



# Instance Segmentation, Tracking and Action Detection of Animals in Wildlife Videos

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von
**Frank Schindler**
aus
Bonn, Deutschland

Bonn, 2024

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

Gutachter / Betreuer: PD Dr. Volker Steinhage
Gutachter: Prof. Dr. Joachim K. Anlauf
Tag der Promotion: 6.11.2024
Erscheinungsjahr: 2024

# Abstract

Monitoring animal species efficiently in their natural habitats is essential to describe and analyze the development of ecosystems and populations and to detect the causes of changes due to climate change or other external influences. Camera traps are increasingly being used to generate video material. Until now, however, the resulting material has either been examined manually by researchers or with systems that require their expert knowledge. Supporting ecologists with AI applications is not only necessary due to the large amount of data and limited number of available experts, but also enables new insights and standardized analyses. Therefore, an automation of this analysis process by adapting the prominent computer vision tasks of instance segmentation, tracking, and action detection to the context of ecology can help to solve important ecological problem statements like population estimation, animal migration or behavioral analysis.

In this doctoral thesis, we present a new approach to perform instance segmentation, tracking and action detection for camera trap videos of animals in one system. Central to our research is how reliable instance segmentation can improve both tracking and action detection.

The ability to accurately detect and track animals in wildlife videos is essential for researchers to analyze animal behavior and identify individual animals. Simply detecting animals by bounding boxes is not enough to distinguish between animals that are in close proximity to each other. Instead, a precise contour of each animal, an instance mask, is required, which is obtained by the instance segmentation. Moreover, an instance mask shows the pose of the animal, which is helpful for a detailed action recognition. We introduce SWIFT (Segmentation With FIltering of Tracklets), a novel multi-object tracking and segmentation (MOTS) pipeline that effectively addresses this problem. SWIFT improves the average precision of the instance masks compared to using state-of-the-art computer vision instance segmentation approaches by 4 percentage points on average for the different datasets. The SWIFT Tracking Algorithm that uses multiple filtering steps to either delete tracks that are found incorrectly or to merge tracks that are not yet connected achieves multi-object tracking and segmentation accuracy scores up to 68.0%.

Furthermore, we propose our action detection system that leverages SWIFT for animal detection and introduces MAROON (MAsk guided Action RecOgnitiON), a novel network for action recognition. The instance masks of SWIFT are used to cutout the animals for the input for MAROON, which allows the network to focus on the actor. MAROON uses a novel triple-stream architecture to extract motion and appearance features with different granularities. MAROON achieves an action recognition accuracy of 72.24% on the Rolandseck Daylight dataset, which includes 11 distinct action classes.

For the evaluation of our models, we introduce five different wildlife camera trap datasets that we manually annotated containing videos of European mammals like red deer, roe deer, fallow deer, boars, hares and foxes. Two of these five datasets were created by us using camera traps in the wildlife enclosure Rolandseck. Each dataset represents a different scenario in wildlife monitoring. These datasets are the first datasets in wildlife monitoring that are annotated with instance masks, track IDs and action labels at once.

# List of Publications

This thesis is based on the following publications:

Frank Schindler and Volker Steinhage. Identification of animals and recognition of their actions in wildlife videos using deep learning techniques. *Ecological Informatics*, 61:101215, 2021a. doi: https://doi.org/10.1016/j.ecoinf.2021.101215.

Frank Schindler and Volker Steinhage. Saving costs for video data annotation in wildlife monitoring. *Ecological Informatics*, 65:101418, 2021b.
doi: https://doi.org/10.1016/j.ecoinf.2021.101418.

Frank Schindler and Volker Steinhage. Instance segmentation and tracking of animals in wildlife videos: SWIFT-segmentation with filtering of tracklets. *Ecological Informatics*, 71:101794, 2022. doi: https://doi.org/10.1016/j.ecoinf.2022.101794.

Frank Schindler and Volker Steinhage. SWIFT - an efficient and effective application of instance segmentation and tracking in wildlife monitoring. *In Workshop Camera Traps, AI and Ecology*, Jena, 2023.

Frank Schindler, Volker Steinhage, Suzanne van Beeck Calkoen, and Marco Heurich. Action detection for wildlife monitoring with camera traps based on segmentation with filtering of tracklets (SWIFT) and mask-guided action recognition (MAROON). *Applied Sciences*, 14(2):514, 2024. doi: https://doi.org/10.3390/app14020514.

# Acknowledgements

First of all, I would like to thank my supervisor PD Dr. Volker Steinhage for the opportunity to carry out my PhD in the Intelligent Vision Systems Group. He always supported me, gave me advice for any questions I had and helped me come up with new ideas through our discussions together. I would also like to thank Prof. Dr. Joachim K. Anlauf, who is the second reviewer and second supervisor of my thesis. Our discussions in our meetings helped me to develop new ideas for my work. I also want to thank the other two members of my thesis committee, Prof. Dr. Michael Meier and Prof. Dr. Christoph Scherber.

I am very grateful that my PhD was funded by the PhD scholarship of the Konrad-Adenauer-Stiftung, which provided me not only with financial, but also non-material support during my doctoral studies. This gave me a lot of freedom for my work and at the same time the interdisciplinary exchange with other scholarship holders was very beneficial. Moreover, I want to thank the Konrad-Adenauer-Stiftung and the Studienstiftung des deutschen Volkes for my scholarships during my Bachelor and Master studies, which have already given me new insights and motivation for doing a PhD.

Further, I want to thank the German Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung (BMBF)), Bonn, Germany for funding the project "Automated Multisensor station for Monitoring Of species Diversity" (AMMOD) (FKZ 01LC1903B), in which I worked during my PhD. This resulted in a valuable interdisciplinary exchange with other researchers in Germany.

I want to thank the other members of my working group, the Intelligent Vision Systems Group, Morris Klasen, Timm Haucke and Florian Huber, who provided a pleasant working environment. Our discussions about our projects were also always very exciting and beneficial.

Additionally, I would like to thank all the students, who participated in project groups or wrote Bachelor theses and Master theses that I supervised during this time: Johannes Bergmann, Hoan Vu, Bertan Karakora, Iva Ewert, Alvin Inderka, Stefan Messerschmidt, Simon Mathy, Jennifer Collard, Pablo Rauh Corro, Elmar Leschke, Philip Gutberlet, Johannes van de Locht, Benedikt Wude and Jon Breid.

I always enjoyed supervising them and the findings from their work also helped my own research.

I would particularly like to thank the Wildpark Rolandseck GmbH and its manager Tessa von Lüdinghausen, who made it possible for me to record red deer and fallow deer with my own wildlife cameras over a period of one year in their wildlife enclosure and gave me valuable information about the animals there. A big thank you goes to the following students who also participated in the data annotation of the Rolandseck datasets as part of their project work and bachelor theses: Bertan Karakora, Iva Ewert, Alvin Inderka, Stefan Messerschmidt, Simon Mathy, Jennifer Collard, Pablo Rauh Corro, Philip Gutberlet, Johannes van de Locht and Benedikt Wude. Without their annotation support, the creation of these new datasets, which is so important for ecology, would have taken much longer. Moreover, I would like to thank Benedikt Wude, who improved parts of the annotation tool in his project work to make it easier to use.

Furthermore, I want to thank Prof. Dr. Bernd Radig for providing the camera trap videos of the Wildlife Crossings dataset.

Moreover, I would like to thank the researchers from the Bavarian Forest National Park, Prof. Dr. Marco Heurich and Suzanne van Beeck Calkoen, for our joint collaboration. In particular, the selection and provision of videos from the national park as another important dataset for my work was very valuable. The professional ecological description of the action classes by Suzanne van Beeck Calkoen was also very valuable for a meaningful evaluation of the data material. I would like to thank the student Hannah Hasert for annotating a large part of the Bavarian Forest datasets.

I am especially grateful for the support of my family and friends, who have been supportive and understanding during this time.

# Contents

# Chapter 1

# Introduction

Comprehending animal behavior is a key aspect of conservation biology (Berger-Tal et al., 2011), making the detection of actions by wild animals critical for ecologists supporting conservation initiatives (Caravaggi et al., 2017). Despite its importance, the manual analysis of wild animal behavior is often neglected due to the extensive workload required to examine all gathered data.

Camera traps have emerged as a popular and effective tool in ecology and conservation, providing a reliable and minimally intrusive visual method for monitoring wildlife (McCallum, 2013; Burton et al., 2015; Boitani, 2016; Caravaggi et al., 2017; Wearn and Glover-Kapfer, 2019). The use of stationary camera traps allows wildlife to be monitored over an extended period of time by recording videos. Over the last decades, camera traps have been adopted for various ecological tasks, including abundance estimation (Hongo et al., 2021; Villette et al., 2017; Henrich et al., 2023), quantification of species diversity (Tobler et al., 2008) detection of rare species (Linkie et al., 2007), investigation of animal activity patterns (Frey et al., 2017) or the analysis of species replacement processes (Caravaggi et al., 2016). The immense volume of data collected from camera traps necessitates the use of artificial intelligence for automatic analysis to efficiently handle and process this information (Green et al., 2020; Mitterwallner et al., 2023). Analyzing data from even a small number of camera traps can require months to process just a few weeks' worth of footage. Various AI models have been developed to manage and analyze camera trap images and videos in order to classify and categorize species in the images (Vélez et al., 2023). However, despite the existence of some AI models capable of classifying species in camera trap videos, there remains a gap in tools specifically designed for the automatic quantification of animal behavior.

To determine the number of animals present for an abundance estimation of the animals (Hongo et al., 2021), the individuals within a video must be detected and not only classified. Moreover, the detection and tracking of individual animals in a video is an essential basis for solving further problems such as action detection

(Schindler and Steinhage, 2021a) or re-identification (Schneider et al., 2019) of individual animals in multiple videos. These advancements facilitate more streamlined and effective behavioral studies of wild animals using camera traps (Caravaggi et al., 2017; van Beeck Calkoen et al., 2021).

## 1.1 Problem Definition and Objectives

The goal of this thesis is to develop the first system that can detect and track animals and describe their actions in camera trap videos. The input for this system are camera trap videos from different European mammals, especially deer species. The final output of the system consists of 6 parts, which are explained below in accordance with the respective parts of the system.

In order to detect animals in a frame of a video and to determine their exact outlines, an instance segmentation is necessary. This means that the detection of an animal in one frame results in (1) a bounding box to pinpoint its location, (2) a segmentation mask to outline its precise shape, (3) a class label to specify the type of animal detected, and (4) a score value to indicate the confidence level of the detection. Exact instance masks are necessary for further processing in an action detection or a re-identification as well as for distinguishing animals that are close to each other.

Multi-object tracking integrates the instance segmentation detections into continuous tracks, enabling the monitoring of individual animals throughout a video sequence. This process involves assigning (5) a unique track ID to each detection, augmenting the previously described detection results. Therefore, tracking allows the analysis of single individuals over an entire video, not just in a single frame or image. While the tracking process is running, incomplete tracks are referred to as tracklets. Only the final results of the tracking are then the derived tracks. The combined task of instance segmentation and tracking is called multi-object tracking and segmentation (MOTS). The goal of a MOTS pipeline is to localize the animals by instance masks and track them in camera trap videos.

Action recognition, also known as action classification, describes the process of attributing an (6) action label to a short video sequence (Simonyan and Zisserman, 2014; Kong and Fu, 2022), where a single actor performs a single action. Action detection combines the tasks of identifying an object in a video and subsequently recognizing the action being performed by that object. This enables the identification of various actions by different actors simultaneously within a video. However, the term action detection can also refer to the temporal identification of actions, not just their spatial detection in a video. This means focusing on pinpointing the beginning and end of an action within the video timeline. In this case, action detection refers again to a single individual rather than multiple actors. In this thesis we consider the first definition of action detection.

