not depend on the specific neural net size and architecture.

The learning task was for the net to learn to predict the final segmentation, in a supervised way, using the DAVIS-2016 training set for which ground-truth segmentations are available. Therefore, SFSeg++ and all neural net ensembles had exactly the same input (at testing and training) and the same supervised training set. Also note that the neural net ensembles also had access to inputs

Noise Type	Noise Combination	Ensemble	SFSeg over Ensemble	Best Learned Ensemble	SFSeg++
U	Top 3 + Noise 12	80.6	80.7	84.8	**85.1**
	Top 3 + Noisy 12	83.5	83.7	84.4	**85.4**
	Top 3 + Noisy 5 + Noise 7	84.2	84.2	84.9	**85.2**
SP	Top 3 + Noise 12	61.5	62.5	**85.5**	**85.5**
	Top 3 + Noisy 12	81.0	83.3	**84.9**	**84.9**
	Top 3 + Noisy 5 + Noise 7	83.8	84.0	**85.0**	**85.0**
BWR	Top 3 + Noise 12	79.6	80.2	85.5	**85.6**
	Top 3 + Noisy 12	82.2	83.0	84.8	**84.9**
	Top 3 + Noisy 5 + Noise 7	84.3	84.3	**85.9**	**85.9**
None	Top 15	83.5	83.6	85.5	**86.6**

Table 8: SFSeg++ Noisy Input: The experiments are done for 3 types of noise: Uniform (U), Salt&Pepper (SP), Black and White Rectangles (BWR) (see Sec. 4.6.3 for details). We test the performance of SFSeg++ in 3 combinations of the noise with the 15 input methods. Note that the largest difference between SFSeg++ and the best learned NN ensemble is on Uniform noise cases, where the input is less affected by noise. This proves that training with SFSeg++ over several input channels exploits more relevant details in input compared with other learned ensembles. It also shows the effectiveness of clustering over the temporal dimension.

over several time frames (5-frames temporal windows), thus the temporal dimension was not absent from the neural net ensembles. Of course that SFSeg++ has access to the full video, indirectly through the space-time power iteration process, but, as stated previously (Sec 4.5.5) it is not the width of the temporal window that matters most, but rather the iterative clustering procedure. Moreover, during the SFSeg++ training phase, for efficiency and without loss in performance, we only consider sub-videos of 5 consecutive frames, which are in fact identical to the ones considered by the most powerful learned ensembles.

NN Ensemble Method	Number of parameters	Jaccard (%)
Unet3D large	1.4 mil	85.4
Unet3D small	330k	85.4
Unet2D	140k	83.9
Shallow2 2D	35k	84.8
Shallow3 2D	5k	84.0
Shallow4 2D	380	85.5
SFSeg++	16	**86.6**

Table 9: Comparison to learned neural net ensembles: we trained various ensembles' net architectures (2D, 3D, UNet based, shallow NN), with different numbers or parameters, using a relatively small training set of 30 videos, starting with the same input maps from the top 15 input methods as SFSeg++. Note that SFSeg++ outperforms the best learned NN ensembles by 1.1%, with only 16 learned parameters (one parameter per input channel and the bias term).

We started creating simple and then more complex neural net ensembles, w.r.t architecture and number of parameters. Interestingly enough, we found out that a deep net ensemble with many parameters is not so strong as one with fewer parameters. Perhaps this is not that surprising considering how small the training set is (30 training videos), so large nets are prone to overfitting on this small training set. Moreover, note that the arithmetic mean is usually a very strong baseline when we average over multiple top methods.

In order to choose a good neural net ensemble for segmentation we started from UNet [138] architecture for 2D (per frame) and adapted it for 3D (per 5 frames), with a various number of parameters (between 380 - 1.4 millions). For the small network ensembles we stacked 2 to 3 convolutional layers (with kernel size of $1, 3, 5$) followed by ReLU non-linearity. As in the case of SFSeg++, we optimize the Focal-Dice loss [170] (with 0.75 coefficient) using an AdamW optimizer with AMSGrad [25] from Pytorch [97], with a ReduceLROnPlateau scheduler (with a reduce factor of 0.5) and a strong L2 regularization, for each architecture choosing the best among learning rates (1e-1, 1e-2, 1e-3, 1e-4, 1e-5), L2 regularizer factor (5e-1, 1e-2, 5e-2, 1e-3, 5e-3) and scheduler patience for reducing the learning rate ($5, 10$ epochs).

Tab. 9 confirms that the 3D versions perform significantly better compared with the 2D ones. As mentioned above, the experiments also come with the interesting result that under our task assumptions (learn an ensemble over very strong state-of-the-art input methods with a small training and validation sets), the smaller nets often perform better.

Key observations: The experimental comparisons between the learnable multi-channel SFSeg++ and different types and sizes of neural net ensembles show a clear advantage for SFSeg++, which is mainly due to several factors: 1) the ability of SFSeg++, through space-time clustering, to better take advantage of the temporal dimension, naturally and with minimal training data required. 2) the ability of SFSeg++ to not overfit for small training sets, given its small set of learnable weights as compared to the much larger neural nets. Note, however, as also stated in the theoretical section on learning (Sec. 4.4.3), that in principle SFSeg++ could also learn its input deep feature extractors, end-to-end, if needed, as long as they are learnable (differentiable: *e.g.* deep nets) themselves.

LEARNED MULTI-CHANNEL SFSEG++ ON DAVIS-2016 This experiment focuses on the improvements SFSeg++ could bring over state-of-the-art and other top, competitive approaches: a learned NN ensemble (as presented in the previous section), basic ensemble (average of 15 single channel outputs) and top single-channel methods. All training was performed only over the DAVIS-2016 trainset of 30 videos, with ground-truth segmentations for all frames available. Again, all methods receive the same input from 15 segmentation methods, as previously described in Sec. 4.6.3. We report results in Tab. 10 and in Fig. 23.

Key observations: While SFSeg outperforms current single state-of-the-art in DAVIS-2016, the learnable multi-channel SFSeg++ outperforms the best learnable NN ensembles by a solid 3.1%. It is interesting to note that SFSeg alone, without learning applied over the single-channel output of the standard average ensemble, further improves over the ensemble. Then, SFSeg++, which has the ability to learn over multiple input channels, outperforms the best learned NN ensemble. These two results strongly indicate the complementary value that learning and iterative space-time clustering bring, over the more traditional feed-forward pass in the conv neural nets, which do not take full advantage of the temporal dimension. These experiments along with the ones from the previous Sec. 4.6.3 also show that in the case of small supervised training data, a general, non-specific, data independent space-time clustering

Method	Jaccard (%)	J Diff over Best (%)	J Diff over Ensemble (%)
Weakest Method (TRC [47])	47.3	-35.1	-36.2
3rd Best Method (COSNet [102])	80.5	-1.9	-3.0
2nd Best Method (AnDiff [186])	81.7	-0.7	-1.5
Best Method (MATNet [201])	82.4	0	-1.1
SFSeg(Best Method)	82.7	+0.3	-0.8
Ensemble	83.5	+1.1	0
SFSeg(Ensemble)	83.6	+1.2	+0.1
Best NN Ensemble	85.5	+3.1	+2.0
SFSeg++	**86.6**	**+4.2**	**+3.1**

Table 10: SFSeg++ performance and comparisons to top methods and ensembles over DAVIS-2016 validation set. When starting from 15 input methods, both SFSeg and SFSeg++ improve over their input, outperforming competition, which comprises of state-of-the-art methods on DAVIS-2016 (+0.3% vs +4.2%) and the classic Ensemble, as average over its input channels (+0.1% vs +3.1%). Most powerful, SFSeg++ proves to bring complementary value to supervised learning, as it outperforms the best learned NN Ensemble by 1.1%.

approach combined with a small set of learned weights could be significantly more effective than models that rely heavily on supervised training data.

4.6.4 TCONT: Temporal consistent SFSeg++

We analyze next the influence of our SFSeg++ algorithm on the temporal consistency of the segmentation masks over the video volume, using the newly introduced TCONT metric, detailed in Sec. 4.4.4. We apply TCONT with respect to ground-truth, for the basic Ensemble and SFSeg++, showing the results in Tab. 11. We also present some qualitative samples focusing on the temporal consistency in Fig. 24.

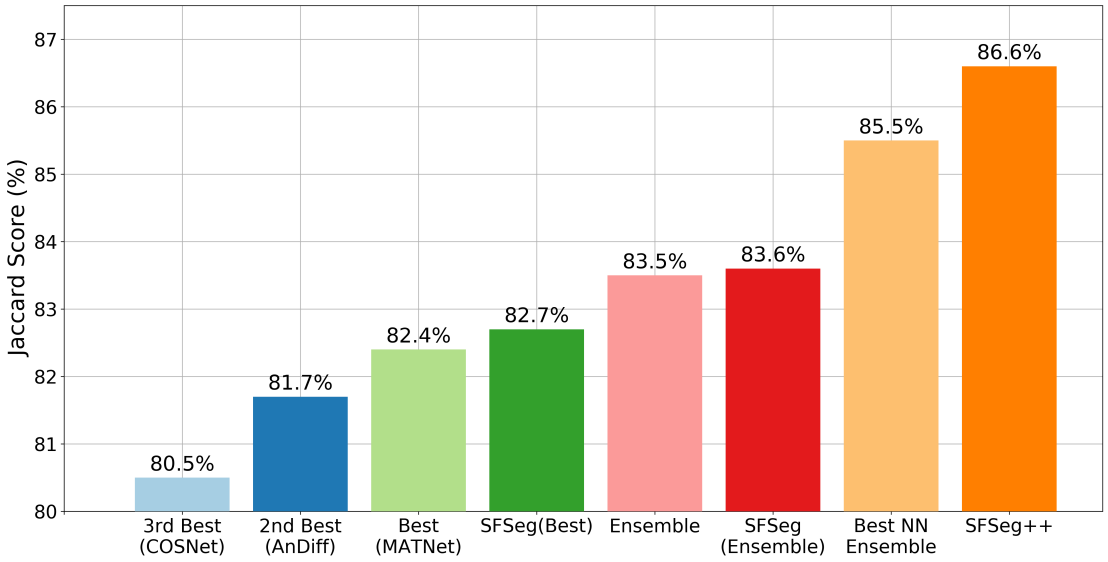

Figure 23: Visual experimental comparisons: to better understand the difference in performance between various methods, we show them here, as plot bars, in ascending order, starting from the top 3 single methods, SFSeg on the Best method, a basic average Ensemble, SFSeg applied over the average Ensemble, the Best learned neural net (NN) Ensemble and SFSeg++. While SFSeg improves over single-channel input methods, SFSeg++ boost is significantly larger, even when compared with the Best learned NN Ensemble.

Key observations: After applying SFSeg++, the temporal consistency of the output is significantly improved by 4.5% with respect to ground-truth. This highlights the fact that our algorithm not only improves the overall segmentation accuracy, but also makes it more consistent with the object motion field, a property that is desired in the context of videos. We believe that the main reason for our improved temporal consistency comes directly from the fact that our special video 3D filtering is designed from the start having both space and time in mind, which makes it more suitable for object segmentation in video than other more traditional approaches that start from the level of individual frames.

Intuitively, the difference between two methods having a similar frame level accuracy, but with different consistency measures at temporal, video level, could be better understood by the following example. Even though the two approaches have, initially similar accuracy, the method which is more stable with respect to the motion field is more likely to produce a segmentation of

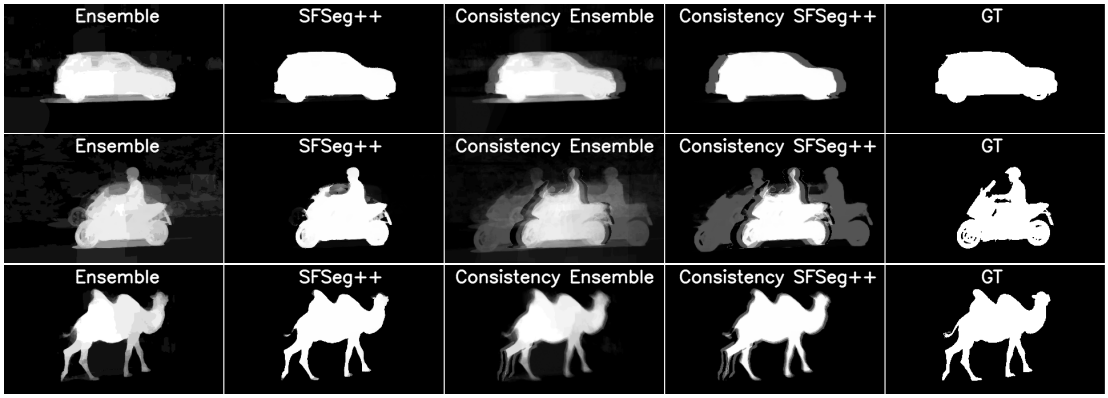

Figure 24: Visualization examples of temporal consistency: in the first two columns we show the basic Ensemble (average over top 15 methods) and our SFSeg++. In the 3^{rd} and 4^{th} column we show the mask results computed by TCONT (our temporal consistency) for the Ensembles and SFSeg++, respectively, while the ground truth is shown in the last column. We show TCONT views for three examples in different videos, computed using Eq. 29, as explained in Sec. 4.4.4, while taking into account a temporal window (left, middle, and right frame) averaged after warping with optical flow. Please observe that the average map, used for measuring consistency, is crisper and of better quality in the case of SFSeg++. This shows on one hand, a better consistency with respect to itself and with respect to ground-truth, given the independently computed object motion field (as estimated by the optical flow).

higher quality after optical flow warping than the one which is less stable with respect to motion. Therefore by measuring such stability with respect to motion, we are in fact looking at the ability of a method to intrinsically consider time and motion, and that ability is definitely a desired property for any video object segmentation procedure.

4.6.5 Running time

The algorithm scales well, its runtime being linear in the number of video pixels, as detailed in Sec. 4.5. For a frame of 480×854 pixels, it takes 0.01 sec per iteration, compared with 0.8 sec for denseCRF. The time penalty of adding SFSeg is minor for most methods, which take several seconds per frame (*e.g.* 4.5 sec per frame [109], 13 sec per frame [103]). We tested on a GTX Titan X Maxwell GPU, in Pytorch [97].

	Temporal Consistency IoU wrt GT (%)
Weakest Method [47]	46.0
Best Method [201]	77.0
Ensemble	75.7
SFSeg++	**80.2**
	+4.5%

Table 11: TCONT metric for measuring temporal consistency. We compute the TCONT metric for best and weakest methods, for the Ensemble over all 15 input methods and for our SFSeg++. SFSeg++ increases the temporal consistency of the input by 4.5% with respect to GT when compared with the basic Ensemble.

4.7 CONCLUDING REMARKS

We formulate video object segmentation as clustering in the space-time graph of pixels. We introduce an efficient spectral algorithm, **Spectral Filtering Segmentation** (SFSeg), in which the standard power iteration for computing the principal eigenvector of the graph's adjacency matrix is transformed into a set of 3D convolutions applied on 3D feature maps in the video volume. Our original theoretical contribution makes the initial intractable problem possible. Then we make the 3D spectral filtering algorithm fully learnable, and extend it to SFSeg++, which performs efficient spectral clustering with 3D convolutions over learned combinations of input channels. We validate experimentally the value of each technical contribution. First, we show theoretically that our fast solution based on first-order Taylor approximation of the original pairwise potential used in spectral clustering is practically equivalent to the original one. Secondly, in experimental comparisons, we show that the single-channel SFSeg consistently improves (for 80% of videos), when applied as a refinement procedure over all top published video object segmentation methods, at a small additional computational cost. Moreover, we also show that SFSeg also achieves top performance in combination with other backbone networks (not necessarily state-of-the-art). Thirdly, we demonstrate the effectiveness and robustness of learning over multi-channels by experimenting with different types of noisy in-

put channels. And last, we introduce TCONT, a temporal consistency measure, which we use to analyze the methods' consistency with respect to the object motion field in video. The analysis indicates that SFSeg++ is indeed more stable with respect to the object' motion field (computed independently), which validates our motivation for the spatio-temporal clustering approach that is meant to preserve such stability in space and time. We test the learnable multi-channel SFSeg++ against powerful learned neural network ensembles, using the same input masks provided by 15 published methods. Thus, we validate all four key aspects of our contributions: **1**) the efficiency of the 3D filtering approach as a way to perform spectral space-time clustering; **2**) the effectiveness of such clustering in real experiments; **3**) the efficiency of learning over multiple channels and its robustness to noise and **4**) the superior results obtained by our spectral approach in all comparative experiments performed. In addition, SFSeg++ with minimal number of parameters for learning is better and more robust in the face of limited supervised training data, than powerful ensembles learned with deep neural networks. The consistent improvements in practice over different top quality input channels, brought by SFSeg and SFSeg++, indicate that our spectral approach brings a new and complementary dimension to clustering in space-time, which is not fully addressed by the current published solutions.