Therefore, our system performs an instance segmentation for the detection of the animals, a multi-object tracking to track the individuals in a video and an action detection to describe the different actions of the animals.

## 1.2 Challenges

Detecting animals in camera trap videos, tracking them and describing their actions is a difficult problem. The recordings of the animals take place during the day and at night. Depending on the animal species and the recording scenario, the day or night recordings play a more important role. For example, daylight shots are easier to take in a wildlife park, as the animals live there largely undisturbed. Recordings from wildlife crossings on highways, on the other hand, are mainly taken at night, as this scenario poses a greater threat to the animals. Different lighting conditions occur in both day and night shots, so that the detection of the animals based on their appearance can be very different. In particular, the lack of color information in the infrared images taken at night poses a particular challenge for the detection and classification of the animals. The lighting of the scenarios at night increases this challenge, as the edges of the image are usually darker and the center can be additionally brightened by an infrared flash. For recordings from wildlife crossings, a light source may also be installed on the opposite side of the camera's shooting location in order to illuminate the scene more evenly. However, the flash of light can also be very bright, which can make detection more difficult. The various challenges mentioned mean that a robust and versatile object detector is needed to locate animals in the various scenarios.

As all the shots take place outside, it is not only the time of the recording that plays a role, but also the location and the weather conditions. The vegetation can be very different. When taking pictures in wildlife parks or at wildlife crossings, there are usually only minor restrictions to the field of view due to trees or bushes. When taking pictures in national parks, for example, on the other hand, many bushes and grasses can often lead to more occlusions of the animals. Changing weather conditions, such as rain or fog, can also present challenges in terms of detection. How relevant these special conditions are for a study of the animals depends on the location of the recordings, i.e. whether the weather or vegetation are typical for these scenarios and occur frequently. Since vegetation can also change throughout the year (especially in the European regions considered here), occlusions from vegetation in fall and winter, for example, are reduced. Due to the occlusions of animals by other animals or objects, it is important for detection that animals are recognized by exact contours in order to identify them clearly. At the same time, possible occlusions also mean frequent interruptions of tracks in the context of multi-object tracking. Therefore, the multi-object tracking approach used must be robust and versatile for different scenarios.

One of the biggest practical challenges for our work is that there are, to the best of our knowledge, no publicly available datasets in the field of wildlife monitoring that are already annotated for the three aforementioned objectives of instance segmentation, tracking and action detection. We therefore had to create our own datasets and obtain data from collaborations. Moreover, the manual data annotation is very time-consuming for the desired tasks. We explain this in detail in the dataset chapter 3.

The classification of the animals is also challenging, as in our work we look in particular at different deer species (red deer, roe deer and fallow deer). These animal classes have a similar appearance especially if they are far away from the camera or the weather conditions and lighting are bad. Here, too, we need a robust object detector and classifier that can be trained with a relatively small amount of training data.

Another major challenge in the context of action recognition is the sometimes very similar actions of the animals. For example, the actions walking and foraging moving are very similar in terms of the movement of the animals and differ only in the position of the animal's head. In addition, actions such as head lowering and head raising are very similar in execution and could not be distinguished by a single frame. Accordingly, temporal information plays an important role here. Moreover, animals often move in groups, where they (partially) occlude each other. This makes an action recognition difficult. Therefore, an action recognition approach is needed that can focus on single animals and is robust to irritating backgrounds.

## 1.3 Contributions

In this thesis, we present the following main contributions to the tasks of instance segmentation, tracking and action detection in general and in our application area, the wildlife monitoring:

1. We present our novel multi-object tracking and segmentation (MOTS) pipeline, SWIFT - Segmentation WIth Filtering of Tracklets (Schindler and Steinhage, 2022). This is the first approach that tracks the exact contours of animals in camera trap videos using instance masks. For the first part of SWIFT we developed a new refinement algorithm, the *SWIFT Refinement Algorithm* (Algorithm 4.1), that combines the results of a trained Mask R-CNN (He et al., 2017) with a HR-Net (Wang et al., 2020a) to refine the instance masks and to improve their quality. The second part of SWIFT consists of our novel tracking algorithm, the *SWIFT Tracking Algorithm* (Algorithm 4.2), that enables a generation of tracks from the derived detections from the first part. Here we use the detected instance masks for more precise tracking than the usual

bounding boxes. The core idea of our tracking algorithm includes multiple filtering stages to delete or combine found tracklets. An exact system overview of SWIFT is given in Figure 4.4 in Section 4.2.

2. Our action recognition network MAROON, MAsk guided Action RecOgnitiON (Schindler et al., 2024), is the first method (in the whole computer vision community) to combine masked input sequences in combination with a triple-stream approach to extract motion and appearance features with different granularities. The use of instance masks allows better recognition of behavioral patterns and is also necessary when several animals are close together. The network architecture of MAROON is depicted in Figure 5.3 in Section 5.3. Our approach distinguishes between up to 13 different action classes. A precise definition of the different action classes is given in Table 3.2 in Section 3.4.1.

3. Our complete action detection system combines SWIFT and MAROON for a successful instance segmentation, tracking and action detection of animals in wildlife videos. The workflow of our complete action detection approach is shown in Figure 5.2 in Section 5.2. Our approach is the first approach to perform all these tasks in one system in the application area of wildlife monitoring. This is important for ecological applications, as it allows action recognition to benefit directly from instance segmentation. If the systems are separated, instance masks are normally not used and bounding boxes are used for action detection instead. In particular, the use of instance masks for improved tracking and more precise action recognition is a key new feature of this system. For the evaluation of our approach, we are introducing five new datasets annotated by us, two of which we recorded ourselves using camera traps. A detailed description of the datasets is given in Chapter 3.

To summarize our contributions: Instance segmentation is the central solution in our overall approach in order to improve both tracking and action recognition. Instance masks allow a more accurate matching in comparison to bounding boxes in the multi-object tracking. Moreover, the action recognition is improved because the instance masks enable a focus on the animal.

## 1.4 Thesis Structure

This thesis is divided into six chapters:

In Chapter 2 we discuss the related work for the topics instance segmentation, multi-object tracking, multi-object tracking and segmentation (MOTS), action recognition and action detection in detail. Moreover, we present all approaches regarding these topics in the context of wildlife monitoring or in general animals. We classify our contributions in the structure of related work from the various task areas.

Chapter 3 deals with the description of the datasets that we introduce in this thesis. We present the special characteristics of the individual datasets and also describe the recording process for our own Rolandseck dataset. Furthermore, we present our adapted and improved annotation tool and describe the difficulties of the annotation process. Parts of this chapter are based on (Schindler and Steinhage, 2021b).

In Chapter 4 we present and explain our novel instance segmentation and tracking approach SWIFT, Segmentation WIth Filtering of Tracklets. We give detailed explanations of our workflow and our designed refinement and tracking algorithms. For our tracking workflow in particular, in which several filtering stages are carried out in succession, we go through the algorithms line by line and explain our design decisions. We compare the instance segmentation and tracking ability of SWIFT with state-of-the-art approaches. Furthermore, we explain in detail the complex COCO (Common Objects in COntext) metrics, MOT metrics and MOTS metrics. This chapter is based on the publications (Schindler and Steinhage, 2021a, 2022, 2023).

Chapter 5 contains the description of our action detection approach. We introduce our new action recognition network MAROON, MAsk guided Action RecOgnitiON. Thereby, we analyze the individual components of our network, in particular the influence of instance masks and the significance of our triple-stream approach. Moreover, we explain how SWIFT and MAROON can be combined to perform action detection on camera trap videos. This chapter is mainly based on (Schindler et al., 2024).

In our final Chapter 6 we summarize this thesis and give an outlook on future research and possible extensions for our designed approach.

# Chapter 2

# Related Work

Our review of related literature is structured into five distinct sections. We begin by examining different techniques used in instance segmentation. Following this, we provide a detailed overview of the latest developments in multi-object tracking (MOT) and discuss how tracking and segmentation are combined in multi-object tracking and segmentation (MOTS). We then explore key research in action recognition and detection. Finally, we highlight important studies in wildlife monitoring that pertain to these topics.

We conclude each of the five sections of this chapter with a brief summary. Finally, we briefly classify our approaches in the literature presented.

## 2.1 Instance Segmentation

Instance segmentation aims to provide each object instance in an image (or video frame) with a pixel-level segmentation mask and a class label. This is particularly crucial in wildlife monitoring for differentiating animals that are in close proximity to one another. Unlike semantic segmentation, which groups all instances of a similar object into one category, instance segmentation identifies each instance separately. For instance, if a frame contains three deer, instance segmentation will individually locate and distinguish each deer with its own segmentation mask. In comparison, a general object detection would only assign a bounding box and a class label to each animal, without differentiating between overlapping instances.

Instance segmentation techniques are typically categorized into two types: single-stage and two-stage approaches, each differing in their underlying object detection mechanisms. Two-stage approaches tend to provide more precise instance masks but are generally slower than their single-stage counterparts. A two-stage detector first creates object proposals, then refines these through bounding box regression and classification to pinpoint the object's exact location. Because of this procedure, these approaches are also called top-down approaches. One disadvantage of this

approach is that the mask quality depends on the bounding box found. In contrast, single-stage detectors bypass the region proposal phase and instead directly sample potential object locations from the input image, enhancing their speed. Therefore, these approaches are also called single-shot approaches.

In our specific application of analyzing wildlife footage from camera traps, real-time detection is not a necessity since the videos are processed post-recording. Therefore, the speed advantage of single-stage detectors does not play a critical role in our context, allowing us to prioritize accuracy over processing speed.

Comprehensive surveys on various instance segmentation and object detection methods are available in the literature (Hafiz and Bhat, 2020; Liu et al., 2020; Gu et al., 2022). These surveys provide detailed reviews of the techniques and advancements in these fields.

Mask R-CNN is one of the most renowned two-stage instance segmentation methods, known for its effectiveness in generating segmentation masks (He et al., 2017). Since its introduction, Mask R-CNN has consistently outperformed the previous winners of the COCO segmentation challenge, achieving a bounding box AP of 37.1 on the official COCO test dataset for instance segmentation. Many subsequent two-stage approaches have been developed based on Mask R-CNN, including those by Huang et al. (2019); Fang et al. (2019); Chen et al. (2019a). Additionally, CD-Net (Lv et al., 2022) builds upon the results of Mask R-CNN and further refines them using a graph convolutional network, demonstrating the ongoing influence and adaptability of Mask R-CNN in the field.

Sequential Grouping Networks (SGN) (Liu et al., 2017) employ a series of multiple networks to generate final instance masks. This process starts by predicting object breakpoints, then grouping these into line segments, and ultimately forming connected components. This method bears some resemblance to our approach in that it utilizes multiple networks sequentially to enhance the accuracy of the instance masks. A key distinction, however, is that our method already produces instance masks after the first network, which are then further refined, whereas the SGN approach only yields partial results after each network stage, culminating in a complete instance mask only at the final stage.

The Path Aggregation Network (Liu et al., 2018) focuses on enhancing the extraction of information in the early layers of the network to improve the features in the later layers. Similarly, the Self-Balanced R-CNN, introduced by Rossi et al. (2022)), aims to enhance the extraction of regions of interest (ROIs) while also reducing the overall number of parameters in the two-stage model. These innovations are part of ongoing efforts to refine the efficiency and effectiveness of instance segmentation techniques.

Prominent examples of single-stage instance segmentation include TensorMask (Chen et al., 2019c) and SipMask (Cao et al., 2020). TensorMask employs a sliding window approach to address complex segmentation tasks involving numerous objects.