Next in the thesis I integrate this space-time approach in the tracking pipeline, showing the benefits of having a spectral clustering solution in video, but at the level of very refined feature space, like the segmentation mask.

5 IMPROVING TRACKING WITH 3D SPECTRAL SEGMENTATION

We propose an object tracking method, SFTrack++, that smoothly learns to preserve the tracked object consistency over space and time dimensions by taking a spectral clustering approach over the graph of pixels from the video, using a fast 3D filtering formulation for finding the principal eigenvector of this graph's adjacency matrix, proposed before in the context of instance segmentation. To better capture complex aspects of the tracked object, we use multi-channel inputs, which permit different points of view for the same input. The channel inputs are in our experiments, the output of multiple tracking methods. After combining them, instead of relying only on hidden layers representations to predict a good tracking bounding box, we explicitly learn an intermediate, more refined one, namely the segmentation map of the tracked object. This prevents the rough common bounding box approach to introduce noise and distractors in the learning process. We test our method, SFTrack++, on five tracking benchmarks: OTB, UAV, NFS, GOT-10k, and TrackingNet, using five top trackers as input. Our experimental results validate the pre-registered hypothesis. We obtain consistent and robust results, competitive on the three traditional benchmarks (OTB, UAV, NFS) and significantly on top of others (by over 1.1% on accuracy) on GOT-10k and TrackingNet, which are newer, larger, and more varied datasets.

5.1 ZOOMING OUT: SEGMENTATION WITHIN TRACKING

The previous chapter presents SFSeg++, an efficient object segmentation solution in video focused on space and time integration. This chapter focuses on SFTrack++, a tracking solution that integrates previous findings on segmenta-

E. Burceanu, SFTrack++: A Fast Learnable Spectral Segmentation Approach for Space-Time Consistent Tracking, Neural Information Processing Systems - Pre-registration Workshop (NeurIPSW) 2020

tion in the tracking task, emphasizing the need to have a more refined representation as an intermediate task rather than using just rough tracking bounding boxes. We analyzing this chapter following the key aspects introduced at the beginning of the thesis:

- **A. Space-time consistency**: Space and time interact here in a similar way like in the previous chapter, with the difference that here the segmentation representation is of a lower quality because it comes as a learned neural transformation from bounding boxes (compared with the output of the top segmentation solutions).
- **B. The power of the consensus**: This point is also similar to SFSeg++. From the segmentation point of view, we iteratively reach the consensus in each neighbourhood, at convergence. And in the second part, we ensemble over top tracking methods.
- **C. Exploiting multiple intermediate representations**: Empirical results over five tracking benchmarks show with confidence that using the segmentation as an intermediate representation for tracking helps a lot. We refine those more natural descriptors of the object and compare the final result against the initial input or other ensemble versions for tracking.
- **D. A limited quantity of supervision**: Except for the clustering part where we refine the middle representation (the segmentation), the approach is fully supervised. We will take a step further in this direction in the next chapter.
- **E. Experts: Making use of existing models**: We use experts (state-of-the-art trackers on various benchmarks) and learn how to combine them for achieving an overall better performance. We see the limits of our 3D spectral segmentation in tracking when working with top methods as input channels.

5.2 CONTEXT

Better using the temporal aspect of videos in visual tasks has been actively discussed for a rather long time, especially with the large and continuous progress in hardware. The first aspect we tackle in our approach is a seamless blending of space and time dimensions in visual object tracking. Current

methods mostly rely on target appearance and frame-by-frame processing [110, 93, 30], with rather few taking explicit care of temporal consistency [10, 71]. In the spectral graph approach, nodes are pixels and edges are their local relations in space and time, while the strongest cluster in this graph, given by the principal eigenvector of the graph's adjacency matrix, represents the consistent main object volume over space and time.

A second observation challenges the rough bounding box (bbox) shape used for tracking. While it provides a handy way to annotate datasets, it is a rather imperfect label since it leads to errors that accumulate, propagate, and are amplified over time. Objects rarely look like boxes, and bboxes contain most of the time significant background information or distractors. Since having a good segmentation for the interest object directly influences the tracking performance, we constrain an intermediary representation, a segmentation map, which aims to reduce the quantity of noise transferred from a frame to the next one. We integrate it into our end-to-end flow, as shown in Fig. 25.

A third point we emphasize on is relying on multiple, independent characteristics of the same object or multiple modules specialized in different aspects. This comes with an improved ability to understand complex objects while increasing the robustness [10]. We, therefore, use the SFSeg++ spectral approach, where for the channel inputs in its formulation we use tracking outputs from multiple solutions.

The **main contributions** of our approach are:

- SFTrack++ brings to tracking a natural, contiguous, and efficient approach for integrating space and time components, using a fast 3D spectral clustering method over the graph of pixels from the video, to strengthen the tracked object's model.
- We explicitly learn intermediate fine-grained segmentation as opposed to rough bounding boxes in our three phases end-to-end approach to a more robust tracking solution.
- We integrate into our formulation a way of learning to combine multiple input channels, offering a wider view of the objects, harmonizing different perceptions, for a powerful and robust approach.

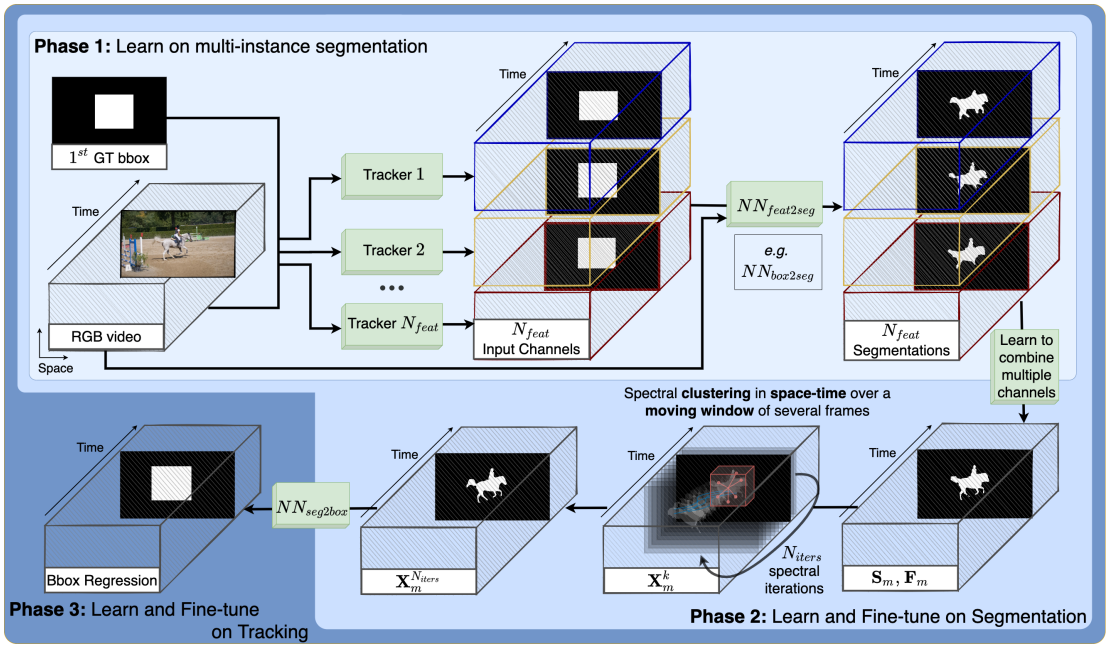

Figure 25: SFTrack++: We start from video's RGB and 1^{st} frame GT bbox of the tracked object. We run state-of-the-art trackers, in an online manner, while fine-tuning $\mathrm{NN}_{\mathtt{feat2seg}}$ network frame-by-frame (pretrained in Phase 1) to transform the extracted feature maps (*e.g.* bboxes) to segmentation maps. Next, we learn to combine multiple segmentation inputs and refine the final mask using a spectral approach, applied also online over a moving window containing the previous N frames, for $\mathrm{N}_{\mathtt{iter}}$ spectral iterations (Phase 2). In Phase 3, we learn a bbox regressor from the final segmentation mask, $\mathrm{NN}_{\mathtt{seg2box}}$ and fine-tune all our parameters on the tracking task.

5.3 RELATION TO PRIOR WORK

GENERAL OBJECT TRACKING Out of the three main trackers families, **Siamese based trackers** gained a lot of traction in recent years for their high speed and end-to-end capabilities [9, 94, 93, 198]. Most approaches focus on exhaustive offline-training, failing to monitor changes w.r.t. the initial template [152, 181, 20], while others update their model online [164, 27]. Nevertheless, the robustness towards unseen objects and transformations at training time remains a fundamental problem for Siamese trackers.

META-LEARNING APPROACHES FOR TRACKING [128, 7, 166] come with an interesting way of adapting to the current object of interest, while keeping a short inference time, by proposing a target-independent tracking model. One major limitation for both those approaches, Siamese and meta-learning trackers, is that they fail to adapt continuously to the real-time changes in the tracked object, using rather a history of several well-chosen patches or even just the initial one. In contrast, our method naturally integrates the temporal dimension, by continuously enforcing the local temporal and spatial object consistency.

DISCRIMINATIVE METHODS [30, 33, 105] on the other hand are classic approaches, focusing more on changes in the tracked object [29] (background, distractors, hard negatives), better integrating the temporal dimension [10] in the method flow. They prove to be robust, but they mostly rely on hand-crafted observations or modules not trainable end-to-end. SFTrack++ provides an end-to-end approach while minimizing the distractors and background noise using the intermediary segmentation map. Our method distances itself from a certain family of trackers, introducing the space and time consistency endorsement via clustering component as an additional dimension of the algorithm.

With a few notable exceptions [171, 162], most tracking solutions use internally hidden layer representations extracted from the previous frame's rough bbox prediction [10, 33, 30, 181], rather than a fine-grained segmentation mask as in our approach. Also, most of them do not take into account multiple perceptions for the input frame and operate over a unique feature extractor [30, 198, 9]. There are a few trackers though that combine two models for adapting to sudden changes while remaining robust to background noise, by explicitly model the different pathways [10, 15]. In contrast, our end-to-end multi-channel formulation learns over 10 input channels, a significantly larger number.

GRAPH REPRESENTATIONS Images and videos were previously represented as graphs, where the nodes are pixels, super-pixels, or regions [70]. This choice directly impacts the running time and performance. Regarding edges, they are usually undirected, modeled by symmetric similarity functions, but there are also several works that use directed ones [160, 189].

SPECTRAL CLUSTERING Basic approaches [125, 113, 90] search for the leading or the smallest eigenvector for the graph's adjacency matrix to solve the clusters' assignments. Spectral clustering was previously used in pixel-level image segmentation [141], with a high burden on the running time and in building

space-time correspondences between video patches [70]. Graph Cuts is a common approach for spectral clustering, having many variations [141, 35, 140]. SFSeg++ Chapter 4 proposes a 3D filtering technique for efficiently finding the spectral clustering solution without explicitly computing the graph's adjacency matrix. Inspired by this method, we learn over multi-channel inputs coming from top trackers output, as an intermediate component in our tracker, as detailed in Sec. 5.4.

5.4 OUR APPROACH

PHASE 1 SFTrack++ algorithm has three phases, as we visually present them in Fig. 25. In Phase 1, we learn a neural net, $\text{NN}_{feat2seg}$, that transforms the RGB and a frame-level feature map extracted using a tracker (*e.g.* bbox from a tracker prediction) into a segmentation mask. Using only the RGB as input is not enough, because frames can contain multiple objects and instances, and we also need a pointer to the tracked object to predict its segmentation.

PHASE 2 Next, we run multiple state-of-the-art trackers frame-by-frame over the input as an online process and extract input channels from them (*e.g.* bboxes). We transform those feature maps to segmentation maps with the previously recalled module, $\text{NN}_{feat2seg}$. Next, we learn to combine and refine the outputs for the current frame using a spectral solution for preserving space-time consistency, adapted to learn over multiple channels. Note that, when applying the spectral iterations, we use a sliding window approach over the previous N frames in the video volume. For supervising this path, we use segmentation ground-truth.

PHASE 3 Finally, we learn a neural net as a bbox regressor over the final segmentation map from the previous phase, $\text{NN}_{seg2box}$, while fine-tuning all the other trainable parameters in the model, using tracking GT.

SPECTRAL APPROACH TO SEGMENTATION. We go next through the following aspects, briefly explaining the connection between them: segmentation $\rightarrow$ leading eigenvector $\rightarrow$ power iteration $\rightarrow$ 3D filtering formulation $\rightarrow$ multi-channel. Image segmentation was previously formulated as a graph partitioning problem, where the segmentation solution [141] is the leading eigenvector of the adjacency matrix. We used it in a similar way for video segmentation

in Chapter 4. Power iteration algorithm can compute the leading eigenvector: $\mathbf{x}_i^{k+1} \leftarrow \sum_{j \in \mathcal{N}(i)} \mathbf{M}_{i,j} \mathbf{x}_j^k$, where $\mathbf{M}$ is the $N \times N$ graph's adjacency matrix, N is the number of nodes in the graph (pixels in the video space-time volume in our case), $\mathcal{N}(i)$ is the space-time neighbourhood of node i and each step k is followed by normalization. The adjacency matrix used in power iteration usually depends on two types of terms: unary ones are about individual node properties and pairwise ones describe relations between two nodes (pairs).

Following this approach, SFSeg rewrites power iteration using 3D filtering for an approximated adjacency matrix. The solution is described in Eq. 31:

$$\mathbf{X}^{k+1} \leftarrow \text{normalized}(\mathbf{S}^p \cdot (\alpha^{-1}\mathbf{1} - \mathbf{F}^2) \cdot G_{3D} * (\mathbf{S}^p \cdot \mathbf{X}^k) - \mathbf{S}^p \cdot G_{3D} * (\mathbf{F}^2 \cdot \mathbf{S}^p \cdot \mathbf{X}^k) + 2\mathbf{S}^p \cdot \mathbf{F} \cdot G_{3D} * (\mathbf{F} \cdot \mathbf{S}^p \cdot \mathbf{X}^k)), \tag{31}$$

where $*$ is a 3D convolution with Gaussian filter G_{3D} over space-time volume, $\cdot$ is an element-wise multiplication, $\mathbf{S}$ and $\mathbf{F}$ are unary and pairwise terms in matrix form with p and α controlling their importance, k is the current spectral iteration and $\mathbf{X}, \mathbf{S}, \mathbf{F}$ matrices have the original video shape ($N_{frames} \times H \times W$).

MULTI-CHANNEL LEARNING FORMULATION. As we detailed in Chapter 4, SFSeg++ extends the single-channel formulation in SFSeg such that it can learn how to combine several input channels, S_i and F_i, for unary and pairwise terms respectively: $\mathbf{S}_m \leftarrow \sigma(\sum_{i=1}^{N_{cs}} w_{s,i}\mathbf{S}_i + b_s\mathbf{1})$, $\mathbf{F}_m \leftarrow \sigma(\sum_{i=1}^{N_{cf}} w_{f,i}\mathbf{F}_i + b_f\mathbf{1})$, where $\mathbf{S}_m$ and $\mathbf{F}_m$ are the multi-channel unary and pairwise maps, respectively, σ is the sigmoid function, N_{cs} and N_{cf} are the number of input channels, $\mathbf{1}$ is an all-one matrix for the bias terms and $w_{s,i}, w_{f,i}, b_s, b_f$ are their corresponding learnable weights. We replace $\mathbf{S}$ and $\mathbf{F}$ in Eq. 31 with their multi-channel versions $\mathbf{S}_m$ and $\mathbf{F}_m$, respectively. We learn $w_{s,i}, w_{f,i}, b_s, b_f$ parameters both over segmentation and tracking tasks. More, SFTrack++ can learn end-to-end, from the original input frames all the way to final output, in the case of end-to-end learnable feature extractors.

5.5 PROTOCOL FOR EXPERIMENTS

Pre-register workshop at NeurIPS 2021

This work was published at the Pre-registration workshop at NeurIPS 2021, where the format was very interesting. You need to propose an experimental protocol supporting an technical sound and relevant idea for the field. Than it has to be validated by a classic double blind peer-review process with a

rebuttal phase, and next you have 4 months to do the proposed experiments and submit all the results. For emphasising the need for this kind of review in the community, focused on the idea and a legit plan for validating it rather than on the state-of-the-art numeric results, I present the work in the same format, following its temporal and natural logic.