Despite its capabilities, the performance of Mask R-CNN generally surpasses that of TensorMask. SipMask enhances instance-specific details by splitting the predicted segmentation mask of an object into sub-regions within a single bounding box, preserving finer details per instance. The Self-Balanced R-CNN (SBR-CNN) (Rossi et al., 2022) introduces a new refinement mechanism for ROIs by using a loop between the detection head and the ROI extractor. Moreover, they improve the ROI extractor by better integrating low and high level features.

Other notable single-stage detectors include CenterMask (Lee and Park, 2020), explicit shape encoding for real time instance segmentation (ESE-Seg) (Xu et al., 2019a), reciprocal object detection and instance segmentation network (RDSNet) (Wang et al., 2020b), single-shot instance segmentation with affinity pyramid (SSAP) (Gao et al., 2019), single-stage instance segmentation based on anchor boxes (Cai and Li, 2021), BorderPointsMask (Yang et al., 2022) segmenting objects by location (SOLO) (Wang et al., 2020c) and its successor SOLOv2 (Wang et al., 2020d), and MetricMask (Wang et al., 2022).

Foundation models are playing an increasingly important role in instance segmentation. A foundation model is a very large model that has been trained on a large amount of data so that it can be used for various application scenarios. The model is no longer specialized or fine-tuned for a specific application, which is also difficult to implement due to its size and the computing power required. The most important representative in the area of instance segmentation is the Segment Anything Model (SAM) (Kirillov et al., 2023). However, the SAM model does not have a built-in classifier, so the model only segments interesting objects in an image, but cannot classify them. As a result, the network requires additional inputs such as positive and negative points or a bounding box for more accurate detection. Otherwise, the foundation model must be combined with another classifier to classify instances. For general object detecting a famous foundation model is DETR with Improved deNoising anchOr boxes (DINO) (Zhang et al., 2022). There are already several approaches that integrate SAM for instance segmentation in special application areas. For example, SAM is used for object segmentation underwater when diving (Lian et al., 2024). In the work of Lu et al. (2024), SAM is used in a workflow to detect new object classes. SAM is combined with DINO to detect basic objects in an image. For these reasons, we also analyze SAM as a possible refinement network in our work.

Video instance segmentation methods incorporate video data directly into the segmentation process and simultaneously perform tracking. These approaches, which integrate both instance segmentation and tracking, are categorized under Multi-Object Tracking and Segmentation (MOTS). For a more detailed discussion on these methods, refer to Section 2.3.

To summarize, the most important distinction for instance segmentation approaches is whether they are single-stage or two-stage approaches. The focus is either on the speed and compactness of the architecture or on the quality of the instance masks.

However, established basic models such as the Mask R-CNN are often used and extended with additional functionalities. Another important basis are the increasingly prominent foundation models such as Segment Anything. With further improvements in the future, specialized approaches will only be necessary for specific application scenarios and the general instance segmentation task can be solved with foundation models.

## 2.2 Multi-Object Tracking MOT

The task of multi-object tracking (MOT) involves identifying objects within a video and generating trajectories that describe their movement throughout the entire video. In MOT, a bounding box is typically sufficient to identify each object.

The surveys of Ciaparrone et al. (2020) and Xu et al. (2019b) offer a comprehensive overview of various contemporary approaches in the field of multi-object tracking.

The tracking-by-detection paradigm is currently the most common and effective method for performing multi-object tracking (MOT). This approach begins with a detection model that identifies objects in a video, followed by data association that links these detections across frames to form continuous tracks. Consequently, improvements in the detection model directly improve tracking accuracy. Our SWIFT method also utilizes this tracking-by-detection framework.

Alternative tracking methodologies that do not adhere to the tracking-by-detection paradigm are often referred to as one-shot tracking or joint-detection-and-tracking approaches. These methods integrate detection and tracking within a single network process. While they generally yield less accurate results than the traditional tracking-by-detection methods, their faster processing speeds make them more suitable for real-time applications.

One of the most famous tracking-by-detection approaches is the simple online and realtime tracking (SORT) (Bewley et al., 2016), which combines the detections of an arbitrary object detector by the Hungarian Matching algorithm and a Kalman filter. There are various tacking approaches that build on this idea. DeepSORT (Wojke et al., 2017) integrates appearance features in the tracking process to better overcome occlusions. In contrast, StrongSORT (Du et al., 2023) introduces two lightweight appearance-free algorithms to better refine the tracks. BoT-SORT (Aharon et al., 2022) includes camera motion compensation and Re-ID features for a more robust data association. The observation centric SORT (OC-SORT) (Cao et al., 2023) improves the Kalman filter as motion model by incorporating the momentum of the object movement to better track non-linear movements. The BYTE track approach (Zhang et al., 2021a) integrates the detection score quality into the tracking process of SORT by first forming tracks from very reliable detections and then adding the less reliant detections. The P3AFormer (Zhao et al., 2022b) does not track

objects by their bounding boxes or center points, but instead uses a transformer architecture for a pixel-wise propagation and association of the pixel-wise distributions of the objects. The Tracktor approach, developed by Bergmann et al. (2019), is a well-known successful implementation of the tracking-by-detection paradigm. Tracktor leverages the regression capabilities of a detection model to facilitate data association, with the significant advantage being that no additional tracking-specific training is required. SparseTrack (Liu et al., 2023b) uses pseudo-depth information to better distinguish between objects in the data association step. This is especially helpful, if there are many objects close to each other. For calculating pseudo-depth, a planar scene is assumed, which, in environments like wildlife areas with hills, may not always be applicable.

Xu et al. (2020a) and Chen et al. (2018a) utilize trained neural networks to match detections across frames. Li et al. (2022b) introduced the IANet, which enhances data association, particularly in scenarios involving object interactions or occlusions, through a geometry refinement network and an identity verification module. Graph-based solutions for linking detections to tracks have been employed by (Ma et al., 2018; Henschel et al., 2018; Sheng et al., 2018). The approach of Brasó and Leal-Taixé (2020) uses a Message Passing Network (MPN) to formulate the tracking problem. They show that with this fully differentiable network learning in multi-object tracking is also possible for the data association and not only the feature extraction. Other notable tracking-by-detection approaches include those by Berclaz et al. (2011); Dicle et al. (2013); Chu et al. (2019).

The single-shot tracker FairMOT (Zhang et al., 2021b), joint detection by embedding (JDE) (Wang et al., 2020f), and the approach by Zhang et al. (2020) perform object detection and Re-ID feature extraction concurrently to track objects within video frames. TransMOT (Chu et al., 2021) and TransTrack (Sun et al., 2020) employ Transformer networks to simultaneously detect and track objects, showcasing the application of advanced neural network architectures in tracking tasks. The Global Tracking Transformer (GTR) (Zhou et al., 2022) analyzes short sequences of frames to form global tracks across these. This tracker can also be trained together with an object detector.

Another special approach to tracking is the DiffusionTrack model (Luo et al., 2024), which uses a denoising technique to perform object detection and data association at once. SiamMOT (Shuai et al., 2021) and SMOT (Li et al., 2020a) utilize motion modeling to track detected objects effectively. MeMOT (Cai et al., 2022) uses a large spatio-temporal memory, where identity embeddings of all tracked objects are stored. With the help of an attention mechanism MeMOT is able to predict the new object states. CenterTrack (Zhou et al., 2020) tracks objects not by their bounding box, but with the center position of the objects. This makes this tracker simpler and faster. The PermaTrack approach (Tokmakov et al., 2021) builds on the idea of CenterTrack and adds a spatio-temporal, recurrent memory. This means that the position of an object in a frame can be determined not only with the information from

the last frame, but with all previous information. Additional single-shot tracking approaches include TrackletNet (Wang et al., 2019), (Li et al., 2022a), and a Graph Neural Network-based approach Wang et al. (2021).

To summarize, the main distinction for multi-object tracking approaches is whether they do a tracking-by-detection approach or if they are single-shot trackers. The tracking-by-detection approaches are the more popular and successful approaches. They profit from the improvement of object detectors because the tracking accuracy is always dependent on the detection quality. For faster inference times a single-shot tracker can be the better choice due to the integrated detection and tracking architecture.

## 2.3 Multi-Object Tracking and Segmentation MOTS

Multi-object tracking and segmentation (MOTS) integrates instance segmentation with multi-object tracking (MOT). MOTS identifies objects in a video using instance masks and associates these masks with their respective trajectories.

Numerous MOTS (Multi-Object Tracking and Segmentation) approaches adopt Mask R-CNN for instance segmentation and develop their tracking frameworks on this foundation. The most basic extension forms MaskTrack R-CNN (Yang et al., 2019a) that extends the Mask R-CNN model by a tracking branch. The SORTS approach (Ahrnbom et al., 2021) focuses on the speed of the video instance segmentation and therefore combines Mask R-CNN with SORT as a tracker and also presents another extension using a Re-ID network. MOTSNet (Porzi et al., 2020) adds a new tracking head to Mask R-CNN and further introduces a mask pooling layer to guide the data association. MaskProp (Bertasius and Torresani, 2020) extends Mask R-CNN by a new propagation branch to propagate instance masks from one frame to all frames of a video. The propagated masks then form the tracks in the video. The work of Lin et al. (2020) implements a variational autoencoder (VAE) on top of a Mask R-CNN. The approach learns spatial and motion features to create representations to detect and track instance masks in videos. (Qi et al., 2021) concentrates on a special video instance segmentation task, where they also consider the occluded parts of instance masks. Therefore, they introduce a novel dataset, where the instance masks are annotated in that way. Moreover, they present a video instance segmentation module, called temporal feature calibration that is a combination of MaskTrack R-CNN (Voigtlaender et al., 2019) and SipMask (Cao et al., 2020) . The authors of (Voigtlaender et al., 2019) extend the architecture of Mask R-CNN by 3D convolutions and an association head. The data association is then performed by computing the Euclidean distance of the generated association vectors. This approach is called TrackR-CNN (which should not be confused with the similar sounding MaskTrack R-CNN (Yang et al., 2019a)).

There are several approaches to video instance segmentation that are based on transformer architectures. In the study by Wang et al. (2020e), the VisTr, a transformer network comprising an encoder and decoder, is utilized to simultaneously perform instance segmentation and tracking. The Minimal Video Instance Segmentation (MinVIS) (Huang et al., 2022) uses an image-based transformer architecture. Therefore, the training process needs less samples and annotations in comparison to other video instance segmentation tasks that learn from video data. The finding that queries, when trained to distinguish between object instances within the same frame, are temporally consistent allows the model to track instances effectively without relying on any manually designed tracking heuristics. The SeqFormer model (Wu et al., 2021) applies the transform attention mechanisms to each frame independently. This is done by detecting an instance in a frame and aggregating the temporal information for learning a video representation. This is the foundation for predicting the instance masks in the following frames. The IDOL framework (Wu et al., 2022b) uses a deformable transformer DETR (Zhu et al., 2020) with contrastive learning. This approach focuses on learning discriminative instance embeddings for data association. The Video Instance segmentation via object Token Association (VITA) approach (Heo et al., 2022) builds on an image transformer and associates the image-based object tokens to form tracks. The Generalized framework for Video Instance Segmentation (GenVIS) approach (Heo et al., 2023) builds on the VITA architecture and uses a novel training strategy that uses multiple clips at once and learns associations between them. Moreover, they introduce a memory to better use information from the past.

Moreover, there are some approaches that create new modules or extend existing frameworks in other ways. The Taxonomy-aware Multi-dataset Joint Training for Video Instance Segmentation (TMT-VIS) (Zheng et al., 2024) introduces two new modules to learn jointly from multiple different datasets. The instance segmentation network SipMask (Cao et al., 2020) can also be extended to the video instance segmentation task by adding a new fully convolutional branch for the tracking vector feature extraction as described by the authors. The Cross-VIS network (Yang et al., 2021) introduces a novel crossover learning scheme. This scheme means that the features of one instance mask in a frame are used to identify the same instance in another frame.

Finally, there are networks and systems specially designed for the task of video instance segmentation. STEm-SEg (Athar et al., 2020) is an end-to-end trainable network designed to process video input as a 3D spatio-temporal volume, where the network learns an embedding for each pixel. PolyTrack (Faure et al., 2021) utilizes center heatmaps to represent detected objects and employs a tracking method similar to that used by CenterTrack (Zhou et al., 2020). The ReMOTS approach (Yang et al., 2020) aims at better learning appearance features for the data association. This is done by training an appearance encoder with predicted masks and short-term tracklets. Therefore, this approach is called self-supervised refining MOTS (ReMOTS). The Space Time Correspondence Network (STCN) (Cheng

et al., 2021b) concentrates on modeling these correspondences for video instance segmentation. They use direct image-to-image correspondences that lead to more efficient and robust matching results. PointTrack (Xu et al., 2020b) uses a tracking-by-points paradigm, where instances are not tracked by image representations but by point cloud representations. The Spatial Granularity Network (SG-Net) (Liu et al., 2021) is a one-stage approach that uses three heads for detection, tracking and segmentation that are jointly optimized and share features. The one-stage design improves the interference speed. EfficientVIS (Wu et al., 2022a) introduces a correspondence learning scheme, which enables a correlation between instance tracklets between clips without the need for an explicit data association.

To summarize, the task of MOTS combines the tasks of instance segmentation and tracking. Therefore, many approaches build on a reliable instance segmentation approach, the Mask R-CNN. This basis is extended in various ways to integrate the tracking ability. Another important field of approaches uses transformer models for the instance segmentation and the tracking part. Moreover, there are various approaches that extend other specialized instance segmentation models to the MOTS task.

## 2.4 Action Recognition and Action Detection

Numerous approaches to action recognition and action detection have emerged in recent years, as detailed in several comprehensive surveys (Simonyan and Zisserman, 2014; Zhang et al., 2019; Bhoi, 2019; Kong and Fu, 2022; Liu et al., 2022).

3D convolutional neural networks (3D CNNs) are a prominent method for extracting spatio-temporal features from videos to classify actions (Tran et al., 2015; Carreira and Zisserman, 2017; Tran et al., 2018; Hara et al., 2018; Wang et al., 2018b). These networks have proven to outperform 2D convolutional approaches (Zhou et al., 2018; Wang et al., 2018a) in action recognition. The X3D model (Feichtenhofer, 2020) expands 2D convolutional networks across space, time, width, and depth to better capture spatio-temporal features needed for accurate action recognition. A significant advancement in 3D CNNs is the adoption of two-stream approaches, initially introduced by Christoph and Pinz (2016). One of the most successful implementations of this concept is the SlowFast network (Feichtenhofer et al., 2019), which separates the processing of spatial and temporal features into two distinct streams. Sheth (2021) further evolve this idea with a three-stream network that uses varying frame counts across streams, integrating the outputs via an LSTM (Long Short-Term Memory) network for action prediction. Additionally, a three-stream convolutional neural network (3SCNN) that utilizes skeleton data for action recognition in 3D has been proposed by Liang et al. (2019). The approach of (Wang et al., 2024) uses a knowledge distillation approach with a generative network. Initially, a feature

representation is learned by generative model-based attention module. After this, a student network is trained with the help of the distilled features.

There are various methods that build on existing CNN architectures to enhance their capability to extract spatio-temporal features from video data. Notable examples include Temporal Relation Network (TRN) (Zhou et al., 2018), SpatioTemporal Module (STM) (Jiang et al., 2019), Temporal Shift Module (TSM) (Lin et al., 2019), Temporal Excitation and Aggregation (TEA) (Li et al., 2020b), MotionSqueeze (Kwon et al., 2020), and TokenLearner (Ryoo et al., 2021). Additionally, the Contrastive Action Representation Learning (CARL) framework (Chen et al., 2022) is specifically designed to learn long-term action representations, focusing on capturing complex activity patterns over extended periods.

Transformer architectures have gained significant popularity for action recognition, leveraging the attention mechanism (Vaswani et al., 2017; Yan et al., 2022; Arnab et al., 2021a). One particularly successful example is the Multiscale Vision Transformer (MViT) (Fan et al., 2021), which combines the transformer architecture with hierarchical multiscale feature extraction. The ActNetFormer (Dass et al., 2024) combines a CNN with a transformer, where the CNN focuses on extracting spatial and (local) temporal features while the transformer especially extracts long-range temporal features. The Cross Attention in Space and Time network (CAST) (Lee et al., 2024) uses two frozen transformer networks, where one forms an expert for extracting spatial features and the other one is an expert for temporal feature extraction. Both transformer networks exchange information and in the end generate a joint prediction.

The application of instance masks to enhance action recognition is a relatively underexplored field, with existing approaches typically confined to specific application areas or possessing certain limitations. SegCodeNet (Sushmit et al., 2020) employs a Mask R-CNN (He et al., 2017) to generate instance masks and incorporates these into a two-stream approach alongside input frames to recognize various activities associated with wearable devices. Similarly, the study of Zaghbani and Bouhlel (2021) utilizes a Mask R-CNN to create instance masks for a single actor in video sequences, which are then used to mask the frames before they are processed by a single-stream CNN for action recognition. Another approach by Hacker et al. (2023) combines RGB frames with human pose information in a two-stream network to analyze actions in table tennis, illustrating the diverse applications of these techniques in action recognition scenarios.

The most direct way to perform action detection is to combine a detector with an action recognition network. However, since these tasks are often analyzed and evaluated independently, comprehensive designs for action detection are relatively scarce in comparison to action recognition approaches in the literature. Additionally, the effectiveness of the action recognition component is heavily dependent on the quality of the initial detection when both components are evaluated collectively. There are also approaches that combine the detection and action recognition in one

network. Despite these challenges, there are notable action detection methodologies. The Asynchronous Interaction Aggregation network (Tang et al., 2020) is designed to capture interactions between individuals. Biswas and Gall (2020) introduced a baseline method for weakly supervised action detection that can handle scenes with multiple actors. The Efficient Video Action Detector (EVAD) (Chen et al., 2023) employs a transformer architecture to accurately localize actors and categorize their actions. The approach of (Arnab et al., 2021b) uses a message passing graph neural network to represent the spatio-temporal relations. The approach can be trained with and without supervision.

There are action detection approaches that train one backbone network for combined localization and action prediction. The Video Action Transformer (Girdhar et al., 2019) uses a transformer to aggregate spatio-temporal features from the person, whose action they want to predict. The attention mechanism learns to concentrate the feature extraction on hands and face, which are most important for action recognition of humans. Another transformer based approach is TubeR (Zhao et al., 2022a), which builds on the DETR transformer and extracts features from action tubelets for localization and action recognition. They introduce a context aware classification head to better detect the start and end point of actions. The STMixer network (Wu et al., 2023) builds on the transformer backbone of ViT (Dosovitskiy et al., 2020). The STMixer is a sparse action detection approach, where the core concept are learnable queries to decode all action instances in parallel. The Watch Only Once (WOO) network (Chen et al., 2021) also uses one backbone for detection and action recognition. The used backbone is mixture between the SlowFast backbone and the Faster R-CNN. Moreover, the authors introduce spatio-temporal action embeddings in the pipeline and use a fusion module. The approach of (Arnab et al., 2022) uses a co-finetune strategy to train the transformer backbone simultaneously on different classification and detection datasets. With this rather simple approach they reach accuracies similar to more complex long term memory concepts.

The already mentioned SlowFast network (Feichtenhofer et al., 2019) also incorporates capabilities for action detection, combining a detector similar to Faster R-CNN within its framework to identify actors before recognizing their actions. Additionally, Yuan et al. (2020) combine a SlowFast network with a human detector to specifically target action detection involving people.

To summarize, there are many different approaches to performing action recognition. While the first successful approaches were based on 2D convolutions, the current models are either built from 3D convolutions or consist of transformer architectures. Important improvements within the models include the use of multiple streams, the use of memory modules and the introduction of new module components within the network layers. The task of action detection is often a combination of a successful detection network with one of the mentioned action recognition networks. But there are also specially designed action detection systems, for example, models that use transformer architectures. Similar to the multi-object tracking the action recognition

is dependent on the detection quality. Therefore, approaches that divide both tasks in separate approaches are more prominent.

## 2.5 Instance Segmentation, MOT, MOTS, Action Recognition and Action Detection in Wildlife Monitoring

There are very few works that consider instance segmentation, MOT, and MOTS in the field of wildlife monitoring. Many approaches for image classification exist (Chen et al., 2014; Okafor et al., 2016; Villa et al., 2017; Norouzzadeh et al., 2018; Willi et al., 2019; Chen et al., 2019b). Moreover, there exist several approaches to detect animals with bounding boxes in images (Verma and Gupta, 2018; Beery et al., 2018, 2019; Falzon et al., 2020; Bonneau et al., 2020; van der Zande et al., 2021). Classification and detection of animals in images is useful, but basic task.

Camera trap videos offer significantly more information that can be extracted and processed by more complex computer vision techniques to facilitate ecological studies. The approach of Yang et al. (2019b) presents a detection pipeline that utilizes attention-based blending of spatial and temporal features to detect gorillas in camera trap footage. While this method detects apes using bounding boxes, it does not incorporate tracking or instance masks. Instance segmentation on images of cows is explored by Ter-Sarkisov et al. (2018); Salau and Krieter (2020); Bello et al. (2021). Hu et al. (2021) introduce a dual attention-guided feature pyramid network to perform instance segmentation on images of pigs. Le et al. (2021) create a dataset containing images of camouflaged animals in nature and a specialized instance segmentation approach, the Camouflaged Fusion Learning (CFL) framework that is optimized for the task of camouflaged instance segmentation.

Multi-object tracking has predominantly been applied to small animals such as fish or insects in controlled laboratory settings, as explored by (Fukunaga et al., 2015; Rodriguez et al., 2017; Sridhar et al., 2019). In a different context, Zeppelzauer (2013) employed a method to track elephants in wildlife videos by constructing a connectivity graph for detected elephant segments, although they did not provide MOT metric values to evaluate tracking accuracy. Gan et al. (2022) utilized a graph convolutional neural network for instance segmentation and tracked objects by analyzing the intersection over union of the detections, eventually identifying actions of piglets in a static indoor environment, which differs significantly from the dynamic conditions of wildlife videos in varying terrains.

To the best of our knowledge, the multi-object tracking and segmentation (MOTS) task has not been extensively explored in the context of wildlife monitoring in previous studies. The challenge in wildlife monitoring is particularly pronounced due to the similar appearance of animals, especially when they are in close proximity. This

is a stark contrast to person datasets where individuals are often distinguishable by their clothing. This similarity complicates the task of achieving precise instance segmentation and dependable tracking. The study by Xue et al. (2021) is the most similar to addressing the MOTS problem in this context. However, their approach focuses on segmenting and tracking only one animal at a time within a video. Additionally, their method necessitates user intervention to set the instance mask of the animal in what they refer to as a 'guidance frame', which may not be practical for extensive wildlife monitoring efforts that aim for automation and scalability.

Action recognition and action detection for wildlife monitoring remains still a relatively unexplored research area. The following approaches represent important publications in the field of action recognition and action detection in wildlife monitoring. Sakib and Burghardt (2020) develop a neural network for action recognition of great apes using a self-created dataset. The Animal Kingdom Dataset is introduced by Ng et al. (2022), which consists of action clips of a huge variety of animal species. Moreover, the authors introduce an action recognition network, the Collaborative Action Recognition network (CARe) that recognizes actions of unseen animal species. In the paper (Brookes et al., 2023) the authors introduce a metric learning approach employing a triple-stream embedding network that utilizes RGB and optical flow features to recognize actions of great apes.