PROTOCOL We test if SFTrack++ brings in a complementary dimension to tracking by having an intermediary fine-grained representation, extracted over multiple state-of-the-art trackers' outputs, and smoothed in space and time. We guide our experiments such that we evaluate the least expensive pathways first. For reducing the hyper-parameters search burden, we use AdamW [25], with a scheduler policy that reduces the learning rate on a plateau. For efficiency and compactness, we use the same channels to construct both the unary and pairwise maps: $\mathbf{S}_i = \mathbf{F}_i$. Their learned weights are also shared $w_{s,i} = w_{f,i}$. We use as input channels bboxes extracted with top single object trackers. We choose 10 top trackers: SiamR-CNN [164], LTMU [27], KYS [10], PrDiMP [33], ATOM [30], Ocean [198], D3S [105], SiamFC++ [181], SiamRPN++ [93], SiamBAN [20], which differ in architecture, training sets and overall in their approaches, but all achieves top results on tracking benchmarks.

TRAINING In Phase 1 we train our $NN_{feat2seg}$ network on DAVIS-2017 [133] and Youtube-VIS [184] trainsets, for each individual object. It receives the current RGB and the output of a tracking method (bbox), randomly sampled at training time. We use the U-Net architecture, validating the right number of parameters (100K - 1 mil) and the number of layers. We use DAVIS-2017 and Youtube-VIS evaluation sets to stop the training. Following the curriculum learning approach, before introducing tracking methods intro the pipeline, we use at the beginning of the tracking GT bboxes (extracted from segmentation GT, as straight bboxes). This allows the $NN_{feat2seg}$ component to get a good initialization, before introducing faulty bbox extractors, namely the top 10 tracking methods mentioned before. For Phase 2, we also train for the segmentation task. We learn the second part of our method to have an intermediary fine-grained representation, extracted over multiple channels, and smoothed in space and time. We validate here N_{iters}, the number of spectral iterations (1-5). We train on DAVIS-2016 [130] and Youtube-VIS datasets. In Phase 3, training for tracking, we learn a regression network, $NN_{seg2box}$ (with 50K-500K parameters), to transform the final segmentation to bbox. We train on TrackingNet [120], LaSOT [42] and GOT-10k [68] training splits.

BASELINES COMPARISON Experiment for comparing with other methods focus on the improvements SFTrack++ could bring over state-of-the-art and other competitive approaches for general object tracking: single method state-of-the-art solutions, a basic ensemble over the trackers, SFTrack++ applied only over the best tracker, SFTrack++ applied over the basic ensemble and the best learned neural net ensemble we could get out of several configurations (2D and 3D versions for U-Net [138] and shallow nets, having a different number of parameters: 100K, 500K, 1 mil, 5 mil, 15 mil). All methods receive the same input from top 10 trackers and train on TrackingNet, LaSOT and GOT-10k train sets as previously described. We evaluate our solution against all baselines on seven tracking benchmarks: **VOT2018** [81], **LaSOT**, **TrackingNet**, **GOT-10k**, **NFS** [49], **OTB-100** [179] and **UAV123** [118]. For the main conclusion of this chapter, we will provide statistical results (mean and variance over several runs) to better indicate a strong positive/negative result, or an inconclusive one.

ABLATIVE STUDIES We vary several components of our end-to-end model to better understand their role and power. We train our **Phase 1** component, $\mathrm{NN}_{feat2seg}$ net, not only for bbox input features but also for other earlier features, extracted from each tracker architecture. We test the overall tracking performance for this case. We remove from the pipeline the spectral refinement in **Phase 2** and report the results. We test the performance of our tracker without the **Phase 3** neural net, $\mathrm{NN}_{seg2box}$, by replacing it with a straight box and rotated box extractors from OpenCV [12]. We test several losses to optimize for both segmentation and tracking tasks: a linear combination between the weighted diceloss [151] and binary cross-entropy, Focal-Tversky [1], and Focal-Dice [170]. For the ablative experiments, we evaluate only on OTB100, UAV123, and NFS30 tracking datasets.

5.6 EXPERIMENTAL RESULTS

COMPARISON WITH OTHER METHODS In Tab. 12 we present the results of our method on five tracking benchmarks: OTB100, UAV123, NFS30, GOT-10k, and TrackingNet, comparing it with other top single methods and ensemble solutions. For single methods, we take into account each input method in our SFTrack++: D3S, SiamBAN, ATOM-18, SiamRPN++, PrDimp-18. We chose only one lightweight configuration per tracker, common across all benchmarks.

	Method	OTB	UAV	NFS	GOT-10k			TrackingNet		
		AUC	AUC	AUC	AO	SR_{50}	SR_{75}	Prec	$Prec_{norm}$	AUC
Single Method	D3S [105]	57.7	45.0	38.6	39.3	39.0	10.1	52.2	67.9	52.4
	SiamBAN [20]	67.6	60.8	54.2	54.6	64.6	40.5	68.4	79.5	72.0
	ATOM-18 [30]	66.7	64.3	58.4	55.0	62.6	39.6	64.8	77.1	70.3
	SiamRPN++ [93]	65.0	**65.0**	50.0	51.7	61.5	32.5	**69.3**	80.0	73.0
	PrDimp-18 [33]	**67.6**	63.5	**62.6**	**60.8**	**71.0**	**50.3**	69.1	**80.3**	**75.0**
Ensemble	Basic (median)	66.6	60.8	55.5	54.7	63.9	31.6	69.0	80.0	73.9
	Neural Net	**71.3**	59.7	58.2	59.5	69.8	42.9	70.6	80.2	74.5
	SFTrack++	70.3	**61.2**	**62.4**	**62.0**	**73.3**	**47.8**	**71.9**	**81.9**	**76.1**
	std	± 0.5	± 0.2	± 0.1	± 0.7	± 0.5	± 1.1	± 0.3	± 0.3	± 1.0

Table 12: Comparison on 5 tracking benchmarks. In the first group we show individual methods, used as input for our SFTrack++. In the second one, we show ensemble methods: a basic (median) and a neural net model with a similar number of parameters like SFTrack++. Our method outperforms both the input or other ensemble methods by a large margin on the challenging benchmarks GOT-10k and TrackingNet, while obtaining competitive results on OTB, UAV, and NFS. For SFTrack++ we report mean and std when training the model from scratch three times. With red we represent the best single method in the column and with blue the best ensemble. The raw results are available in the supplementary material.

For ensembles we use 1) a basic per-pixel median ensemble followed by the same bounding box regressor used in all experiments (see Sec. 5.6.1) and we also trained a more complex one: 2) a neural net having an UNet architecture (with 5 down-scaling and 5 up-scaling layers) and a similar number of parameters like SFTrack++ ($\approx$ 4.3 millions). We observe that the variation across different runs (including training from scratch) of our method is very small, showing a robust result and a clear conclusion. On newer, larger, and more generic datasets like GOT-10k and TrackingNet, our method surpasses others by a large margin, while on OTB100, UAV123, and NFS30 it has competitive results.

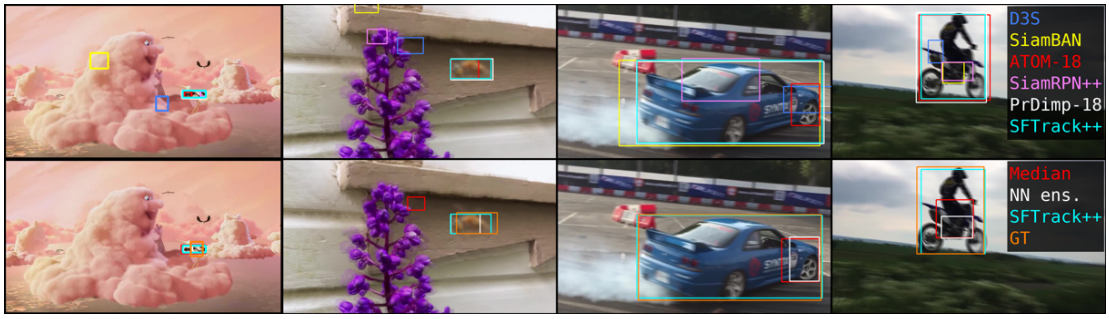

Figure 26: Qualitative results. We compare in the first line SFTrack++ with the input from individual methods. In the second line, we show the ground truth (in orange) and ensemble methods results that receive the same input as SFTrack++. We notice that even though the other ensembles fail to find a good bounding box, SFTrack++ manages to combine the input methods better, even in cases with a high variance among input methods.

ABLATION STUDIES To validate the components of our method, we test in Tab. 13 different variations, reporting results on OTB100, UAV123, and NFS30. First, we remove the spectral refining component from phase 2, taking out the temporal dependency and leaving the per frame predictions independent. In the next experiment, we remove the neural net from phase3 $\text{NN}_{\text{segm2bbox}}$. In the next chunk, we investigate the number of input methods. Last, we vary the number of spectral iterations from phase 2. The conclusions from this wide ablation are the following: 1) the spectral refining component is very important, emphasizing the initial intuition that preserving the object consistency in space and time using our proposed spectral approach improves the overall performance in tracking. 2) The quality of the input in our SFTrack++ method is important, but the more methods we use, the better. 3) We obtain better results using only one single spectral iteration. We tried the loss functions mentioned in the protocol, but we did not see relevant variations in IoU for the validation set, so we settle to BCE.

QUALITATIVE RESULTS Since SFTrack++ is an ensemble method, we show in Fig. 26 difficult cases and how it compares with individual methods and with other ensembles, starting from the same input (namely, all single methods from the first line, as explained in Sec. 5.6). We see how our method outperforms the others, even in those hard cases where an agreement seems hard to achieve.

SFTrack++ variations	**OTB**	**UAV**	**NFS**	**OTB+UAV+NFS**
w/o Spectral Refinement (phase two)	**71.6**	60.5	60.9	64.0
w/o $NN_{segm2bbox}$ (phase three)	65.5	57.4	58.5	60.2
Median (over 5 methods) as input	70.8	60.8	60.0	63.7
Best method (PrDimp-18) as input	67.1	59.7	61.3	62.5
Top 3 methods as input	64.8	60.8	61.1	62.1
2 spectral iterations	70.3	60.9	61.8	64.1
3 spectral iterations	68.0	61.0	60.1	62.9
SFTrack++ (1 iter, 5 methods)	70.3	**61.2**	**62.4**	**64.5**

Table 13: Ablations on OTB100+UAV123+NFS30 benchmarks. In the first group we remove from the pipeline phase 2 and phase 3, respectively. The results show that both components are crucial for the method. Next, we vary the number of input methods (1 to 5) but also their quality (best or median). We see that even the quality of the input matters, using more methods as input improves the overall performance. In the last part, we validate the number of spectral iterations, a single iteration achieving the best score, which slowly degrades over more iterations.

5.6.1 Documented modifications

There were several aspects where we need to deviate from the original protocol. Those aspects did not affect the core proposal and were mainly motivated by making the experiments less expensive in terms of computational cost, as detailed below.

NUMBER OF BENCHMARKS AND INPUT TRACKERS For each input tracker method considered, we run it in advance on all benchmarks (on training, valid, and test splits) to generate pre-processed input. We also generate the ground-truth bounding box segmentations for all benchmarks. This speeds up our training, making the overall training and testing self-contained, independent w.r.t. the input methods' code. We drop out the VOT2018 benchmark because its evaluation protocol consists of running a tracker on sub-videos, therefore we should have run all the input trackers code online, and this would have been too time-consuming. We resized each frame to keep its aspect ratio, having its

maximum dimension of 480 pixels. For training, for each video in the tracking benchmarks, we used only a sample with 5 frames. The pre-processed data for one single tracker, on segmentation and tracking benchmarks, for training, valid, and testing splits takes ≈ 1 TB (without LaSOT). Since LaSOT has a very large number of frames, we decided to drop it out to make the experiment time manageable. We considered that if we test our proposal on 5 trackers and 5 benchmarks, our core concept of the proposal would not be affected and the results would be sufficiently general and conclusive. We chose the 5 trackers (out of the 10 in the proposal) that were the easiest to integrate with the PyTracking [28] framework.

$\text{NN}_{\text{feat2segm}}$ **MODULE.** Since the considered trackers were very different, we couldn't find a proper way to extract similar features from each tracker model and we decided to drop this ablation.

BOUNDING BOX REGRESSION For extracting the bounding box coordinates out of the segmentation mask we use in all our experiments the region proposal from scikit-learn [161], with a 0.75 threshold for binarization. We did not perform an ablation study on bounding box regression because this would not be our contribution and did not influence our core proposal, this solution for bounding box regression being good enough to emphasize what we followed in our approach.

5.7 CONCLUSION

The pre-registered hypotheses that SFTrack++, our spectral approach that improves the space-time consistency of an object, improves the tracking performance proved to be valid.

Our method is not choosing the best input method, but it learns how to combine all inputs towards a superior performance, not only w.r.t. each input, but also w.r.t. a basic and a learned ensemble solution. SFTrack++ is robust, with a very low variance when re-training the entire pipeline from scratch, as shown in Tab. 12. We also noticed that the output of our method is very bold and it does not depend on a carefully chosen threshold. In Tab. 13 we show the importance of integrating the spectral clustering module in our SFTrack++ algorithm.

As a side observation, SFTrack++ has a weak performance on very small tracked objects when compared with single methods (UAV videos), but when compared with other ensembles, it achieves top results. This might be due to the large variance in the chosen individual methods predictions for the small objects.

In conclusion, SFTrack++ pre-registered proposal hypothesis validates through the proposed experimental protocol, with clear positive results.

Next in the thesis I will go beyond tracking and segmentation, and I show that having multiple intermediate representation working towards a selection based consensus improves each of the representation. Having this in mind, we show in the next chapter that this is a valid hypothesis in 2D space. As future work, I propose a spatio-temporal model to integrate in the multi-task graph also temporal nodes, searching to achieve cross-task consensus over space-time dimensions, using only supervision from expert models and improve over them.

6 BEYOND SEGMENTATION AND TRACKING: TOWARDS CONSENSUS IN MULTI-TASK GRAPH OF EXPERTS

Babies learn with very little supervision by observing the surrounding world. They synchronize the feedback from all their senses and learn to maintain consistency and stability among their internal states. Such observations inspired recent works in multi-task and multi-modal learning, but existing methods rely on expensive manual supervision. In contrast, our proposed multi-task graph, with consensus shift learning, relies only on pseudo-labels provided by expert models. In our graph, every node represents a task, and every edge learns to transform one input node into another. Once initialized, the graph learns by itself on virtually any novel target domain. An adaptive selection mechanism finds consensus among multiple paths reaching a given node and establishes the pseudo-ground truth at that node. Such pseudo-labels, given by ensemble pathways in the graph, are used during the next learning iteration when single edges distill this distributed knowledge. We validate our key contributions experimentally and demonstrate strong performance on the Replica dataset, superior to the very few published methods on multi-task learning with minimal supervision.

6.1 ZOOMING OUT: GENERALIZE FROM TWO TO MULTIPLE TASKS

The previous chapters show the importance of the consensus in tracking, next to the natural, yet efficient, approach for better connecting the time and space when segmenting objects, iteratively reaching the agreement. Then we

E. Haller, E. Burceanu, M. Leordeanu, Self-Supervised Learning in Multi-Task Graphs through Iterative Consensus Shift, The British Machine Vision Conference (BMVC) 2021

emphasize the need of having the segmentation as an intermediate task in tracking. In this chapter, we dive more in the direction of having multiple tasks and shifting iteratively towards their consensus. The proposed solution here focuses only on the spatial aspect of the predictions, all consensual results being based on per-frame decisions. We plan to further develop this multi-task graph solution over time, as detailed in future work, Chapter 7. We analyzing this chapter following the key aspects introduced at the beginning of the thesis:

- **A. Space-time consistency**: Not yet. The multi-task graph solution in this chapter contains only per-frame tasks, without having the temporal dimension. We will approach the temporal dimension of the graph in future work.
- **B. The power of the consensus**: The participants and the process vary at multiple levels, bringing in a lot of advantages of combining independent aspects:
  - different architecture/complexity: simple/similar edges and complex experts
  - different training datasets/knowledge of the world/data distribution
  - different semantic level of the domains: simple edges vs complex segmentation
  - multiple-paths for consensus
- **C. Exploiting multiple intermediate representations**: This is the focus of this approach, exploiting multiple domains (13 different tasks), reaching a more powerful consensus.
- **D. A limited quantity of supervision**: This is also one major point of the approach, using only pre-trained experts as ground-truth, which we outperform through iterations both with individual models and in consensus. The approach is unsupervised from the destination task point of view.
- **E. Experts: Making use of existing models**: Figuring out an appropriate consensus algorithm based on selection allows us to use multiple experts. These experts contain knowledge from multiple datasets (diverse even from the task point of view) and in-depth knowledge from multiple research groups like the model architecture, training, optimization

tricks, and task formulation, gathered over a long period. We use them as initialization for the tasks' labels in the graph, overcoming their initial performance, over multiple iterations.

6.2 CONTEXT

Seeing the world from multiple perspectives and with different interpretations offers an invaluable source of information, as shown by works using various features extractors [6, 100, 58, 187] or multiple supervised tasks [26, 191]. While the multi-tasks approaches attain a more comprehensive and fundamentally better understanding of the visual scene than the single-task ones, they require a larger amount of labeled data for the same input.

Our proposal for satisfying the unlabeled data problem, different from current multi-modal [132, 129] and multi-task-graph approaches [92, 190], is to take advantage of the existing pre-trained experts, on many different tasks in the state-of-the-art literature and offer a principled way to learn unsupervised on novel target domains, by using their selective consensual knowledge within a graph neural structure. Each node in our graph represents a specific interpretation or task. Each graph edge is a neural net that transforms one task into another. Pseudo labels for a given task are obtained by combining all pathways' outputs reaching the corresponding node. Very different from the recent NGC model [92]), we connect all tasks with neural networks and do not need any labeled data on the target domain. The experts we start with can be pre-trained independently on any, potentially very different, datasets. Also, different from the recent [92, 190], we show that an intelligent consensus-finding selection procedure is required to create strong pseudo-labels from many unsupervised pathways that could lead to a significant improvement over the initial experts. The unsupervised, consensus-based learning process continues for several iterations of the graph, with experimentally provable improvements from one iteration to the next. Our **key contributions** that are theoretical, as well as experimental, are:

1. **We introduce iterative Consensus Shift (CShift)**, an algorithm for unsupervised multi-task learning on novel target domains, applied to neural graph models. CShift exploits, using an intelligent selection procedure, the consensual agreements among multiple graph pathways, where each path transforms differently an input task into another. These ensembles of pathways become powerful unsupervised teachers in the multi-task

	Learning on target domain	Uses ensem.	Ensemble as supervision	Selection in ensem.	Unsup. domain adapt.	Full graph
XTC [190]	sup.	✗	N/A	N/A	✗	✓
NGC [92]	semi-sup.	✓	✓	✗	✗	✗
CShift (ours)	unsup.	✓	✓	✓	✓	✓

Table 14: Main differences between our CShift algorithm and state-of-the-art methods for learning in Multi-Task Graphs.

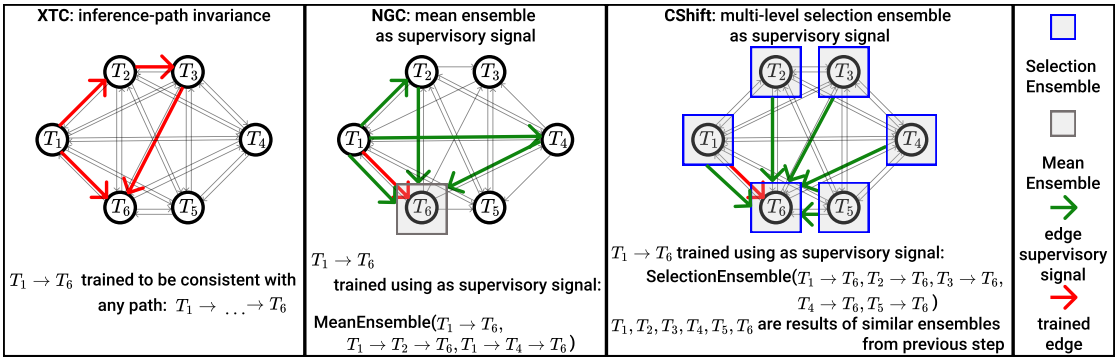

Figure 27: Illustration of the different training strategies employed by Multi-Task Graph methods in comparison to our approach (CShift).

graph, on novel target domains. The learning process continues over multiple graph iterations. After each iteration the pseudo-labels at every node, given by the consensual output at that node, shift, thus departing from and improving over the initial experts.

2. **We show in extensive experiments** on a recent multi-task dataset that CShift learns unsupervised to self-improve, over multiple tasks, from one iteration to the next, while also significantly outperforming the state-of-the-art experts used to generate the first pseudo-labels.

6.3 RELATION TO PRIOR WORK

Relation to Ensembles and Experts. The idea of different paths working together to reach a common goal was frequently demonstrated [139] over time.

Our solution is related to the idea of guiding the learning process using a set of expert models, an approach that proved effective for instance for tasks like video and image retrieval [38, 100, 115, 117].

RELATION TO UNSUPERVISED REPRESENTATION LEARNING. Recent formulations use pretext tasks [52, 195, 116, 194], clustering [17, 193, 158], minimize contrastive noise [57, 114, 155] or are based on generative adversarial models [36]. Different from these methods, we use as pseudo-ground truth the consensual output over multiple paths reaching the same task.

RELATION TO MULTI-MODAL LEARNING. Several recent works combine modalities and tasks. Different from the method in [132], using multiple modalities for predicting multiple tasks, we rely on the complex interactions between tasks' output domains, modeled as a bidirectional graph. Different from GDT [129] or GDT-related methods using multi-modal data transformation as self-supervision [154, 114], we learn complex interactions between tasks without ground truth, relying only on consensus and selection as supervision.

RELATION TO MULTI-TASK LEARNING. [191] studies the underlying connections between different tasks and proves that such relations can be exploited to effectively reduce the labeled data required for training. Different from XTC [190], we assume no multi-task annotated data on the target domain and rely solely on per-task expert models, pre-trained on different domains to provide the initial pseudo-labels. Moreover, the consensus shift method learns completely unsupervised on the target domain, relying solely on selection and consensus over multiple pathways reaching a given task node, over multiple graph iterations. From the architectural point of view, our model is related to the recent Neural Graph Consensus (NGC) model [92], which also connects multiple interpretations and tasks into a single graph of neural networks. Our model differs from NGC in four essential aspects: **1)** the proposed selection mechanism is highly adaptive (different for each pixel, sample, and iteration), allowing a dynamical adjustment of the graph structure as detailed in Sec. 6.4.2 and Fig. 29, compared with the simple average in NGC; **2)** our fully connected graph guides the learning process only based on unsupervised consensus, as opposed to having a fixed, pruned architecture based on supervised data (NGC); **3)** we use the ensemble labels both as a supervisory signal at a node and as input for all its out-edges, making learning more efficient; **4)** we initialize the graph with pseudo-labels generated by out-of-distribution experts, while NGC assumes a fully supervised initialization.

In Tab. 14 and Fig. 27 we summarize the main differences between CShift and the recent XTC [190] and NGC [92].

6.4 OUR APPROACH

We propose a novel Multi-Task Graph (Fig. 28-B) and the learning CShift algorithm, which uses as supervision the consensual output, extracted through an intelligent selection procedure, over multiple graph pathways that reach a given node. As mentioned previously, each node in the graph represents a task, a view, or an interpretation of the world. Each edge is a neural net that transforms one task at one node into another, from a different node. Our graph is directed and fully connected. In Fig. 28 we illustrate the main steps of our approach. All edges of our graph are initially trained using pseudo-labels for nodes, generated by out-of-domain experts. After initializing the edges (Fig. 28-A), we introduce **the view associated with a node**, computed as the ensemble result of all the node's in-edges. These views will become the pseudo-ground truth labels, during the subsequent learning iterations in the graph and constitute a major difference between our model and both the recent XTC and NGC models. The views (pseudo-labels), will shift from one learning iteration to the next, using the CShift learning algorithm, described next in more detail. CShift uses the views at each node, transmits information through the out-edges towards other nodes, and also collects the information from the in-edges, in order to create the new views, at the next iteration, by a selection mechanism that establishes the consensus over the multiple incoming edges. The process is repeated in principle until an equilibrium is found, at convergence. It should be clear at this point that the iterative graph learning phase is unsupervised and that the initial experts used could be created completely independently, using information from other datasets and domains, as our experiments will show. In Fig. 28-C we present a visual scheme of CShift, while in Alg. 5 we state its main formalized steps.

6.4.1 Multi-Task graph

We formally define the Multi-Task Graph over a set T of tasks (*e.g.* semantic segmentation, single-image depth estimation, surface normals - Sec. 6.5), each illustrating a different view of the scene. There are one-to-one correspondences between graph nodes and the set of tasks, and each edge is an encoder-decoder

Algorithm 5 - CShift: Multi-Task Graph Learning with Consensus Shift

X_i - input data sample i, all tasks $Y_{i;d}$ - pseudo-label for data sample i, task d
$Expert_d$ - expert for task d $e_{s\to d}$ - NN edge from task s to task d
n_{iters} - number of CShift iterations T - the set of all tasks
$part_{1..n_{iters}}$ - training dataset split, with n_{iters} parts, one for each iteration

Results: 1) CShift node views $Y_{i;d}$; 2) all trained edges $e_{s\to d}$

// Use the experts to generate the initial pseudo-labels
$Y_{i;d} \leftarrow Expert_d(X_i)\ \forall d \in T, \forall i \in part_1$
for $k \leftarrow 1$ to n_{iters} **do**
  for all $d \in T$ **do**
    // Update views using previous pseudo-labels
    $X_{i;d} \leftarrow Y_{i;d},\ \forall i \in part_k$
    // Learn from pseudo-labels
    train $e_{s\to d}\ \forall s \in T,\ \forall$ samples $\in \{(X_{i;s}, X_{i;d}) | \forall i \in part_k\}$
    for all $i \in part_{k+1}$ **do**
      // Compute neighbourhood for task d
      $\mathcal{N}(X_{i;d}) \leftarrow \{e_{s\to d}(X_{i;s}) | \forall s \in T\} \cup \{X_{i;d}\}$
      // Generate new pseudo-labels
      $Y_{i;d} \leftarrow f(\mathcal{N}(X_{i;d}), \mathbf{W}_{i;d})$

neural net transformation between source and target task nodes (Fig. 28-A). Consequently, our graph $G = (T, E)$ with $E = \{e_{s\to d} | e_{s\to d}(X_s) = X_d, s, d \in T, s \neq d\}$, where X_s is the scene representation under task s, and $e_{s\to d}$ is the neural network transforming the view between source task s and destination task d. The graph edges are initialized using pre-trained expert teachers, one for each considered task.

PASSING AN IMAGE THROUGH THE GRAPH: given a raw rgb frame, we associate it to the rgb node. Next, we aim to enforce the consensual constraint in the graph: no matter what path the input rgb takes through the graph, being transformed from one node to the next, it should have the same representation (view) at the same final node.

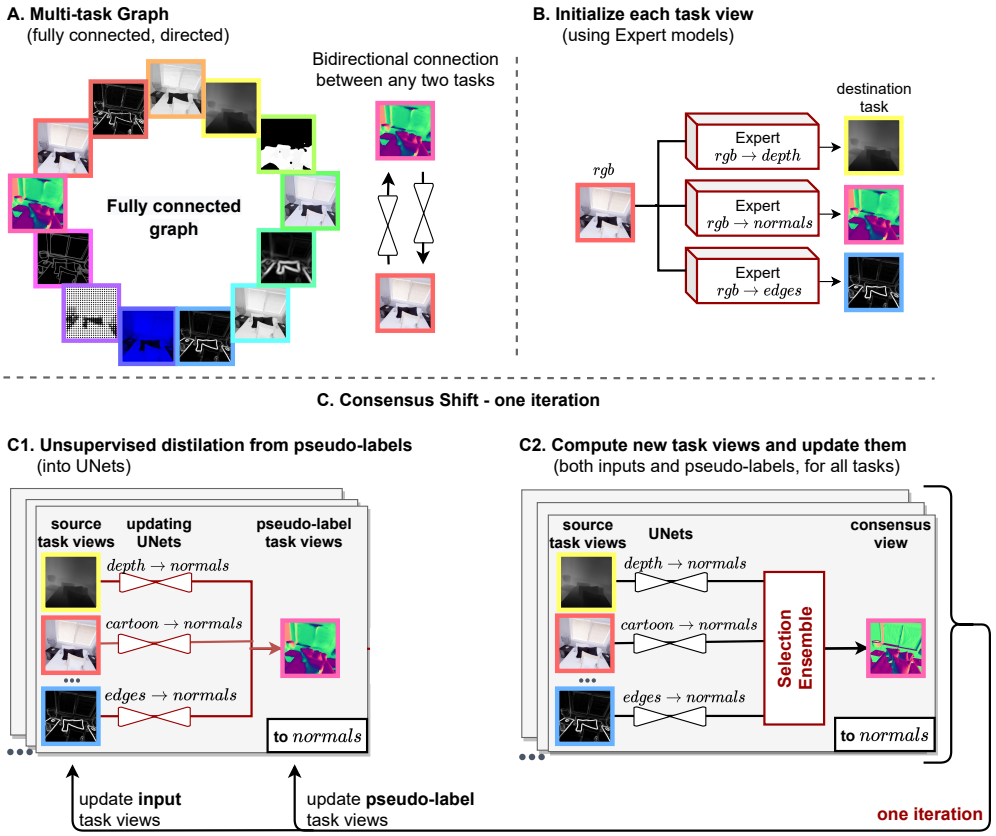

Figure 28: Main architectural elements of our Consensus Shift (**CShift**) system. **A.** Initialization from expert models. Based on the rgb image, the experts (which are considered as black boxes in our system) predict the initial pseudo-labels for each considered task. We emphasize that the experts are trained on different distributions than our target domain. **B.** Illustration of the fully-connected Multi-Task Graph, with 13 nodes corresponding to the 13 considered tasks. The edges are transformations between their source and destination tasks/nodes. **C.** The iterative process of CShift. Given the pseudo-labels for all tasks, we train each graph edge. Further, for each node, we compute its new pseudo-labels as the consensual representation of its in-edges, through the selection ensemble mechanism. The newly computed labels of a node become its supervisory signal in the next iteration. They also become inputs for all the out-edges of the node.

INITIALIZING THE GRAPH: Other approaches [92, 190] start with a supervised training phase of all the out-edges of the rgb node $\{e_{rgb \to d} | \forall d \in T\}$, requiring multi-task annotated datasets. As we work in the unsupervised regime for the

target domain, our initial views (pseudo-labels) for different tasks are obtained from a set of out-of-distribution expert models (Fig. 28-B). Each task node d has an associated expert: Expert_d. Then the initial edges are trained by distilling the knowledge of the experts (see the list of experts in Sec. 6.5).

6.4.2 Consensus Shift learning

The consensus between edges reaching a given task d provides a robust view for d. With each learning cycle, the node values (pseudo-labels) shift towards stronger consensus, following the CShift algorithm (Alg. 5). The new labels are then used to distill the single edges connecting them and thus set up the next learning stage. Each transformation is, in fact, the last step of a longer graph path, starting in the rgb node and ending in a destination node d. All paths should ideally be in consensual agreement, but in practice, they are not, so an intelligent mechanism is needed to extract the robust knowledge shared by the majority. We employ the discovery of consensus among multiple outputs from incoming edges. Intuitively, CShift is an adaptive combination (weighted median) over the output of all edges reaching a destination. It is based on a similarity measure between those views, computed at pixel-level using a distance function between pairs of views coming from different source tasks and using a kernel to smooth over the neighboring pixels.