With our previous publications we contributed to the tasks of instance segmentation (Schindler and Steinhage, 2021a,b, 2022, 2023), multi-object tracking (Schindler and Steinhage, 2021b, 2022, 2023) and action recognition and action detection (Schindler and Steinhage, 2021a; Schindler et al., 2024).

To summarize, research in the three fields of our work, instance segmentation, multi-object tracking and action detection, in the context of wildlife monitoring are very scarce. There are many works, which concentrate on classifying images and detecting animals in images. Video data is mainly considered in closed environments like laboratories or farms, where animals are tracked and detected. There are only a few works that consider action recognition for apes.

## 2.6 Where do SWIFT and MAROON fit into the Related Work?

Our entire workflow is an action detection system, which in turn consists of two parts: SWIFT, which initially performs multi-object tracking and segmentation, and MAROON, which performs action recognition. The instance segmentation part of SWIFT consists of a Mask R-CNN and a refinement network. Therefore, it is a two stage approach and builds on the famous Mask R-CNN like other instance segmentation approaches. We further combine this with a refinement algorithm that we designed. In this refinement step we use a refinement network, which can

also be a foundation model for generating very exact masks. The SWIFT Tracking Algorithm is a tracking-by-detection approach because we want to generate exact tracks and do not perform an online tracking. Our design to use multiple filtering stages, where tracklets are combined and deleted is novel. The basis tracklets are generated from a combination of a motion model and the Hungarian matching, which resembles the SORT based approaches. Both parts of SWIFT together form a MOTS system. Accordingly, our approach is one of the approaches based on a Mask R-CNN. However, as just described, the two components of detection and tracking are separate in our case. This modularity makes it easier to modify our system, for example to benefit more easily from future improvements to foundation models.

MAROON is our action recognition network and builds on the SlowFast architecture. So our approach uses 3D convolutions and the idea of multiple streams. In comparison to SlowFast, we extend the two-stream approach to triple-stream approach for extracting motion and appearance features with a finer granularity. Another important improvement is the use of instance masks to cut out the animal, allowing the network to focus on the actor. The combination of SWIFT and MAROON forms our overall workflow, which is able to perform action detection. Here, too, we follow the idea of modularity, so that we implement our subsystems sequentially one after the other. This allows us to better control the dependency of action recognition on the detector and, similar to MOTS, to better benefit from future improvements to individual components.

In our application area, we are the first to examine all these problem statements at once. This makes it possible to combine the findings from the different areas. As we show in this thesis, instance segmentation enables improved tracking and action recognition. Even for the partial tasks of instance segmentation, multi-object tracking and action recognition there are not really comparable approaches for the field of wildlife monitoring. Therefore, our system is a novelty in wildlife monitoring and enables researchers to analyze camera trap videos easier and better.

# Chapter 3

# Datasets and Data Acquisition

Annotated datasets for the combined tasks of instance segmentation, MOT, MOTS and action detection in the context of wildlife monitoring, to the best of our knowledge, do not exist or are not publicly available. There are also no annotated datasets for each individual task. Therefore, we created our own datasets, the Rolandseck datasets, for this thesis. In addition, we were able to obtain data material from the cooperation with the Nationalpark Bayerischer Wald, which forms the Bavarian Forest datasets. We divide these two datasets into a daylight and a nighttime dataset. In addition, we analyze the Wildlife Crossings dataset, which consists exclusively of nighttime images. In total, we examine the five datasets Rolandseck Daylight, Rolandseck Nighttime, Bavarian Forest Daylight, Bavarian Forest Nighttime and Wildlife Crossings. In Table 3.1, we compare the most important aspects from our five datasets. In the following sections, we describe our datasets in detail.

## 3.1 Rolandseck Datasets

With permission of the Wildpark Rolandseck GmbH, we captured video footage of fallow deer *(Dama dama)* and red deer *(Cervus elaphus)* in their natural environment from November 2020 to December 2021, resulting in over 6000 recorded videos. We first had to view the data material to filter out videos that were not usable (e.g. no animals are present, camera was tilted extremely). This initial filtering resulted in about 3000 remaining videos of varying quality. The data annotation process is described in detail in Section 3.4. Because the data annotation forms a bottleneck we had to choose representative videos for different scenarios, backgrounds, animals present and actions. Within the 3000 videos, there are often scenes that show similar situations and behavior. Using similar videos is not helpful for training neural networks.

For the recordings we used two *Victure HC500 Trail Cameras*. Our camera traps are shown in Figure 3.1. On the left picture, the camera is open so that settings for

| Dataset | No. of videos / No. of Frames | Frames per second (FPS) | Resolution | Avg. no. of animals per video | Different action classes |
|---|---|---|---|---|---|
| Rolandseck Daylight | 34 / 27,979 | 30 | 1728×1296 | 5.88 | 11 |
| Rolandseck Nighttime | 34 / 27,025 | 30 | 1728×1296 | 2.91 | 9 |
| Bavarian Forest Daylight | 54 / 19,469 | 15 | 1280 × 720 | 1.13 | 7 |
| Bavarian Forest Nighttime | 31 / 5,789 | 8 | 1280 × 720 | 1.06 | 7 |
| Wildlife Crossings | 41 / 2,506 | 8 | 1280 × 720 | 2.15 | 7 |

**Table 3.1:** Comparative overview of the five datasets used. We compare the technical aspects of the videos, such as the size of the datasets and the recording quality. We also look at the average number of animals and the number of different action classes.

the recordings can be made and already created recordings can be viewed on the small display. The cameras are waterproof and inexpensive to buy (about 100 euros including the necessary batteries and SD card). All videos are 30 seconds long with 30 fps (frames per second) and a high definition resolution of 1728 x 1296 pixels. Videos in the dataset were only shortened if there were no animals visible in the clips anymore. Our camera traps are equipped with PIR (Pyrocelectric Infrared) sensors, which detect temperature changes in its field of view. Therefore the cameras are only triggered when an animal moves into the field of view (not for example by a moving branch). The camera traps were placed at different locations to show changing backgrounds and different settings. The general site locations stayed the same, but we changed the exact position of the cameras (e.g. changed the tree, where the camera is mounted) each months to prevent always showing the same background.

As already mentioned, we split our annotated videos in two separate datasets depending on the recording time. The Rolandseck Daylight dataset consists of 34 annotated videos. These videos are all captured during daylight. The following 11 different actions are present in the videos: foraging moving, foraging standing, grooming, head lowering, head raising, resting, running, standing up, vigilant lying, vigilant standing, walking. A definition of all action classes is given in Table 3.2. The Rolandseck Nighttime dataset consists of 34 annotated videos. These videos are

**Figure 3.1:** Our *Victure HC500* camera traps mounted on the trees in the wildlife park Rolandseck (Germany).

all captured during nighttime and therefore show no color information. In Figure 3.2 we show exemplary frames of both datasets showing fallow deer and red deer at day and night.

The Rolandseck datasets contain high-resolution images with a high frame per second rate compared to the other datasets (see Table 3.1). In addition, larger groups of animals appear in these datasets. This is a major challenge, particularly for detection and tracking because many animals move close to each other and thus partially or even completely occlude each other. In general, animals camouflage themselves with the environment through their fur color, which makes detection more challenging than detection in a person dataset, where each person wears different clothing (Le et al., 2021). The night images are a challenge overall due to the lack of color information. In addition, the PIR sensor illuminates the center of the image more than the edges. As a result, animals in the middle of the image appear brighter and animals are more difficult to recognize when entering and leaving the scene. This can also be seen in the sample images in Figure 3.2.

## 3.2 Bavarian Forest Datasets

The datasets Bavarian Forest Daylight and Bavarian Forest Nighttime were captured in the Nationalpark Bayerischer Wald at different sites. The video material was selected in cooperation with ecologists from the Bavarian Forest National Park. The description of the action classes was carried out by the ecology experts and the


**Figure 3.2:** Exemplary frames from Rolandseck Daylight (top) and Rolandseck Nighttime (bottom). Fallow deer are shown on the left and red deer on the right.

data annotation was done jointly. The videos show roe deer *(Capreolus capreolus)* and red deer *(Cervus elaphus)*. These videos were selected from a larger dataset based on the number of individuals and different types of behavior present. The daylight videos are recorded with 15 FPS and a resolution of 1280 x 720. The nighttime videos have the same resolution, but only 8 FPS. The Bavarian Forest Daylight dataset contains 54 annotated videos and the Bavarian Forest Nighttime dataset consists of 31 annotated videos. The following 7 different action classes are represented in this dataset: foraging standing, grooming, head lowering, head raising, vigilant standing, walking, sudden rush. In Figure 3.3 we show exemplary frames of both datasets showing roe deer and red deer at day and night.

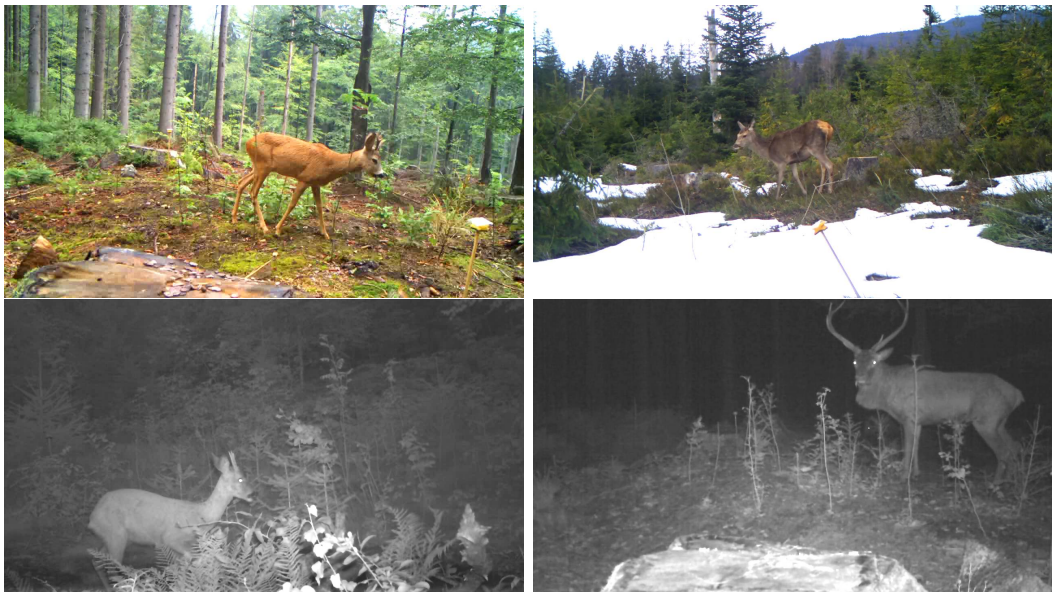


**Figure 3.3:** Exemplary frames from Bavarian Forest Daylight (top) and Bavarian Forest Nighttime (bottom). Roe deer are shown on the left and red deer on the right.

In contrast to the Rolandseck Daylight dataset these two datasets generally show a lower number of individuals per video. The reason for this is that the recordings were taken in a national park that does not have a narrow spatial boundary like the wildlife park. This represents an important difference that will be of interest when investigating the influence of instance masks for our system. Moreover, the resolution of the frames and the frames per second rate is lower than the values of the Rolandseck datasets.