Given a destination node d, all edges reaching this node $\{e_{s\to d} | s \in T, s \neq d\}$ are transformations of the source tasks. Considering a sample X_i, CShift iteratively updates the sample's view, associated with task d, $X_{i;d}$. We next define $\mathcal{N}(X_{i;d})$, the neighbourhood of $X_{i;d}$ as the set of all transformations from all different views of the sample, towards the destination, joined with the current pseudo-label:

$$\mathcal{N}(X_{i;d}) = \{e_{s\to d}(X_{i;s}) | \forall s \in T\} \cup \{X_{i;d}\}. \tag{32}$$

The current task representation is replaced by the consensual one, computed as a function f gathering information from all the neighbours of $X_{i;d}$, parameterized by pixel-level weights $\mathbf{W}_{i;d} \in \mathbb{R}^{h\times w\times |T|}$ (where (h, w) is the image size):

$$X_{i;d} \leftarrow f(\mathcal{N}(X_{i;d}); \mathbf{W}_{i;d}), \tag{33}$$

capturing the consensus between predictions. $\mathbf{W}_{i;d}$ has a channel associated with each task node, which indicates the similarity of the corresponding pre-

diction with the current value of d. For a location (x, y) and a given task s, the weights are computed as follows:

$$\mathbf{W}_{i;d}[x, y, s] = \frac{K(\mathrm{dist}(e_{s\to d}(X_{i;s}), X_{i;d})[x, y])}{\sum_{Z\in\mathcal{N}(X_{i;d})} K(\mathrm{dist}(Z, X_{i;d})[x, y])}, \tag{34}$$

where $\mathrm{dist} : \mathbb{R}^{h\times w} \to \mathbb{R}^{h\times w}$ is a distance function capturing the similarity between two different prediction maps and $K : \mathbb{R} \to \mathbb{R}$ is the kernel function that determines the weight of nearby points. The algorithm aims to identify the areas of the prediction maps that are perceptually similar and push them further in the ensemble while downgrading regions that seem to be noisy and uncorrelated with the other predictions. We propose a selection ensemble algorithm that automatically extracts the most representative consensual representation of the in-edges. The adaptive per-pixel weighting allows CShift to keep the most relevant information from all input maps, even when some maps are less reliable, treating similarities per region (kernels at pixel-level). In Sec. 6.5.1 we instantiate dist, K, f and provide ablations experiments proving that the selection strategy is robust to noisy connections. We show in Fig. 29 how the selection based consensus works at pixel-level.

6.4.3 Unsupervised domain adaptation

CShift requires no human-annotated data for the target domain. We take advantage of existing state-of-the-art expert models that distill research years and valuable expertise and provide reliable pseudo-labels for each of the considered tasks. When applied to novel domains, the weakness of these experts is that they are trained on different, out-of-domain distributions. We first transfer their knowledge in our graph edges. Then our learning method, by exploiting and enforcing the overall consensus among all tasks, allows the graph to adapt by itself to the target domain, thus overcoming the domain gap, as shown in our tests (Sec. 6.5.2 and Sec. 6.5.3). To emphasize the domain adaptation capabilities of CShift, we employ the Maximum Mean Discrepancy [55] (MMD) method for measuring the domain dissimilarity between our target domain and the expert source domains. MMD is a strong and widely used [78, 101, 182] non-parametric metric for comparing the distributions of two datasets. We follow the methodology in [55] and compute the unbiased empirical estimate of squared MMD. Our experiments show (Sec. 6.5.2) that there is a large distributional shift between our target domain and the domains of the original expert

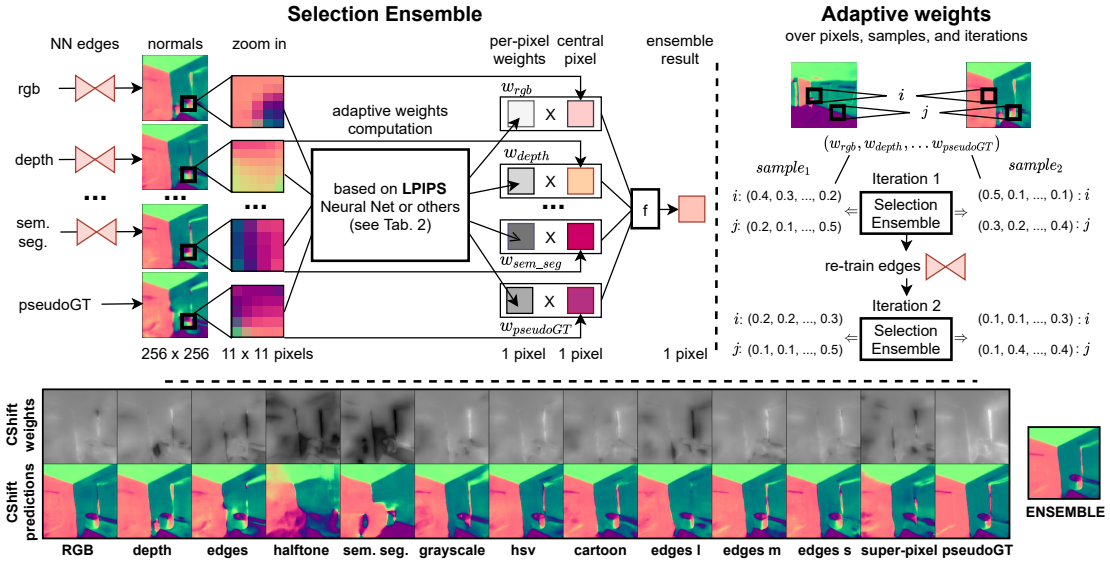

Figure 29: CShift selection flow, stripped down to pixel-level. The complex weights are highly adaptive, modifying, in effect, the underneath graph structure through their values. The last two rows show CShift per-pixel weights and the corresponding predictions from source tasks to normals task destination. Note how the sharp zones correlate with high ensemble weights.

models. This result, along with the results in Sec. 6.5 prove the unsupervised domain adaptation capabilities of our method.

6.5 EXPERIMENTAL ANALYSIS

Dataset. We perform experiments on Replica [177], a dataset of photo-realistic 3D indoor scenes, comprising 18 scenes, with a total of 48 rooms. In practice, we consider two iterations for training our Multi-Task Graph and use two unsupervised train sets (9600+9600 samples), two validation sets (960+960 samples), and a test set of 960 samples (never seen during the unsupervised learning process). During training and validation, only the raw rgb images are available. Ground truth is used only for evaluation on the test set.

EXPERT MODELS. We employ per-task expert models, with input rgb only, trained on out-of-domain data, to initialize the per-node pseudo-labels (Fig. 28-A). Conceptually, in the first learning iteration, the experts replace the direct edges starting from the rgb node. Then, the computed views of a node become

supervisory signals for the in-edges of that node and inputs for the out-edges (*e.g.* for training edge $e_{\texttt{depth}\rightarrow\texttt{halftone}}$, the depth input is obtained by applying the depth expert over the rgb frame and the halftone pseudo-label by extracting the halftone from rgb.)

Our graph contains a total of 13 task nodes, including the rgb one, thus we consider 12 experts ranging from trivial color-space transformations to heavily trained deep nets: **1)** halftone computed using python-halftone; **2)** grayscale and **3)** hsv computed with direct color-space transformations; **4)** depth and **5)** surface normals obtained from the XTC [190] experts; **6, 7, 8)** small, medium and large scale edges extracted using a Sobel-Feldman filter [43], and more complex **9)** edges extracted using the DexiNed [149] expert; **10)** super-pixel maps extracted using SpixelNet [183]; **11)** cartoonization got from WBCartoon [175] and **12)** semantic segmentation maps computed with HRNet [167]. The deep nets expert models are trained on a large variety of datasets: **4)** and **5)** Taskonomy [191], **9)** BIPED [150], **10)** SceneFlow [112] + BSDS500 [4], **11)** FFHQ [73], **12)** ADE20k [199]. Note that these datasets are built for a different purpose and has a different distribution than our domain.

We evaluate our model on two tasks: depth and surface normals predictions from rgb. The XTC experts' output on depth and normals tasks are not aligned with annotations from Replica as their original datasets use different conventions. Thus, on depth, following the methodology of self-supervised methods [200], we performed a histogram specification alignment between expert results and ground truth annotations. On normals, we removed the 3rd channel in the XTC expert according to Replica, which has normals with only 2 independent channels.

IMPLEMENTATION AND TRAINING DETAILS. Each graph edge is a neural network with a UNet architecture, as previously validated in NGC [92] and XTC [190]. All 156 graph edges have $\approx$ 4.3 million parameters, with 4 down-scaling and 4 up-scaling layers and a proper number of input and output channels, depending on the source and destination tasks. We optimize them by jointly minimizing L2 and the Structural Similarity Index Measure [176] (SSIM) losses for regression tasks equally-weighted for each neural net. For training edges going to classification tasks (semantic segmentation or halftone), we use Cross-Entropy loss. As optimizer we work with SGD with Nesterov (lr=5e-2, wd=1e-3, momentum=0.9), and a ReduceLROnPlateau scheduler (patience=10, factor=0.5, threshold=1e-2, min lr=5e-5). Models are trained for 100 epochs during the first iteration, with 9600 training samples. For the second iteration, we perform 100 additional training epochs on both train sets, reaching a total

Method	depth	normals	rgb
Expert [190]	14.58	8.30	-
Mean Ensemble [92]	12.94	7.95	4.30
CShift w/ Variance	12.80	7.91	2.12
CShift w/ PSNR	12.89	8.12	4.25
CShift w/ SSIM	12.80	7.89	2.38
CShift w/ L1	12.81	7.73	2.16
CShift w/ L2	12.79	7.72	2.45
CShift w/ LPIPS	**12.77**	**7.61**	**2.06**

Table 15: Ablation study on different distance metrics on Replica dataset, for the first iteration. In all considered configurations CShift overcomes the initial expert models and in all, except the PSNR case, the Mean Ensemble.

of 19200 training samples. Note that for the second iteration, we train all edges from scratch. For readability, all reported scores in tables are the L1 value $\times 100$.

6.5.1 Consensus Shift

ENSEMBLE SELECTION METHOD. At the core of the selection procedure is the distance function dictating each prediction map's per-pixel weights, as detailed in Sec. 6.4.2. In experiments, we instantiate f (Eq. 33) to the weighted median, and for the kernel function K (Eq. 34) we use the identity. We consider different distance metrics, ranging from local per-pixel distances to global perceptual measures, to understand the proposed selection strategy's power: **1)** L1 and **2)** L2 distances at pixel-level **3)** Peak signal-to-noise ratio (PSNR) to measure the noise of the predictions; **4)** Structure Similarity Index Measure (SSIM) [176] that analyses the luminance, contrast and structural differences; **5)** Learned Perceptual Image Patch Similarity (LPIPS) [196] that is a deep model trained to identify perceptually similar images. We also consider the **6)** per-pixel variance among the multi-path predictions, to quantify their consensus.

In Tab. 15 we present the results of CShift under different selection strategies, for the first iteration. We compare our method against the expert models used to generate the initial pseudo-labels and with a baseline mean ensemble

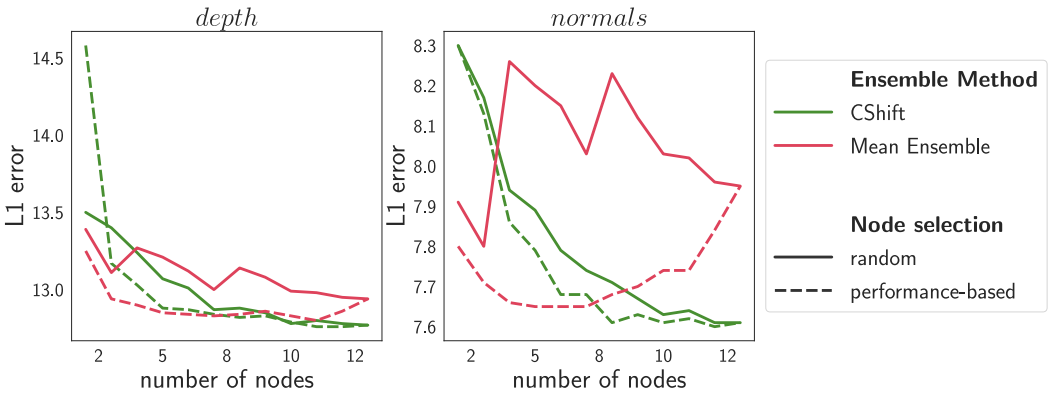

Figure 30: Our selection ensemble is stable under different node selection strategies. It proves its ability to extract relevant information even from low-performing edges, as the performance slightly increases with each new node and its corresponding edge, irrespective of the edge performance. For the mean ensemble, we observe an unstable evolution under the random node selection strategy and a performance decrease when low-performing edges are added to the ensemble in the case of performance-based selection.

model, which is similar to some extent to [92], as our models are only trained using the expert models while NGC[92] employs a supervised initialization step. Our proposed selection ensemble overcomes the expert models in all the considered configurations while overcoming the simple mean ensemble in all setups except for the PSNR configuration, a combination that is slightly worse for the normals task. CShift w/ LPIPS is our top-performing configuration, but all the other metrics prove reliable for the selection ensemble, highlighting its importance in the unsupervised learning process. For all the subsequent experiments, we use CShift w/ LPIPS as our default configuration. Besides depth and normals, we also report the results for the rgb task, measuring the graph model's ability to reconstruct its original raw input from the ensemble results of its many paths.

CONSENSUS UNDER DIFFERENT SETS OF NODES. To validate that our model is robust to the set of the considered nodes (and their corresponding in and out edges), we perform an experiment where we start with a small graph containing only two nodes, and, step-by-step, increase the number of nodes until reaching all 13 nodes. The nodes are added to the graph in a specific order. We analyze two ways of establishing this order: 1) random - nodes are randomly

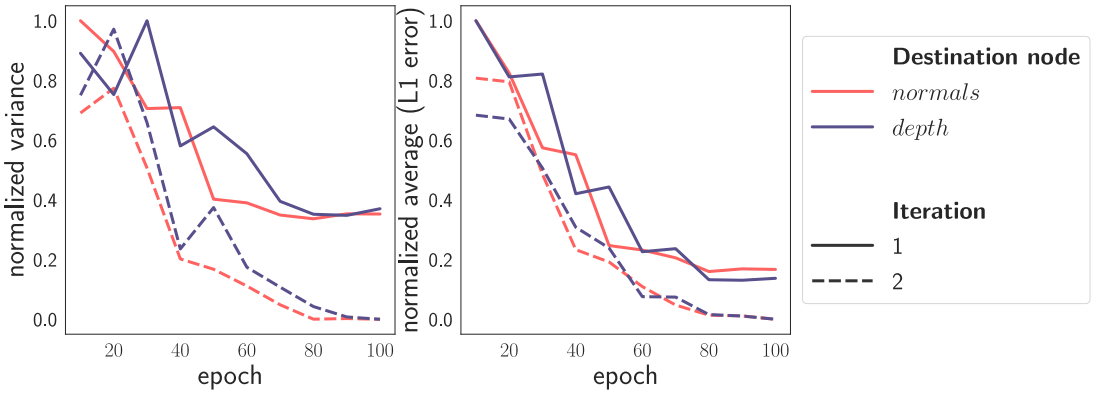

Figure 31: The first plot shows that the average variance over single edges in an ensemble decreases over the training epochs and iterations. Thus, the graph's average consensus improves from one learning iteration to the next, while the average L1 error of edges decreases (second plot). This proves the CShift approach's effectiveness and validates that the edges evolve towards the ground truth rather than collapsing in trivial solutions.

sorted; 2) performance-based - nodes are sorted according to their individual performance (evaluated w.r.t. to ground truth annotations) and added in this order. In Fig. 30 we present the results of our experiment, comparing CShift with a mean ensemble baseline, for two destination tasks: depth and normals. In both scenarios, the performance of CShift increases with the number of nodes, proving that our ensemble selection mechanism is able to extract relevant information even from low-performing edges. We significantly overpass the baseline in all configurations. We highlight the performance fluctuations of this baseline that seems to be highly dependent on the performance of the individual edges reaching the ensemble. Our solution is stable even under a random node selection strategy.

EDGES IMPROVEMENT BETWEEN ITERATIONS. We give an in-depth analysis of how individual edges evolve over training epochs and CShift iterations. First, in Fig. 31-a, we see how the variance between the edges in an ensemble (reaching depth or normals task) decreases with more training. This is natural since each edge in the ensemble uses the same pseudo-ground truth annotations. But, in the second iteration, the variance is even smaller, showing a smoother training optimization for the edges (which are trained from scratch for this second iteration). This could be explained by the new pseudo-labels

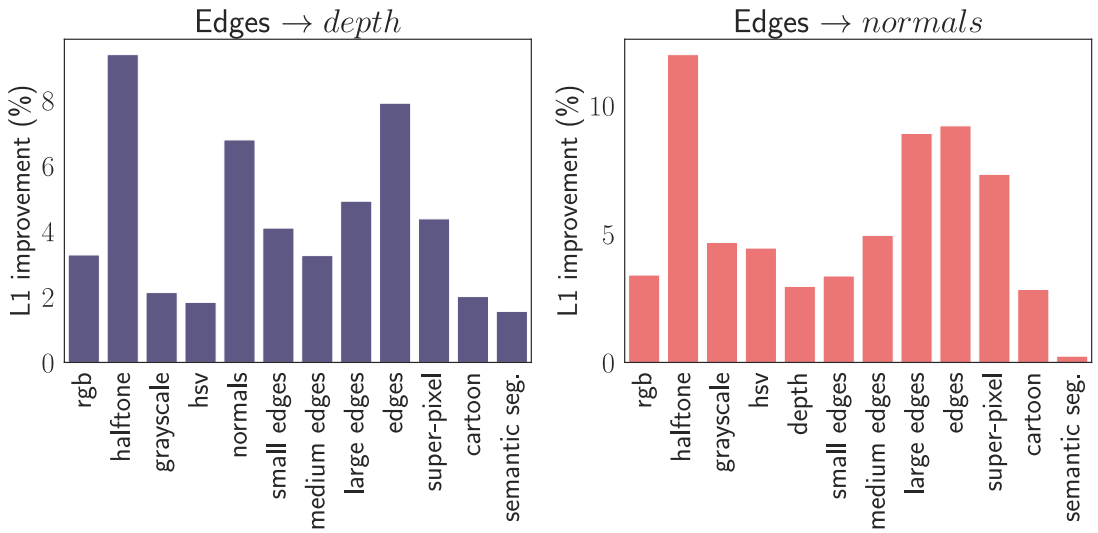

Figure 32: We present the relative performance improvement of the individual edges between iterations. The performance of each edge increases up to almost 12%, proving the capacity of CShift to iteratively adapt to the new domain, in an unsupervised manner.

coming from iteration 1 ensembles, rather than experts, making the training process simpler. Next, to validate that the edges do not collapse to a bad representation, we also plot the average L1 error in Fig. 31-b, confirming that all the edges improve their performance towards the ground truth. In Fig. 32 we show those relative improvements per edge, between the two CShift iterations.

6.5.2 Domain adaptation

Here we look closer at the gap between the input distribution on which the `depth` and `normals` experts were trained on, and the one for our training and testing dataset, Replica. The experts for these two tasks [190] are trained on the Taskonomy dataset, a real-world dataset. Replica is a synthetic one, and to validate our experiments, we add another synthetic dataset to the comparison: Hypersim [137]. We compute the discrepancy in distribution between Replica and those datasets, using MMD as described in Sec. 6.4.3. The analysis was performed both for the input level and the expert's mid-level features. For computing MMD, we average over multiple runs, each containing 100-1600 samples per dataset. The results in Tab. 16 show that there is a significant

	rgb	depth	normals
MMD(replica$_1$, replica$_2$)	5.4	17.8	17.4
MMD(replica$_1$, hypersim)	3.4	20.1	20.6
MMD(replica$_1$, taskonomy)	13.1	23.3	20.2

Table 16: We report the MMD between our target domain (Replica dataset) and the domains of the depth and normals expert models (Taskonomy dataset), considering both rgb input and mid-level embeddings of the experts. Compared to another synthetic dataset (Hypersim), we observe a smaller distribution shift than for Taskonomy, which contains real-world samples. We also validate our assumptions by comparing two different splits of Replica. For readability, we report MMD $\times 100$.

domain shift in the input for the pre-trained experts on Taskonomy, both at the rgb level but also through the eyes of the experts (depth and normals columns). Notice that the Hypersim dataset is closer to Replica (compared with Taskonomy) since both use synthetic data.

6.5.3 Comparisons with the experts

QUALITATIVE VIEWS FOR MULTIPLE TASKS. In Fig. 33, we show the differences between the expert output, used as initial pseudo-ground truth for our graph edges (Fig. 28-A), and the output of our CShift algorithm. Notice that the output of CShift looks smoother and partially corrects the mistakes of the expert model, adding significant value in the output.

QUALITATIVE RESULTS FOR TASKS WITH GT. We compare next in Fig. 34 our results with the Experts, with respect to ground-truth. In the last column, we show with green pixels where CShift outperform the Expert and with red the ones where CShift is weaker. Notice that CShift improves the output at a profound level, bringing in new information in the scene (see the bike in the second row or even the walls in the third). This is due to the multiple different source tasks for the ensembles' in-edges.

COMPARISON WITH OTHER METHODS. In Tab. 17 we provide a quantitative analysis of our method. Starting from pseudo-labels provided by the experts (depth and normals), we improve their quality by a large margin both with

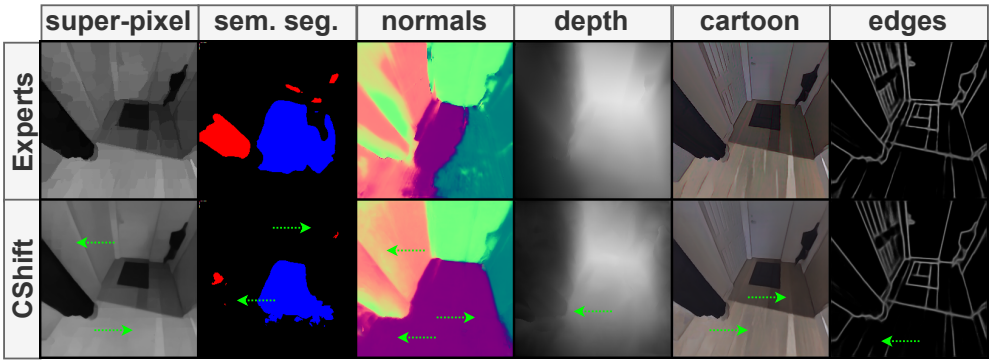

Figure 33: Qualitative results on multiple tasks. On the first row, we show the experts' output, used as pseudo-labels for training our edges. On the 2nd line we show the output of our CShift. Green arrows point to places where CShift adds significant value in the prediction. From left to right, we see how CShift improves the representation for multiple tasks: super-pixel output has less noise; semantic segmentation removes almost all pixels wrongly classified as ceiling ones; surface normals are significantly corrected; depth catches new details from the curtain; cartoonization output is also less noisy, and on edge detection task it removes some of the noisy edges coming from the floor texture.

an ensemble and with a direct link from rgb (except for Hypersim's normals where GT is extremely detailed, while our single edge is a simple UNet with 4.3 mil params). We achieve this performance over two CShift iterations, without adding any supervised information and we also largely outperform the basic mean ensemble. Also, notice that the direct edges in our graph significantly improves over CShift iterations (in average, and individually, the direct edge from rgb). Since we treat all tasks unitary, we also train edges with rgb as a destination. Interestingly, the rgb reconstruction performance improves over iterations.

6.6 CONCLUDING REMARKS

We present the consensus shift (CShift) algorithm in multi-task graphs, able to learn unsupervised in novel domains, using as supervision the selective consensus, among the multiple pathways reaching a given task node. The unsupervised capability, fully connected structure, powerful ensemble selection

	Method	Replica $d_{task} \downarrow$			Hypersim $d_{task} \downarrow$		
		depth	normals	rgb	depth	normals	rgb
	Expert [190]	14.58	8.30	-	15.11	**9.10**	-
Iter 1	Average of direct edges	14.32	9.34	6.33	16.91	12.55	11.34
Iter 1	Edge: rgb $\rightarrow$ d_{task}	13.42	8.23	-	15.97	11.75	-
Iter 1	Mean Ensemble [92]	12.94	7.95	4.30	14.84	10.56	8.22
Iter 1	**CShift**	12.77	**7.61**	2.06	13.98	9.36	3.56
Iter 2	Average of direct edges	13.70	8.83	5.00	15.74	11.37	9.01
Iter 2	Edge: rgb $\rightarrow$ d_{task}	**12.98**	**7.95**	-	**15.03**	10.09	-
Iter 2	Mean Ensemble	12.87	7.91	3.18	14.20	9.90	6.31
Iter 2	**CShift**	**12.71**	**7.61**	**1.51**	**13.75**	**9.02**	**1.84**
	CShift Boost	**12.8%**	**8.3%**	-	**9.0%**	**0.9%**	-

Table 17: Quantitative results. We compare the L1 error of our method over each iteration against the initial experts, on the destination tasks for which we have GT annotations. CShift ensemble outperforms XTC experts on depth and normals by a large margin, without any additional supervision. Even single, direct edges (rgb $\rightarrow$ d_{task}) improve over iterations, achieving better results compared with the experts in most of the cases (except for Hypersim's normals). With blue we represent the best single edge in the column and with red the best ensemble. For readability, we show the L1 error $\times 100$.

for creating pseudo-labels and then shifting them from one graph learning iteration to the next, make our approach significantly different from related ones. All key aspects of our approach (choice of ensemble selection, multi-task structure, and domain adaptation capabilities) are experimentally validated, while the comparisons to related works prove superior capabilities in the unsupervised learning case. We believe that CShift brings theoretically interesting and practically valuable contributions in an area of research, that of multi-task unsupervised learning, which is of major importance, but not sufficiently studied.

I propose next a model that integrates the findings from this chapter not only under the spatial constraint, frame-by-frame, but also including the tem-

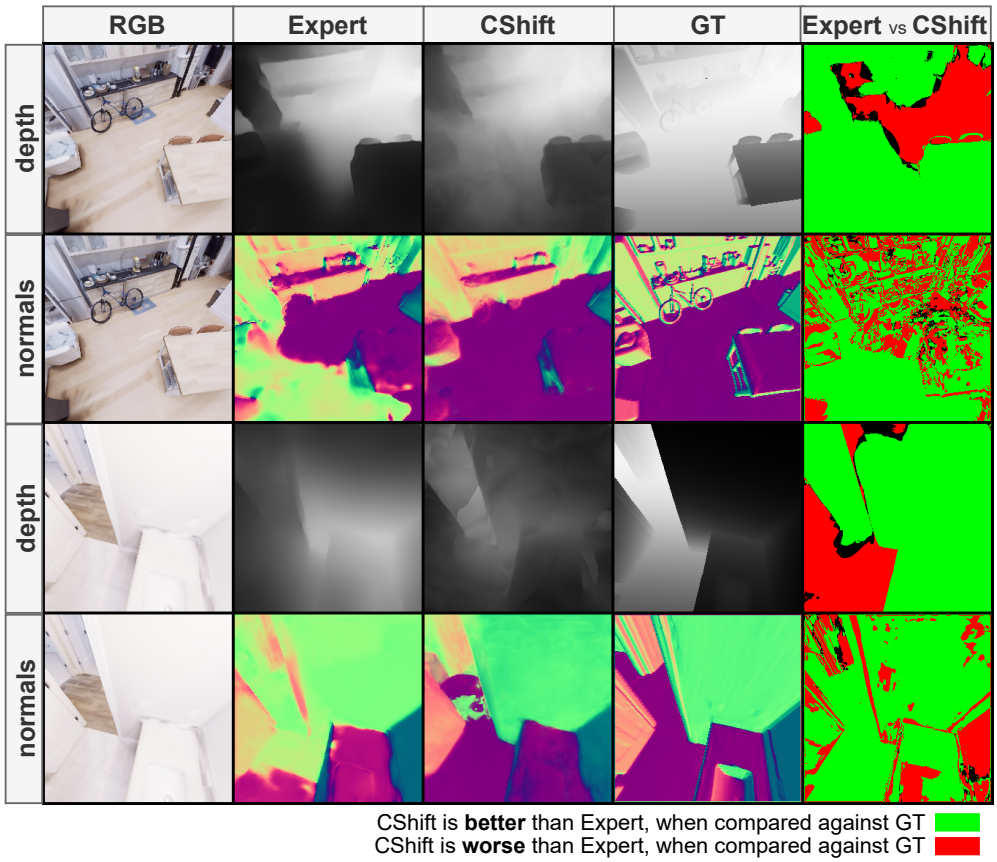

Figure 34: Qualitative results on depth and surface normals, for which we have ground truth annotations. We compare the Expert (col 2) with CShift (col 3). In the last column, we visually show the differences. With green are pixels for which the prediction is improved and with red the ones where the CShift prediction is weaker. On the second line, we see how CShift adds a bike in the scene, where initially in the Expert it is completely missing. This shows that the tasks are interconnected and that CShift takes advantage of their intrinsic links in an unsupervised manner, improving the results not only numerically, but also bringing in new fundamental information extracted from the consensus of all tasks.

poral dimension. This way, the video object segmentation and tracking nodes are integrated in the multi-task graph, and more, every task, in each frame access the temporal information in order to provide better and more robust representations.

7 FUTURE WORK

During my PhD work, I tried to understand the complexity of an object in a video, focusing on its natural consistency in both space and time. I investigate those aspects in the context of object tracking and segmentation tasks. I start with a tracker solution composed of many diverse parts. We introduced an algorithm to coordinate them to function together as a society, each part's role being monitored and evolving over time. For making it possible, we observe and prove a mathematical property in our formulation that allows us to learn many linear classifiers with only one closed-form equation. We observe that one main limitation of our solution is linked to the formal definition of tracking: the bounding boxes introduce a lot of noise. So combining this limitation with our direction of better integrating time and space, we come with an approach for object segmentation in video using spectral clustering in the space-time volume of pixels. Our contribution here is finding a good approximation based on Taylor expansion for power iterating in the video volume, proved both theoretically and empirically. We claim that the main cluster in space-time volume of the video is in fact the segmentation of the main object. So by finding a better and more refined cluster we improve our initial segmentation. We expand this solution to learn an ensemble over multiple inputs seen as multiple-channels for power iteration. Our next contribution is using spectral clustering in space-time on an intermediary task (video object segmentation) significantly improves the tracking results.

7.1 MULTI-TASK GRAPH OF EXPERTS IN SPACE-TIME

After analyzing in-depth and proving the importance for each part of the ensemble in STP, Chapter 3, we postulate the importance of a more refined representation rather than the rough bounding box when tracking objects. So we approached the segmentation task in SFSeg++, Chapter 4, and next integrate the segmentation as an intermediary output in the tracking pipeline resulting SFTrack++, Chapter 5.

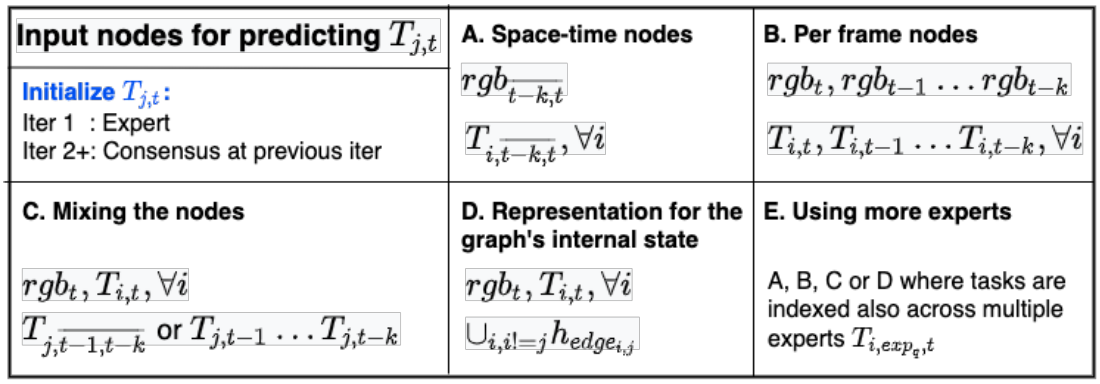

Figure 35: Consensus Shift over spatial and temporal nodes.

Next, the intuition was that if one single middle task was useful, having more of them might bring additional information and value to the final prediction. We successfully tested this hypothesis in CShift, Chapter 6 with image tasks. So a natural step further is contouring the findings from CShift and expanding the approach to time, passing from image-level to video. This is important because it will allow us to add more complex tasks (like video tracking and segmentation) and because the edges will also capture temporal dependencies, the graph being able to see more complex relations between the nodes.

7.1.1 Graph architecture baselines

There are multiple ways in which we can introduce the temporal dimension. The biggest problem we need to consider is the amount of computation required to train the graph edges from the fully connected graph. In CShift, the spatial only version of the graph, for 13 nodes, had 156 edges to train ($13 * 12$). There is great value in having an automatic selection-based algorithm for consensus, because we don't need to prune the graph edges by hand using prior knowledge. So the spatio-temporal CShift will also have this characteristic. I present next various directions to evaluate in a thorough ablation and summarize them in Fig. 35.

A. SPACE-TIME NODES The first obvious approach for expanding to time is to transform the nodes. The input represents now multiple frames instead of a single one. Those can be closed to the current moment in time (*e.g.* $t-1, t-2$, but there can also be a more distant sample $t-p$ frame, maybe a more confident one that would bring robustness in the formulation. This is by far the

fittest solution from the number of edges point of view. For N tasks, we have only $N * (N - 1)$ edges, increasing the number of parameters per edge (from one map to k maps as input, where k is the number of frames in the volume). A disadvantage of this approach could be that 3D convolutions do not capture very well the temporal aspect.