## 3.3 Wildlife Crossings Dataset

The fifth dataset that we consider is the Wildlife Crossings dataset. The data material was recorded by camera traps that were positioned at wildlife crossings

at the federal motorway 7 near the city Oberthulba. The video data was provided by the Bavarian Highway Directorate, Germany, but we had to manually annotate all of the videos with the instance masks, track IDs, animal classes and action labels. All videos are recorded at nighttime. Therefore, we do not divide this dataset in a Daylight and Nighttime dataset. Each video is about 10 seconds long with 8 fps (frames per second) and a resolution of 1280 x 720 pixels. The videos include red deer *(Cervus elaphus)*, wild boar *(Sus scrofa)*, hares *(Leporidae)* and foxes *(Vulpes vulpes)*. The Wildlife Crossings dataset consists of 41 annotated videos. The following 7 different action classes are represented in this dataset: foraging standing, foraging standing, head lowering, head raising, vigilant standing, walking, running. In Figure 3.4 we show exemplary frames of all 4 different animal classes of the Wildlife Crossings dataset.

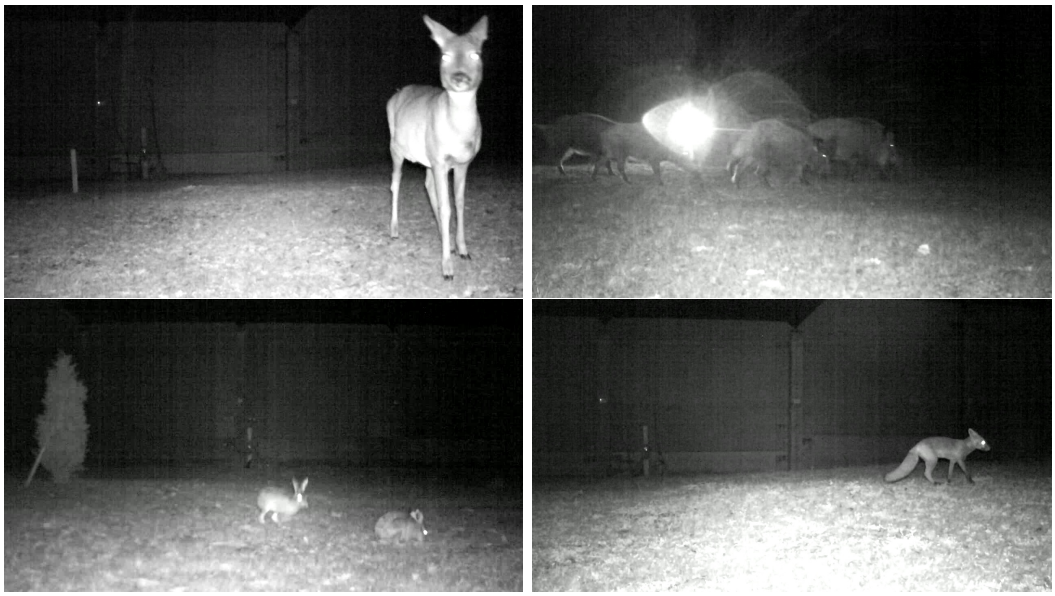


**Figure 3.4:** Exemplary frames from Wildlife Crossings dataset. All videos are nighttime videos. The 4 occurring animal classes are red deer (top left), wild boar (top right), hares (bottom left) and foxes (bottom right).

The Wildlife Crossings dataset contains the most different species from all five considered datasets. This is especially interesting for the classification aspect of the instance segmentation. Moreover, the videos have a rather low resolution, which makes it difficult to recognise the animals, and at the same time they have a low temporal resolution of 8 fps, which in turn leads to blurring when the animals move quickly. Due to the special scene of wildlife crossing over a highway, groups of animals also appear more frequently here (in contrast to the national park), usually running through the picture from left to right or vice versa. In addition, for better illumination of the scene a flashlight on the opposing side of the camera is used (cf. example image of wild boars in Figure 3.2). However, this also leads sometimes to very bright points of light in the videos, which can also cause difficulties

for detection.

## 3.4 Data Annotation

The annotation of data material is an often underestimated task, which is particularly important when application areas are examined that have not yet been intensively explored. As already mentioned, there are few to no annotated publicly accessible datasets in wildlife monitoring, especially for the objectives considered in this work. We therefore had to carry out the data annotation ourselves.

For our tasks we need to assign (1) a segmentation mask, (2) a bounding box, (3) a class label, (4) a track ID and (5) an action label to each animal instance in each frame of each video. Taking this into account, the annotation of a single video with less than 10 animals, for example, 30 seconds long and recorded at 30 FPS, already means the assignment of several thousand instance masks and labels. One advantage of this is that the animals usually only change slightly between two frames, so that copying the previous annotation to the next frame is usually very advantageous.

The programs and methods available for annotating instance masks and track IDs have developed considerably in recent years. At the beginning of our work, we used the VGG Image Annotator (VIA) Version 2.0.8 (Dutta and Zisserman, 2019). This tool allows the user to set the contour of the animal as a polygon line. The user can copy the annotated animals to the next frame. But even a slight movement of the animal requires the user to manually shift many points of the polygon line. In 2021, the interactive tool RITM (Reviving Iterative Training with Mask Guidance for Interactive Segmentation) was published by Sofiiuk et al. (2022). This annotation tool supports the user with an AI approach. The user sets positive and negative clicks to describe which regions are part of an object and what are not. These clicks are fed into a trained HR-Net (High-resolution Network) (Wang et al., 2020a) to create and adapt the instance masks. This tool was further improved with the Simple Click approach (Liu et al., 2023a). The user interface has remained the same, but the underlying AI support is now a transformer model. Both approaches are designed for the annotation of single images (not videos).

We have improved the approach of (Liu et al., 2023a) to make it suitable for annotating videos. Therefore, we enable the loading of all frames from one folder. We introduce some short-cut key bindings to enable fast annotation. So it is possible to load the next or previous frame and to switch between the currently annotated object mask. In addition, we add a short-cut key binding for saving the annotated mask to save time. When the user switches to the next or previous frame the current annotations are copied. This new feature is very helpful for video annotation because the animals only move slightly between two frames. Moreover, the copied masks are fed into the HR-Net (respectively the transformer model) as initialization for a more robust prediction. In Figure 3.5 we show the user interface of RITM / Simple Click

and highlight our additions in red. On the right side we add information about the current frame and annotation status. The file name of the current frame is displayed and the number of the frame from the entire video. It also shows which animal or instance mask is currently being edited. Below this, one new button has been added that allows the correction mode to be switched on and off. In the correction mode the program loads the already annotated masks and allows the user to correct faulty annotations. Moreover, we added the continue function on the left side. This button automatically loads the last frame, where an instance mask is available. This is very useful and time-saving if the annotation process is interrupted and continued later. Without this button, the last frame must be manually selected and loaded and then the appropriate instance mask loaded.

The improvements concerning the correction mode and continue button were done in cooperation with the student Benedikt Wude during his project group in the summer term of 2023 (cf. Acknowledgements).


**Figure 3.5:** The user interface of the annotation tool RITM / Simple Click is shown. We have highlighted our improvements and additions to the user interface in red. We have added the Continue button at the top. On the right-hand side, we have added a display specifically for videos that shows the current frame of the video and which animal is currently being annotated. Below this is the new button for switching to Correction mode.

There also exist annotation tools that are designed for video annotation. The tool MIVOS (Cheng et al., 2021a) is also an interactive instance segmentation tool. The user annotates one frame with positive and negative clicks supported by AI. Then this information is propagated for the next frames to interpolate the annotation. This prediction does not work really well for longer periods of time and is also faulty if animals move close to each other. In our publication (Schindler and Steinhage, 2021b) we used a combination of the Mask R-CNN (He et al., 2017) and the Tracktor

approach (Bergmann et al., 2019) to automatically annotate videos. The instance segmentation approach has to be trained initially. Therefore, it is necessary that at least a few videos are manually annotated before. A general problem with the automatic annotation of frames is that even a 95% correctly annotated object or animal may require post-processing by the user. For example, if an animal's ear is not annotated, this cannot be learned correctly by a neural network later on. However, accurately annotated animals are even more important for a correct evaluation of the models. For those reasons, it is difficult to completely automate the annotation. Thus, interactive instance segmentation supported by AI is currently the best solution.

The Rolandseck datasets were annotated by us and also by students working in projects in our working group (cf. Acknowledgements). The Bavarian Forest datasets were annotated by us and by a student from the research group of the national park Bavarian Forest (cf. Acknowledgements). The annotations of the Wildlife Crossings dataset were done by us.

### 3.4.1 Action Definitions and Action Distributions

For the task of action detection, the actions of all individuals must be described by a start and end time. However, in order to train an action recognition network, further pre-processing of the data material must be done. The videos from the datasets can contain multiple animals and the animals perform different actions throughout one single video. To be usable for action recognition, each sequence must contain exactly one animal that performs only one action. Therefore, we extract each animal individually from the videos using the given bounding box information. We then split this video of the individual animal at the time points where the animal's action changes. Thus, from the complete videos of the datasets, many (action) sequences are created, which show only one animal performing only one action during the whole sequence. The length of the sequences can vary from a short section (1 second) to a total video length (30 seconds). It depends only on how long an action is performed. In Figure 3.6 we show the distribution of the action class sequences in our datasets. As it is common in the field of action recognition, especially in the field of wildlife monitoring, long tailed distributions are present in all datasets (Zhang et al., 2023). This means that the action classes are not evenly distributed in the dataset. For example, the classes walking, vigilant standing, and foraging standing are significantly more common than the classes grooming, resting, or sudden rush in all datasets.

The Table 3.2 provides an overview of all 13 different action classes that occur and a description of them. Both the action classes and the descriptions were created in cooperation with ecologists from the Bavarian Forest National Park who are involved in wild animal behavior research.