B. PER FRAME NODES In this configuration, we keep one node per frame and multiple frames. So instead of a volume with k frames from the previous approach, here we have k nodes, each with connections with all other tasks. So for predicting $T_{j,t}$ we need to learn the connections from $T_{i,t}$, $T_{i,t-1}$, ... $T_{i,t-k}$, rgb_t, rgb_{t-1}, ... rgb_{t-k}, where k is the number of frames in the past which we keep as neighbours for the current one. So we have $N * (N - 1) * k$ number of edges for this approach. This architecture certainly will not scale for a large number of edges and neighbour frames, but would give us an important clue whether this is a better approach from the architecture point of view when compared with the rest.

C. MIXING THE NODES In this variant, we plan to use temporal information only for the current task, and for the rest we will use only the current frame. Depending on the results, this might prove to be a fair compromise between having many connections and a good performance in the spatio-temporal graph.

D. REPRESENTATION FOR THE GRAPH'S INTERNAL STATE Other architecture variation would keep an internal representation of the previous frame for the entire graph. With every new iteration, we take into account one step longer paths in the graph. To summarize the information from the previous frame, we could use a hidden state for each edge and concatenate them in a multi-channel map. Then we bring this map back as input for each edge. This approach doubles the total number of trainable edges in the graph ($N * (N - 1) * 2$).

E. USING MORE EXPERTS Since there are multiple experts available for a particular task, we can consider a node to be uniquely determined by an expert-task combination. In consequence, for each fixed source task, given a destination task, we will learn one edge for each expert (as opposed to one edge per task). Of course, this solution is costly from the resources point of view, but it is useful to evaluate its performance. This will test if adding more experts in the graph (for the same task) gives the model room for improvement.

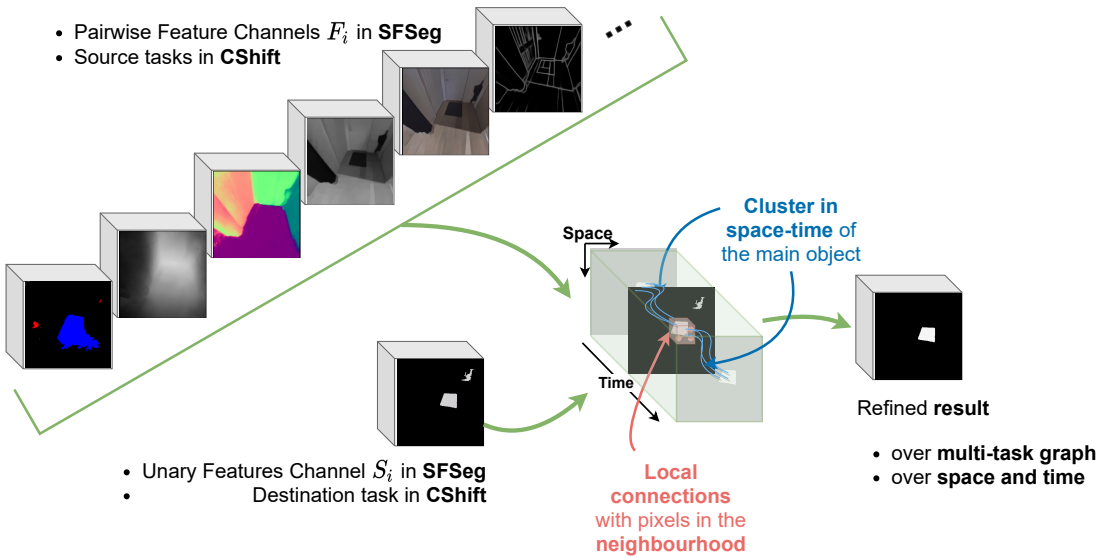

Figure 36: Combining CShift and SFSeg for space-time consistency, where tracking or segmentation is the destination task.

7.1.2 Combining CShift with SFSeg

I propose next a way to take advantage of multi-task graph formulation advantages from CShift and SFSeg space-time consistency in object tracking and segmentation tasks context. In SFSeg, we use unary and pairwise terms, like presented in Chapter 4, Eq. 17. For unary terms, we always need to consider a clustering representation (like outputs from segmentation and tracking), where in each pixel we have a probability for that pixel to be part of the target object cluster. Next, for pairwise terms, we can use any representation that helps us better quantify the difference between any two pixels in the representation. We used those channels in SFSeg, SFSeg++ and SFTrack++ approaches maps with segmentation or tracking output, from one or multiple solutions, but having the same meaning as the unary term. In theory, there is no restriction for this other than using normalized distances, but we never explored this path. So I propose analyzing this aspect in order to take advantage of multiple domain representation in the consistent spatial-temporal SFSeg. I would start by using for pairwise terms the multi-task per frame improved representations obtained from CShift, and next extract their spatial consistency combined with the temporal aspect through SFSeg, like shown in Fig. 36.

Compared with the above space-time graph architecture baselines, this approach has limitations from the target task point of view. Those limitations

arise from the fact that we can only apply it for tasks for which the output represents a binary clusterization problem (like single object tracking or single instance segmentation).

EPILOGUE

ETHICAL CONSIDERATIONS

Video object tracking and segmentation tasks involve having access to a dense video record. Even though the task purpose might be moral and useful for society (*e.g.* monitor shoplifting and automatically trigger the alarm instead of having a human watching the video), the fact that humans might appear in the flow raises many privacy concerns. Also, continuously monitoring a specific area (or more) is scary and can have severe consequences if used abusively. This kind of abuse is possible mainly because the video stream is a rich source of information. Hence, allowing a large amount of secondary information (given the purpose task) to be recorded can lead to drifts from the original motivation. Moreover, this kind of action and re-purposing depends to a large extent on the current decision-maker. For instance, in the recent Black Lives Matter protest in 2020 there were abusive punishments applied to those that only participated, rather than acting violent (which was the original purpose for having cameras in the public domain).

factual example

For those reasons and many more other examples expanded even at the level of entire countries, it is clear that tracking poses ethical concerns, and we should look carefully at the impact when designing this kind of technology. I, together with several Bitdefender colleagues with an academic contribution in the field of cryptography, have a patent-pending application for preserving privacy in such a system. We propose a theoretical guaranteed solution that enables computations over encrypted data. In particular, a valid usecase is predicting instance segmentation over multiple users, where each user can decrypt only his/her part of the result.

patent

HOMOMORPHIC ENCRYPTION It was shown that deep models can work under homomorphic encryption layer [23], ensuring computations over encrypted data, protecting the user data and giving guaranties that the server can not see any bit of decrypted input information or the deep model result over the input.

TECHNOLOGY READINESS LEVEL There is a lot of research happening now at the intersection of those two fields: cryptography and AI for privacy-preserving

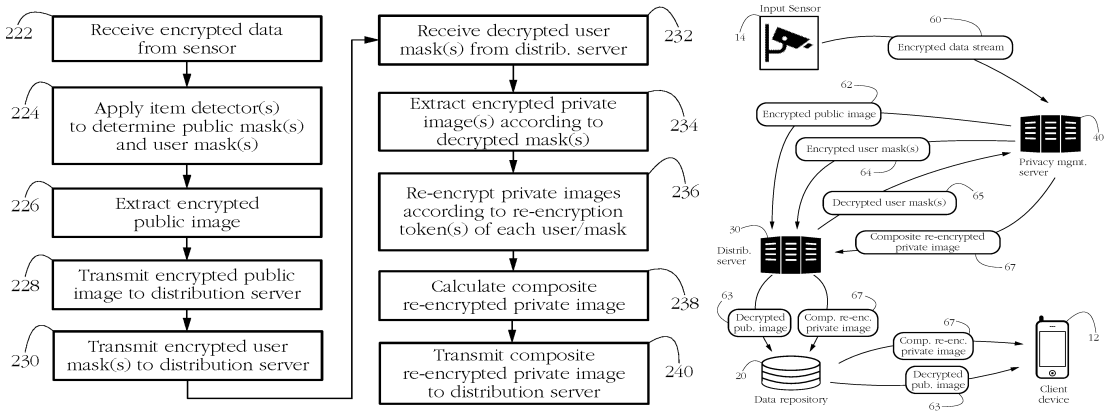

Figure 37: Patent Pending for Privacy Preserving Image Distribution.

solutions, both at large stakeholders in the industry like Google, Facebook, Microsoft, Amazon, Apple, NVIDIA, but also in the startups, where dedicated hedge funds focus on this. But from the technology point of view, the results are yet in early stages, with unfeasible speeds and model dimensions for real-time guaranties (20 layers neural network on a single MNIST encrypted image takes 13 seconds on a basic CPU).

PATENT PENDING We come up with a solution to use the homomorphic encryption scheme in the context of having multiple objects in the monitored scene, and multiple users that need to access different parts in the video's space-time volume. For instance, let's suppose we have a monitoring system installed in a kindergarten that triggers alerts for bullying cases. The parents should only receive information about their child. Moreover, the surveillance officer should not see the decrypted video flow (unless an alert is raised). I show in Fig. 37 a snippet of our proposed system.

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

BIBLIOGRAPHY

[1] N. Abraham and N. M. Khan. A novel focal tversky loss function with improved attention u-net for lesion segmentation. In *IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, 2019.

[2] L. Agapito, M. M. Bronstein, and C. Rother. Best displacement flow. In *European Conference on Computer Vision (ECCV)*, Lecture Notes in Computer Science, 2015.

[3] O. Akin, E. Erdem, A. Erdem, and K. Mikolajczyk. Deformable part-based tracking by coupled global and local correlation filters. *Journal of Visual Communication and Image Representation*, 2016.

[4] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2011.

[5] V. Belagiannis, F. Schubert, N. Navab, and S. Ilic. Segmentation based particle filtering for real-time 2D object tracking. In *European Conference on Computer Vision (ECCV)*, 2012.

[6] W. H. Beluch, T. Genewein, A. Nurnberger, and J. M. Kohler. The power of ensembles for active learning in image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[7] L. Bertinetto, J. F. Henriques, J. Valmadre, P. H. S. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *Neural Information Processing Systems (NeurIPS)*, 2016.

[8] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr. Staple: Complementary learners for real-time tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional Siamese networks for object tracking. In *European Conference on Computer Vision Workshops (ECCVW)*, 2016.

[10] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte. Know your surroundings: Exploiting scene information for object tracking. In *European Conference on Computer Vision (ECCV)*, 2020.

[11] D. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[12] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. 2008.

[13] A. Bugeau and P. Pérez. Track and cut: Simultaneous tracking and segmentation of multiple objects with graph cuts. *EURASIP Journal on Image and Video Processing*, 2008.

[14] E. Burceanu and M. Leordeanu. Learning a robust society of tracking parts. *arxiv*, 2017.

[15] E. Burceanu and M. Leordeanu. Learning a robust society of tracking parts using co-occurrence constraints. In *European Conference on Computer Vision Workshops (ECCVW)*, 2018.

[16] E. Burceanu and M. Leordeanu. A 3D convolutional approach to spectral object segmentation in space and time. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[17] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arxiv*, 2020.

[18] L. Cehovin, A. Leonardis, and M. Kristan. Robust visual tracking using template anchors. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016.

[19] D. Chen, Z. Yuan, Y. Wu, G. Zhang, and N. Zheng. Constructing adaptive complex cells for robust visual tracking. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[20] Z. Chen, B. Zhong, G. Li, S. Zhang, and R. Ji. Siamese box adaptive network for visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[21] J. Cheng, Y.-H. Tsai, W.-C. Hung, S. Wang, and M.-H. Yang. Fast and accurate online video object segmentation via tracking parts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[22] 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.

[23] I. Chillotti, D. Ligier, J.-B. Orfila, and S. Tap. Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for TFHE. In *ePrint*, 2021.

[24] P. Chockalingam, N. Pradeep, and S. Birchfield. Adaptive fragments-based tracking of non-rigid objects using level sets. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[25] P. Chrabąszcz, I. Loshchilov, and F. Hutter. Back to basics: Benchmarking canonical evolution strategies for playing Atari. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[26] R. Cipolla, Y. Gal, and A. Kendall. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[27] K. Dai, Y. Zhang, D. Wang, J. Li, H. Lu, and X. Yang. High-performance long-term tracking with meta-updater. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[28] M. Danelljan and G. Bhat. PyTracking, 2019.

[29] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. ECO: Efficient convolution operators for tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[30] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. ATOM: accurate tracking by overlap maximization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[31] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg. Discriminative scale space tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017.

[32] M. Danelljan, A. Robinson, F. Shahbaz Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *European Conference on Computer Vision (ECCV)*, 2016.

[33] M. Danelljan, L. Van Gool, and R. Timofte. Probabilistic regression for visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[35] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *IEEE International Conference on Data Mining (ICDM)*, 2001.

[36] J. Donahue and K. Simonyan. Large scale adversarial representation learning. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[37] R. Dondera, V. I. Morariu, Y. Wang, and L. S. Davis. Interactive video segmentation using occlusion boundaries and temporally coherent superpixels. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014.

[38] M. Douze, A. Ramisa, and C. Schmid. Combining attributes and fisher vectors for efficient image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[39] D. Du, H. Qi, W. Li, L. Wen, Q. Huang, and S. Lyu. Online deformable object tracking based on structure-aware hyper-graph. *IEEE Transactions on Image Processing (TIP)*, 2016.

[40] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman. Temporal cycle-consistency learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[41] A. Faktor and M. Irani. Video object segmentation by non-local consensus voting. In *British Machine Vision Conference (BMVC)*, 2014.

[42] H. Fan, H. Ling, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, and C. Liao. LaSOT: A high-quality benchmark for large-scale single object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[43] J. A. Feldman, G. M. Feldman, G. Falk, G. Grape, J. Pearlman, I. Sobel, and J. M. Tenenbaum. The Stanford hand-eye project. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1969.

[44] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)*, 2004.

[45] I. Fine, A. R. Wade, A. A. Brewer, M. G. May, D. F. Goodman, G. M. Boynton, B. A. Wandell, and D. I. A. MacLeod. Long-term deprivation affects visual perception and cortex. *Nature Neuroscience*, 2003.

[46] D. Forsyth. Object detection with discriminatively trained part-based models. *Computer*, 2014.

[47] K. Fragkiadaki, G. Zhang, and J. Shi. Video segmentation by tracing discontinuities in a trajectory embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[48] G. Frobenius. "About a Fundamental Theorem of Group Theory. II. *Session Reports of the Royal Prussian Academy of Sciences*, 1903.

[49] H. K. Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey. Need for speed: A benchmark for higher frame rate object tracking. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[50] J. Gao, T. Zhang, and C. Xu. Graph convolutional tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[51] J. J. Gibson. *The Ecological Approach to Visual Perception*. 2014.

[52] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations (ICLR)*, 2018.

[53] M. F. Glasser, T. S. Coalson, E. C. Robinson, C. D. Hacker, J. Harwell, E. Yacoub, K. Ugurbil, J. Andersson, C. F. Beckmann, M. Jenkinson, S. M. Smith, and D. C. Van Essen. A multi-modal parcellation of human cerebral cortex. *Nature*, 2016.

[54] M. Godec, P. Roth, and H. Bischof. Hough-based tracking of non-rigid objects. *Computer Vision and Image Understanding (CVIU)*, 2013.

[55] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Scholkopf, and A. J. Smola. A kernel method for the two-sample-problem. In *Neural Information Processing Systems (NeurIPS)*, 2006.

[56] B. Griffin and J. Corso. Tukey-inspired video object segmentation. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[57] M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[58] J. Guérin and B. Boots. Improving image clustering with multiple pretrained CNN feature extractors. In *British Machine Vision Conference (BMVC)*, 2018.

[59] E. Haller and M. Leordeanu. Unsupervised object segmentation in video by efficient selection of highly probable positive features. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[60] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 2020.

[61] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[62] Z. He, Y. Fan, J. Zhuang, Y. Dong, and H. Bai. Correlation filters with weighted convolution responses. In *IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017.

[63] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2015.

[64] S. Herculano-Houzel. The human brain in numbers: A linearly scaled-up primate brain. *Frontiers in Human Neuroscience*, 2009.

[65] S. Hickson, S. T. Birchfield, I. A. Essa, and H. I. Christensen. Efficient hierarchical graph-based segmentation of RGBD videos. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[66] C. L. A. Ho, R. Zimmermann, J. D. Flórez Weidinger, M. Prsa, M. Schottdorf, S. Merlin, T. Okamoto, K. Ikezoe, F. Pifferi, F. Aujard, A. Angelucci, F. Wolf, and D. Huber. Orientation preference maps in microcebus murinus reveal size-invariant design principles in primate visual cortex. *Current Biology*, 2021.

[67] P. W. Holland. Weighted ridge regression: Combining ridge and robust regression methods, 1973.

[68] L. Huang, X. Zhao, and K. Huang. GOT-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[69] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[70] A. Jabri, A. Owens, and A. A. Efros. Space-time correspondence as a contrastive random walk. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[71] A. Jabri, A. Owens, and A. A. Efros. Space-time correspondence as a contrastive random walk. *arxiv*, 2020.

[72] S. Jain, B. Xiong, and K. Grauman. FusionSeg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos. *arxiv*, 2017.

[73] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[74] M. Keuper, B. Andres, and T. Brox. Motion trajectory segmentation via minimum cost multicuts. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[75] M. Keuper, S. Tang, B. Andres, T. Brox, and B. Schiele. Motion segmentation multiple object tracking by correlation co-clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[76] A. Khoreva, F. Perazzi, R. Benenson, B. Schiele, and A. Sorkine-Hornung. Learning video object segmentation from static images. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[77] Y. J. Koh and C.-S. Kim. Primary object segmentation in videos based on region augmentation and reduction. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[78] W. M. Kouw and M. Loog. A review of domain adaptation without target labels. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[79] M. Kristan, J. M. A. Leonardis, R. Felsberg, P. M., L. Čehovin, T. Vojír, G. Häger, and et al. The visual object tracking VOT2016 challenge results. In *European Conference on Computer Vision Workshops (ECCVW)*, 2016.

[80] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Cehovin Zajc, T. Vojir, G. Hager, A. Lukezic, A. Eldesokey, and G. Fernandez. The visual object tracking VOT2017 challenge results. In *IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017.

[81] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. P. Pflugfelder, L. Zajc, T. Vojír, G. Bhat, A. Lukežič, A. Eldesokey, G. Fernández, and et al. The sixth visual object tracking VOT2018 challenge results. In *European Conference on Computer Vision Workshops (ECCVW)*, 2018.

[82] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with Gaussian edge potentials. *arxiv*, 2011.

[83] S. Kumar and M. Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *IEEE International Conference on Computer Vision (ICCV)*, 2003.

[84] C.-H. Kuo and R. Nevatia. How does person identity recognition help multi-person tracking? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[85] J. Kwon and K. M. Lee. Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping Monte Carlo sampling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[86] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)*, 2001.

[87] C. Lanczos. An iteration method for the solution of the eigenvalue i problem of linear differential and integral operators. In *Journal of Research of the National Bureau of Standards*, 1950.

[88] D. Lao and G. Sundaramoorthi. Extending layered models to 3D motion. In *European Conference on Computer Vision (ECCV)*, 2018.

[89] Y. J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[90] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. *IEEE International Conference on Computer Vision (ICCV)*, 2005.

[91] M. Leordeanu, M. Hebert, and R. Sukthankar. An integer projected fixed point method for graph matching and MAP inference. In *Neural Information Processing Systems (NeurIPS)*, 2009.

[92] M. Leordeanu, M. Pirvu, D. Costea, A. Marcu, E. Slusanschi, and R. Sukthankar. Semi-supervised learning for multi-task scene understanding by neural graph consensus. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2021.

[93] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan. SiamRPN++: Evolution of Siamese visual tracking with very deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[94] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. High performance visual tracking with Siamese region proposal network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[95] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg. Video segmentation by tracking many figure-ground segments. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[96] S. Li, B. Seybold, A. Vorobyov, A. Fathi, Q. Huang, and C.-C. J. Kuo. Instance embedding transfer to unsupervised video object segmentation.

In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[97] S. Li, Y. Zhao, R. Varma, O. Salpekar, P. Noordhuis, T. Li, A. Paszke, J. Smith, B. Vaughan, P. Damania, and S. Chintala. PyTorch distributed. *Proceedings of the VLDB Endowment (PVLDB)*, 2020.

[98] T. Lim, S. Hong, B. Han, and J. Han. Joint segmentation and pose tracking of human in natural videos. *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[99] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[100] Y. Liu, S. Albanie, A. Nagrani, and A. Zisserman. Use what you have: Video retrieval using representations from collaborative experts. In *British Machine Vision Conference (BMVC)*, 2019.

[101] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Deep transfer learning with joint adaptation networks. In *International Conference on Machine Learning (ICML)*, 2017.

[102] X. Lu, W. Wang, C. Ma, J. Shen, L. Shao, and F. Porikli. See more, know more: Unsupervised video object segmentation with co-attention Siamese networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[103] J. Luiten, P. Voigtlaender, and B. Leibe. PReMVOS: Proposal-generation, refinement and merging for video object segmentation. In *Asian Conference on Computer Vision (ACCV)*, 2018.

[104] A. Lukezic, L. Cehovin Zajc, and M. Kristan. Deformable parts correlation filters for robust visual tracking. *IEEE Transactions on Cybernetics (SMC)*, 2018.

[105] A. Lukezic, J. Matas, and M. Kristan. D3S – a discriminative single shot segmentation tracker. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[106] A. Lukezic, T. Vojir, L. C. Zajc, J. Matas, and M. Kristan. Discriminative correlation filter with channel and spatial reliability. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[107] W. Luo, X. Zhao, and T. Kim. Multiple object tracking: A review. *arxiv*, 2014.

[108] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-SVMs for object detection and beyond. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2011.

[109] K.-K. Maninis, S. Caelles, Y. Chen, J. Pont-Tuset, L. Leal-Taixe, D. Cremers, and L. Van Gool. Video object segmentation without temporal information. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[110] K.-K. Maninis, S. Caelles, Y. Chen, J. Pont-Tuset, L. Leal-Taixe, D. Cremers, and L. Van Gool. Video object segmentation without temporal information. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[111] N. Marki, F. Perazzi, O. Wang, and A. Sorkine-Hornung. Bilateral space video segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[112] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[113] M. Meila and J. Shi. A random walks view of spectral segmentation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2001.

[114] A. Miech, J.-B. Alayrac, L. Smaira, I. Laptev, J. Sivic, and A. Zisserman. End-to-end learning of visual representations from uncurated instructional videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[115] A. Miech, I. Laptev, and J. Sivic. Learning a text-video embedding from incomplete and heterogeneous data. *arxiv*, 2018.

[116] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *European Conference on Computer Vision (ECCV)*, 2016.

[117] N. C. Mithun, J. Li, F. Metze, and A. K. Roy-Chowdhury. Learning joint embedding with multimodal cues for cross-modal video-text retrieval. In *ACM on International Conference on Multimedia Retrieval (ICMR)*, 2018.

[118] M. Mueller, N. Smith, and B. Ghanem. A benchmark and simulator for UAV tracking. In *European Conference on Computer Vision (ECCV)*, 2016.

[119] K. P. Murphy. *Machine learning - a probabilistic perspective*. Adaptive computation and machine learning series. 2012.

[120] M. Müller, A. Bibi, S. Giancola, S. Al-Subaihi, and B. Ghanem. TrackingNet: A large-scale dataset and benchmark for object tracking in the wild. In *European Conference on Computer Vision (ECCV)*, 2018.

[121] H. Nam, M. Baek, and B. Han. Modeling and propagating CNNs in a tree structure for visual tracking. *arxiv*, 2016.

[122] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[123] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[124] G. Nebehay and R. Pflugfelder. Clustering of static-adaptive correspondences for deformable object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[125] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Neural Information Processing Systems (NeurIPS)*, 2001.

[126] P. Ochs and T. Brox. Object segmentation in video: A hierarchical variational approach for turning point trajectories into dense regions. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[127] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[128] E. Park and A. C. Berg. Meta-tracker: Fast and robust online adaptation for visual object trackers. In *European Conference on Computer Vision (ECCV)*, 2018.

[129] M. Patrick, Y. M. Asano, R. Fong, J. F. Henriques, G. Zweig, and A. Vedaldi. Multi-modal self-supervision from generalized data transformations. *arxiv*, 2020.

[130] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[131] F. Perazzi, O. Wang, M. Gross, and A. Sorkine-Hornung. Fully connected object proposals for video segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[132] A. Piergiovanni, A. Angelova, and M. S. Ryoo. Evolving losses for unsupervised video representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[133] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbel'aez, A. Sorkine-Hornung, and L. {Van Gool}. The 2017 DAVIS challenge on video object segmentation. *arxiv*, 2017.

[134] N. Pourian, S. Karthikeyan, and B. S. Manjunath. Weakly supervised graph based semantic segmentation by learning communities of image-parts. *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[135] E. Real, J. Shlens, S. Mazzocchi, X. Pan, and V. Vanhoucke. YouTube-BoundingBoxes: A large high-precision human-annotated data set for object detection in video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[136] F. Reda, R. Pottorff, J. Barker, and B. Catanzaro. flownet2-pytorch: Pytorch implementation of FlowNet 2.0: Evolution of optical flow estimation with deep networks, 2017.

[137] M. Roberts and N. Paczan. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. arxiv 2020.

[138] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2015.

[139] O. Sagi and L. Rokach. Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery*, 2018.

[140] S. Sarkar and P. Soundararajan. Supervised learning of large perceptual organization: Graph spectral partitioning and learning automata. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2000.

[141] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2000.

[142] G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah. Part-based multiple-person tracking with partial occlusion handling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[143] M. Siam, C. Jiang, S. Lu, L. Petrich, M. Gamal, M. Elhoseiny, and M. Jagersand. Video object segmentation using teacher-student adaptation in a human robot interaction (HRI) setting. *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[144] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arxiv*, 2014.

[145] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2014.

[146] H. Song, W. Wang, S. Zhao, J. Shen, and K. Lam. Pyramid dilated deeper ConvLSTM for video salient object detection. In *European Conference on Computer Vision (ECCV)*, 2018.

[147] H. Song, W. Wang, S. Zhao, J. Shen, and K.-M. Lam. Pyramid dilated deeper ConvLSTM for video salient object detection. In *European Conference on Computer Vision (ECCV)*, 2018.

[148] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. Lau, and M.-H. Yang. VITAL: Visual tracking via adversarial learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[149] X. Soria, E. Riba, and A. Sappa. Dense extreme inception network: Towards a robust CNN model for edge detection. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[150] X. Soria, E. Riba, and A. Sappa. Dense extreme inception network: Towards a robust CNN model for edge detection. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[151] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2017.

[152] R. Tao, E. Gavves, and A. W. M. Smeulders. Siamese instance search for tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[153] B. Taylor, V. Karasev, and S. Soattoc. Causal video object segmentation from persistence of occlusions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[154] Y. Tian, D. Krishnan, and P. Isola. Contrastive multiview coding. *arxiv*, 2019.

[155] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola. What makes for good views for contrastive learning. *arxiv*, 2020.

[156] P. Tokmakov, K. Alahari, and C. Schmid. Learning motion patterns in videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[157] P. Tokmakov, K. Alahari, and C. Schmid. Learning video object segmentation with visual memory. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[158] P. Tokmakov, M. Hebert, and C. Schmid. Unsupervised learning of video representations via dense trajectory clustering. *arxiv*, 2020.

[159] A. Torralba. MIT AI Lab. https://groups.csail.mit.edu/vision/torralbalab/, 2021.

[160] A. Torsello, S. R. Bulò, and M. Pelillo. Grouping with asymmetric affinities: A game-theoretic perspective. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[161] G. Varoquaux, L. Buitinck, G. Louppe, O. Grisel, F. Pedregosa, and A. Mueller. Scikit-learn. *GetMobile: Mobile Computing and Communications*, 2015.

[162] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe. MOTS: Multi-object tracking and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[163] P. Voigtlaender and B. Leibe. Online adaptation of convolutional neural networks for video object segmentation. In *British Machine Vision Conference (BMVC)*, 2017.

[164] P. Voigtlaender, J. Luiten, P. H. Torr, and B. Leibe. Siam r-CNN: Visual tracking by re-detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[165] T. Vojir, J. Noskova, and J. Matas. Robust scale-adaptive mean-shift for tracking. *Pattern Recognition Letters*, 2014.

[166] G. Wang, C. Luo, X. Sun, Z. Xiong, and W. Zeng. Tracking by instance detection: A meta-learning approach. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[167] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao. Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[168] L. Wang, H. Lu, Y. Wang, M. Feng, D. Wang, B. Yin, and X. Ruan. Learning to detect salient objects with image-level supervision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[169] N. Wang, Y. Song, C. Ma, W. Zhou, W. Liu, and H. Li. Unsupervised deep tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[170] P. Wang and A. C. S. Chung. Focal dice loss and image dilation for brain tumor segmentation. In *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2018.

[171] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr. Fast online object tracking and segmentation: A unifying approach. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[172] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr. Fast online object tracking and segmentation: A unifying approach. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[173] S. Wang, H. Lu, F. Yang, and M.-H. Yang. Superpixel tracking. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[174] S. Wang and J. Siskind. Image segmentation with minimum mean cut. In *IEEE International Conference on Computer Vision (ICCV)*, 2001.

[175] X. Wang and J. Yu. Learning to cartoonize using white-box cartoon representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[176] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing (TIP)*, 2004.

[177] T. Whelan, M. Goesele, S. J. Lovegrove, J. Straub, S. Green, R. Szeliski, S. Butterfield, S. Verma, and R. Newcombe. Reconstructing scenes with mirror and glass surfaces. *ACM Transactions on Graphics*, 2018.

[178] C. S. Won and H. Derin. Unsupervised segmentation of noisy and textured images using Markov random fields. *Graphical Models and Image Processing (CVGIP)*, 1992.

[179] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2015.

[180] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 1993.

[181] Y. Xu, Z. Wang, Z. Li, Y. Yuan, and G. Yu. SiamFC++: Towards robust and accurate visual tracking with target estimation guidelines. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2020.

[182] H. Yan, Y. Ding, P. Li, Q. Wang, Y. Xu, and W. Zuo. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[183] F. Yang, Q. Sun, H. Jin, and Z. Zhou. Superpixel segmentation with fully convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[184] L. Yang, Y. Fan, and N. Xu. Video instance segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[185] L. Yang, Y. Wang, X. Xiong, J. Yang, and A. K. Katsaggelos. Efficient video object segmentation via network modulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[186] Z. Yang, Q. Wang, L. Bertinetto, S. Bai, W. Hu, and P. Torr. Anchor diffusion for unsupervised video object segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[187] M. Ye and Y. Guo. Progressive ensemble networks for zero-shot recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[188] C.-P. Yu, H. Le, G. J. Zelinsky, and D. Samaras. Efficient video segmentation using parametric graph partitioning. *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[189] S. X. Yu and J. Shi. Grouping with directed relationships. In *Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, 2001.

[190] A. R. Zamir, A. Sax, N. Cheerla, R. Suri, Z. Cao, J. Malik, and L. J. Guibas. Robust learning through cross-task consistency. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[191] A. R. Zamir, A. Sax, W. Shen, L. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[192] Y. Zeng, D. Samaras, W. Chen, and Q. Peng. Topology cuts: A novel min-cut/max-flow algorithm for topology preserving segmentation in N–D images. *Computer Vision and Image Understanding (CVIU)*, 2008.

[193] X. Zhan, J. Xie, Z. Liu, Y.-S. Ong, and C. C. Loy. Online deep clustering for unsupervised representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[194] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European Conference on Computer Vision (ECCV)*, 2016.

[195] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[196] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[197] T. Zhang, C. Xu, and M.-H. Yang. Multi-task correlation particle filter for robust object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[198] Z. Zhang and H. Peng. Ocean: Object-aware anchor-free tracking. In *European Conference on Computer Vision (ECCV)*, 2020.

[199] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ADE20K dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[200] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[201] T. Zhou, J. Li, S. Wang, R. Tao, and J. Shen. MATNet: Motion-attentive transition network for zero-shot video object segmentation. *IEEE Transactions on Image Processing (TIP)*, 2020.

[202] G. Zhu, F. Porikli, and H. Li. Beyond local search: Tracking objects everywhere with instance-specific proposals. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[203] T. Zhuo, Z. Cheng, P. Zhang, Y. Wong, and M. Kankanhalli. Unsupervised online video object segmentation with motion property understanding. *IEEE Transactions on Image Processing (TIP)*, 2020.

This PhD was proposed and partially financed by the Doctoral Studies Program of the Francophone Center in Mathematics, through the 2016 partnership between Francophone University Agency, Institute of Mathematics "Simion Stoilow" of the Romanian Academy, and the University of Bucharest.

Elena Burceanu: *Efficiently Exploiting Space-Time Consensus for Object Segmentation and Tracking in Video* PhD thesis, 2022.

elena.burceanu@gmail.com