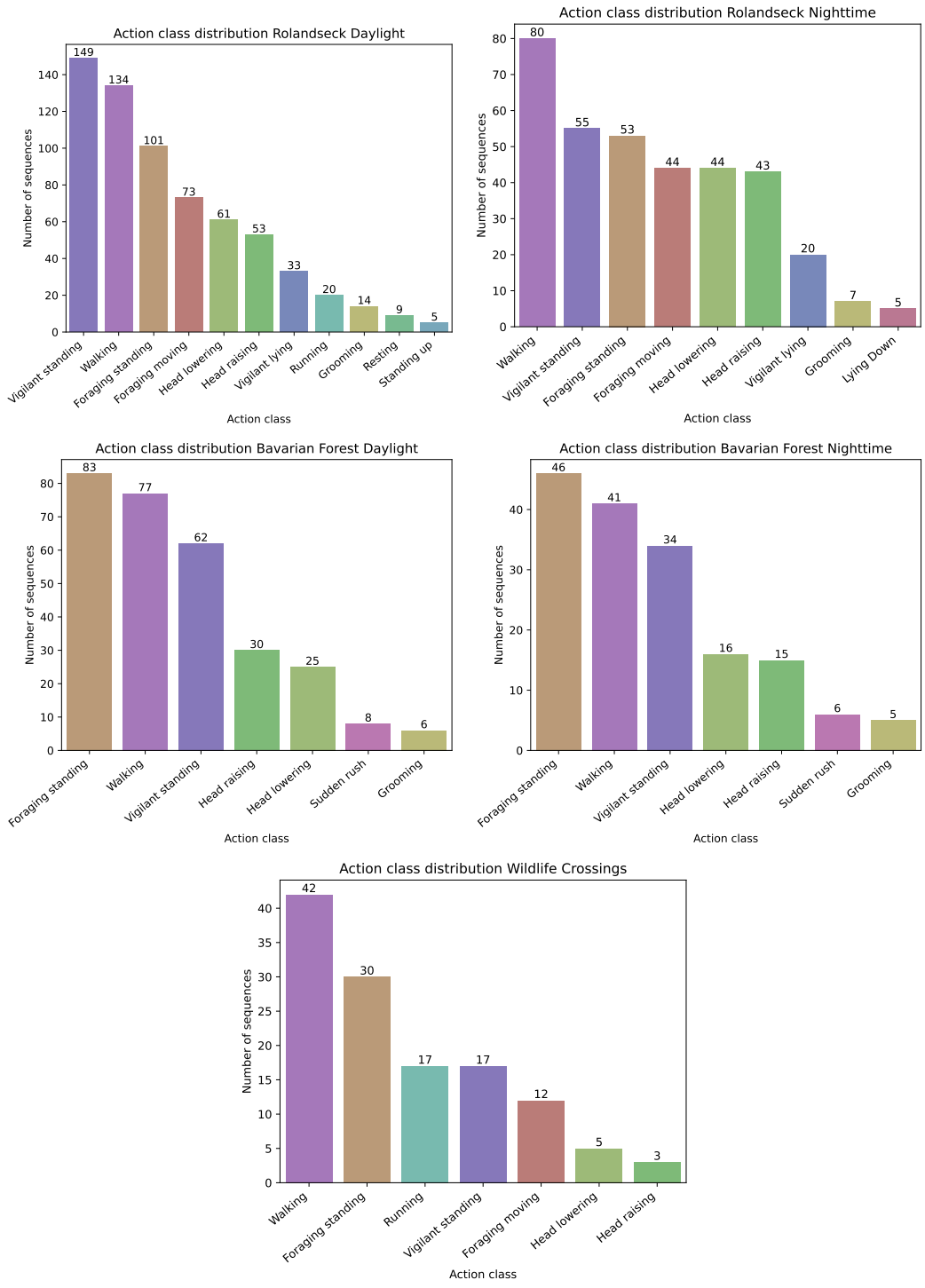



**Figure 3.6:** Action class distribution of the datasets Rolandseck Daylight, Rolandseck Nighttime, Bavarian Forest Daylight, Bavarian Forest Nighttime and Wildlife Crossings. The same action classes in different datasets are depicted in the same color.

| Action class | Description |
|---|---|
| Foraging moving | The animal is in motion or walking with its head lowered toward the ground or a food source such as bushes, its eyes open. Its mouth is close to the ground or the food source, and its jaws may be moving. |
| Foraging standing | The animal is standing with its head lowered toward the ground or a food source such as bushes, its eyes open. Its mouth is close to the ground or the food source, and its jaws may be moving. |
| Grooming | The animal is standing and either scratching or licking itself. Its mouth is in contact with a random part of its body, and its head is moving slightly. |
| Head lowering | The animal's head transitions from a raised position (aligned with or above the body) to a lowered position (closer to the ground or nearly in line with the body) within a brief time. |
| Head raising | The animal's head is moved from a lowered position (close to the ground, up to nearly in line with the body) to a raised position (aligned with the body or higher) within a brief period. |
| Lying down | The animal shifts its posture from standing to lying down. |
| Resting | The animal is lying on the ground with its torso slightly to the side. Its legs may be stretched away from the body or bent underneath and beside it. The head is resting on the ground, and the animal's eyes may be either closed or open. |
| Running | This describes a swift, purposeful forward movement that is faster than walking. The animal maintains a relatively tense posture with its head raised and eyes open. At any given time, at least two hooves are off the ground. This movement ranges from a trot to full speed. |
| Sudden Rush | The animal transitions directly from standing to running, without walking, in less than one second. |
| Standing up | The animal moves from a lying position to standing up. |
| Vigilant lying | The animal is lying down with its head held high, occasionally turning its head and twitching its ears. |
| Vigilant standing | The animal is standing with a tense posture, its head held parallel to the body or higher. It looks around and occasionally twitches its ears. |
| Walking | This describes a relatively slow, purposeful forward movement not associated with feeding or chewing. The animal maintains a relaxed posture with its head held parallel to the body or higher. At all times, at least one hoof is off the ground. |

**Table 3.2:** Definition of all occurring action classes.

# Chapter 4

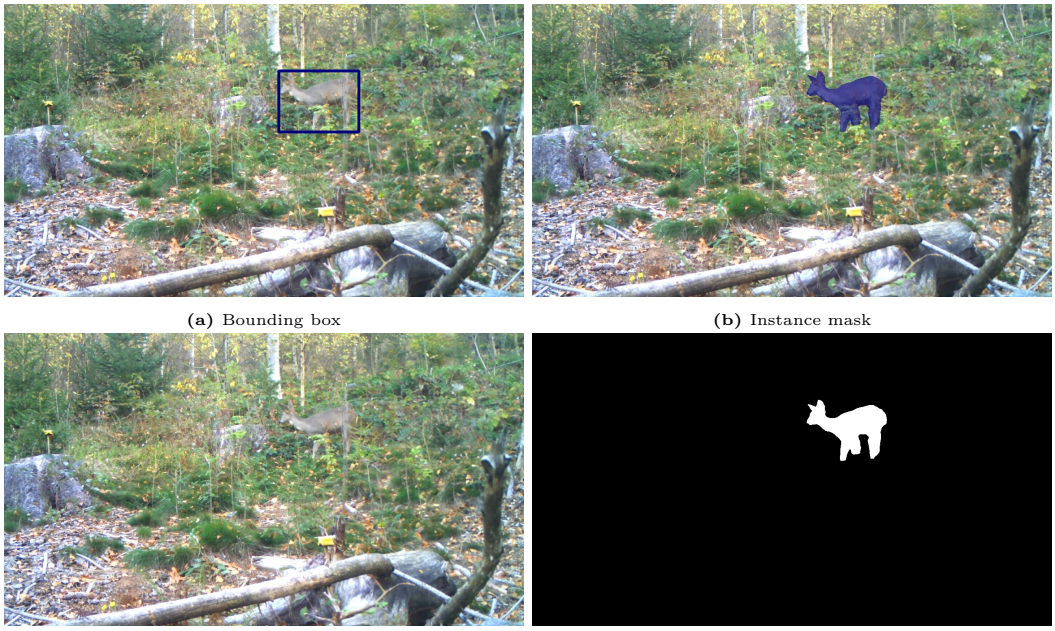
# Instance Segmentation and Tracking

In this chapter, we present our approach for instance segmentation and tracking of wild animals. The goal of instance segmentation is to detect all animals in a frame by assigning (1) a class label, (2) a bounding box, (3) an instance mask and (4) a score value. Tracking links the detections of the instance segmentation in a video to tracks by assigning (5) a unique track ID to each animal in each frame of the video. So the combined task is called MOTS, multi-object tracking and segmentation.

Most of the results of this chapter are published in (Schindler and Steinhage, 2022) and (Schindler and Steinhage, 2023).

## 4.1 Introducing Bounding Boxes, Instance Masks and Tracks

The goal of object detection is to localize all desired objects in an image or, in our case, a frame of a video by bounding boxes and to assign class labels to these objects. Each bounding box is usually described by the pixel positions of the top left and bottom right corners of the rectangle, $b = (x_1, y_1, x_2, y_2)$. However, there are also alternative representations in which a bounding box is described by its center point and the height $h_b$ and width $w_b$ of the box, $b = (x_c, y_c, w_b, h_b)$. The second definition is useful, for example, for tracking using the Kalman filter or the particle filter. A conversion between the two forms of representation is therefore often helpful. However, we will use the first definition for detection. In Figure 4.1 (a) we show a bounding box for the animal in image (c). Each detection also gets a score value $s$ with $0 \leq s \leq 1$ to describe its reliability. Object classification is often included in object detection. This means that each detection gets a class label $c$. Therefore, each detection $d$ is described by the tuple $(b, c, s)$, which consists of the bounding box $b$, the class label $c$ and the score value $s$.



**(a)** Bounding box **(b)** Instance mask

**(c)** Image **(d)** Binary mask

**Figure 4.1:** The bounding box (a) is a rectangle containing the animal. The instance mask (b) describes the exact contours of the animal, which enables more precise localization. The instance mask is often represented by a binary mask (d) that is saved separately from the image (c).

Instance segmentation extends this task by localizing the objects with their exact contour and not only the bounding box information. In Figure 4.1 (a) and (b) we show visually the difference between a bounding box and an instance mask. For an instance segmentation, each detection additionally gets an instance mask $m$. The instance mask $m$ is typically a binary mask, where each pixel that is part of the desired object is assigned a 1 (or 255 depending on the range of pixels) and all other background pixels are assigned a 0. This representation is also shown in Figure 4.1 (c) and (d). This results in the definition of a detection $d$ in the context of instance segmentation as the tuple $(b, c, s, m)$.

Multi-object tracking aims to detect and track all objects of a video. Tracking connects the detections of the same instance from different frames to form tracks. This allows to follow the same object in multiple frames, which is essential to count objects in videos or form a basis for other tasks like action detection. Therefore, each detection $d$ is assigned a track ID $t$.

Multi-object tracking and segmentation (MOTS) extends this task by including the instance mask in this process. In Figure 4.3 we show the difference between multi-object tracking and MOTS. The representation of a detection $d$ for the task of multi-object tracking and segmentation is the tuple $(b, c, s, m, t)$, which consists of the bounding box $b$, the class label $c$, the detection score $s$, the instance mask $m$ and the track ID $t$.

**(a)** Object detection **(b)** Instance segmentation

**Figure 4.2:** For an object detection (a) each object is localized with a bounding box. The instance segmentation (b) adds an instance mask for each object. Often both approaches include an object classification, which results in the written object class above the bounding box.

**(a)** Multi-object tracking

**(b)** Multi-object tracking and segmentation (MOTS)

**Figure 4.3:** The Multi-Object Tracking (MOT) (a) combines detections from different frames and connects the same instances to tracks. This is done by assigning a track ID to each detection and shown visually by assigning the same color to the same instances. The Multi-Object Tracking and Segmentation (MOTS) (b) includes the instance masks in the tracking process.

## 4.2 Segmentation With FIltering of Tracklets - SWIFT

This section provides a summary of our Multi-Object Tracking and Segmentation (MOTS) approach SWIFT. The workflow of SWIFT is depicted in Figure 4.4. Initially, SWIFT conducts instance segmentation with our refinement algorithm (cf. Section 4.3) to accurately detect all animals and their precise outlines. Subsequently, our tracking algorithm (cf. Section 4.4), comprising multiple filtering stages, is applied to link detections across all video frames into continuous tracks, ensuring seamless tracking throughout the video sequence.

In the Figure 4.4 referencing SWIFT's workflow, the algorithms responsible for each step are marked in blue. We will outline the technical specifics of these algorithms using pseudo-code in the forthcoming sections. Moreover, we'll delve into the functionality and underlying design principles of each algorithm, providing a detailed, line-by-line explanation within the respective sections.

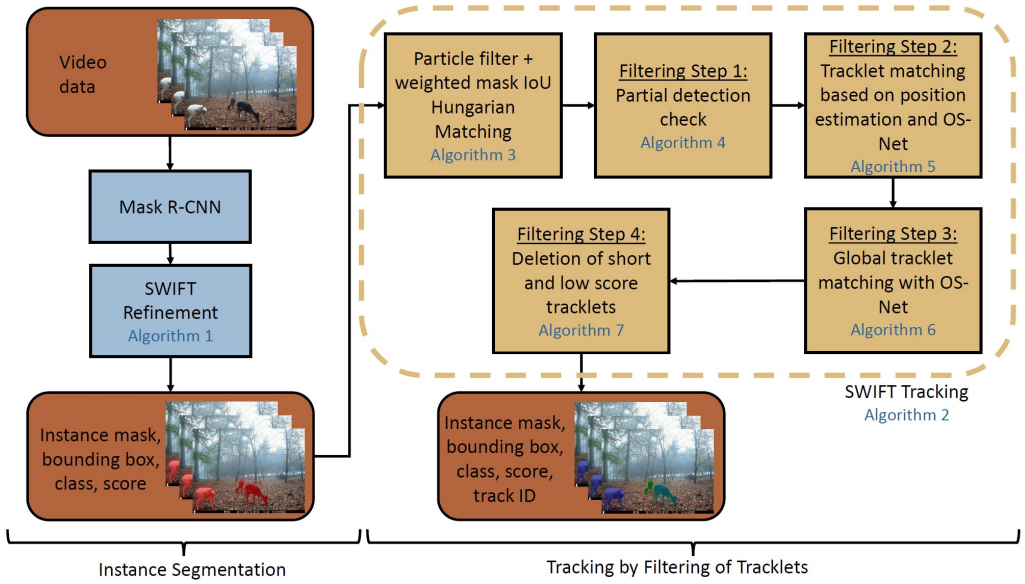



**Figure 4.4:** The workflow of SWIFT: The instance segmentation part generates instance masks, bounding boxes, class labels and score values in the input video. To achieve this, a robust and reliable instance segmentation approach, the Mask R-CNN, first detects basic instance masks of the animals. These are then improved by the newly developed SWIFT Refinement. An HR-Net is used for this. The tracking part forms tracks from the detections. Our SWIFT Tracking Approach uses several filtering stages that combine and delete tracklets. The basic tracklets are formed by a particle filter tracking with a Hungarian matching. In filtering step 1, tracklets that are spatially and temporally part of other tracklets are deleted. In filtering step 2, a motion model is used to estimate the position of animals according to occlusions and tracklets are matched by a Re-ID network, the OS-Net. In the filtering step 3, tracklets that cannot be explained by a motion model and can only be matched by a Re-ID network are matched, for example when an animal leaves the scene and then re-enters it. The final filtering step 4 deletes very short and unreliable tracklets.

## 4.3 Instance Segmentation with SWIFT

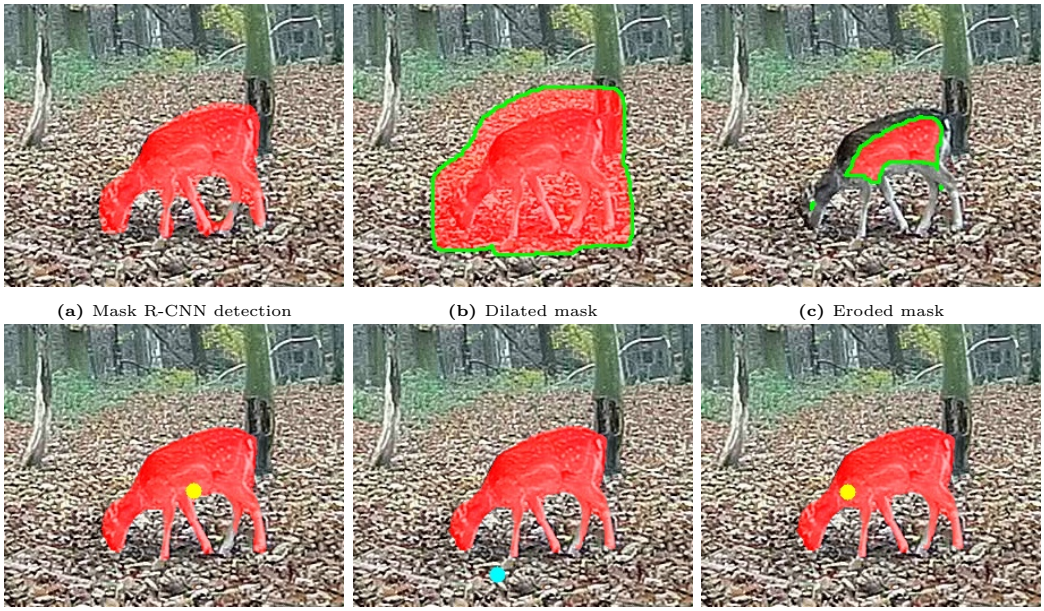
Our approach to instance segmentation is founded on the use of a Mask R-CNN (He et al., 2017) that effectively generates reliable detections and produces good instance masks. These initial masks are then further improved to high-quality masks through our refinement algorithm.

We choose Mask R-CNN as our base detection model due to its proven reliability across various application settings, often serving as a basis for advancements in instance segmentation and MOTS (cf. Section 2.1). Furthermore, we have shown the effectiveness of Mask R-CNN in wildlife monitoring in our publications (Schindler and Steinhage, 2021a) and (Schindler and Steinhage, 2021b). Mask R-CNN first extracts features in its backbone from the input image, followed by proposing regions of interest, which lay the basis for bounding box and mask regression, as well as classification. To tailor Mask R-CNN to our specific needs, we adjust the input size to match our frame resolution precisely, which is higher than the default setting and prevents the loss of input image detail and information due to image reduction. Additionally, we calculate the image mean and standard deviation for our datasets to achieve more accurate normalization of the input frames. We replace the standard ResNet-50 by a ResNeXt-101 to enhance feature extraction capabilities. Although this deeper backbone slows down training and inference, it excels at generating superior instance masks, aligning with our objectives to generate high-quality instance masks. The network is trained for 3 epochs, starting with an initial learning rate of 0.0005, a momentum of 0.9 and weight decay of 0.0005. These parameters were determined in different experiments. We decrease the learning rate by a factor of 0.1 after 1 and 2 epochs, corresponding to points where the loss plateaus, indicating the need for a reduced learning rate to continue progress, although further reduction beyond this point does not prove beneficial.

For refining instance masks, we employ a technique commonly used in interactive segmentation, where users designate positive (inside the target mask) and negative (outside the target mask) points through mouse clicks. This method allows for the precise creation of instance masks with just 3 to 10 clicks (depending on the complexity of the object).

We follow the interactive segmentation strategy of Sofiiuk et al. (2021), who demonstrated the superiority of their refinement network over other methods in this domain. Especially the use of an HR-Net (Wang et al., 2020a) backbone within the refinement network facilitates the generation of high-resolution instance masks. In our application, without the direct involvement of a user to place clicks, we design an algorithm that automatically assigns these positive and negative clicks.

The pseudo-code of the refinement process is shown in Algorithm 4.1, which we explain in detail below. An illustrative example of this process is provided in Figure 4.5.



**(a)** Mask R-CNN detection **(b)** Dilated mask **(c)** Eroded mask

**(d)** Positive center click **(e)** Negative first click **(f)** Positive first click

**Figure 4.5:** The process of setting positive and negative clicks is fundamentally based on the operations of dilation and erosion. The outline of the $mask_{eroded}$ and $mask_{dilated}$ is shown in green. Four positive and negative points are evenly sampled. In the lower row of images, the development of the mask after the first three clicks is shown, the center click and the first negative and positive click. Positive clicks are shown as yellow dots and negative as blue dots.

### 4.3.1 SWIFT Refinement Algorithm

The objective of the SWIFT Refinement Algorithm (Algorithm 4.1) is to enhance the accuracy of detections, specifically the instance masks produced by Mask R-CNN, through the automated application of positive and negative clicks in a refinement network. This algorithm is executed on all detected objects $D$ within each frame $i$. A score threshold $\tau_{score}$ is employed to ensure that only detections $d$ with a high confidence score $d.score$ undergo refinement (line 2 of Algorithm 4.1). This is important for two main reasons: firstly, instance masks with a low score often represent incorrect detections or partial detections (e.g., capturing only a portion of an animal), and secondly, each refinement process entails additional processing time. We choose a threshold of 0.5, aligning with the common benchmark for considering a detection as correct in performance metrics.

Our approach for setting positive and negative points for refinement draws on the principles of morphological operations, specifically dilation and erosion. This means that we uniformly reduce the detected instance mask from the Mask R-CNN via erosion and uniformly expand it through dilation. The extent of dilation and erosion applied is tailored to the size of the detected animal, steered by parameters $\phi_d$ for dilation and $\phi_e$ for erosion. The goal is for the dilated mask, $mask_{dilated}$, to fully encompass the target object, while the eroded mask, $mask_{eroded}$, should fit

entirely within the object's contours. Should the eroded mask $mask_{eroded}$ fragment into several smaller regions, only the largest region is considered. Through experiments we estimated various degrees of erosions and dilations that are effective based on the segmentation masks' coverage area (i.e., the pixel count within the mask). Therefore the function $f_{getParam}$ returns the desired parameters $\phi_d$ and $\phi_e$ (line 3 of Algorithm 4.1).

**Algorithm 4.1:** SWIFT Refinement Algorithm

**Input :** Frame $i$; all detections $D$ found by Mask R-CNN in $i$; threshold values $\tau_{score}$ and $\tau_{iou}$; refinement network $RN$
**Output:** Refined detections $D$

1 **for** $d \in D$ **do**
2  **if** $d.score > \tau_{score}$ **then**
3  $\phi_e, \phi_d \leftarrow f_{getParam}(area(d.mask))$ ▷ // estimate by how much mask must be enlarged or reduced depending on its size
4  $mask_{dilated} \leftarrow$ dilation$(d.mask, \phi_d)$ ▷ // enlarge the mask
5  $mask_{eroded} \leftarrow$ erosion$(d.mask, \phi_e)$ ▷ // reduce the mask
  `// Get positive and negative points`
6  $p_{pos} \leftarrow$ sample(polygon($mask_{eroded}$),4) ▷ // sample 4 points on contour of smaller mask
7  $p_{neg} \leftarrow$ sample(polygon($mask_{dilated}$),4) ▷ // sample 4 points on contour of bigger mask
8  $p_{center} \leftarrow$ computeCenterpoint$(d.mask)$ ▷ // get point in middle of mask
  `// Set positive and negative clicks`
9  $mask_{refined} \leftarrow RN(i, d.mask, p_{center},$ positive) ▷ // RN computes refined mask using the frame, mask and click information
10  **for** $j = 1$ **to** $4$ **do**
11  $mask_{refined} \leftarrow RN(i, mask_{refined}, p_{neg}[j],$ negative)
12  $mask_{refined} \leftarrow RN(i, mask_{refined}, p_{pos}[j],$ positive)
13  **end**
14  **if** $IoU(d.mask, mask_{refined}) > \tau_{iou}$ **then**
15  $d.mask \leftarrow mask_{refined}$
16  $d.box \leftarrow$ adaptBbox$(d.mask)$ ▷ // adapt the bounding box to the new mask
17  **end**
18  **end**
19 **end**
20 Return $D$

Following the creation of $mask_{dilated}$ and $mask_{eroded}$ (line 4 to 5 of Algorithm 4.1), we proceed to select points from the bounding polygon (the outline) of the masks (line 6 to 7 of Algorithm 4.1). The points derived from $mask_{dilated}$ serve as the

negative clicks, whereas those from $mask_{eroded}$ act as the positive clicks. Additionally, we calculate the mask's center point to place the initial positive click (line 8 of Algorithm 4.1). We have experienced that it is beneficial to set the first click in the center of the object to get better results. An effective number of clicks for both positive and negative selections is determined to be 4, with points evenly distributed along the polygon to ensure one point is positioned at the top, right, bottom, and left, respectively. Consequently, a total of 9 points are utilized in the refinement process. The work of (Sofiiuk et al., 2021) shows that 1 to 6 points are sufficient to get high quality masks for different datasets. The strategic placement of a point on each side of the object is crucial, especially since there is no user intervention for click placement. This method proves particularly important in scenarios where multiple animals are in close proximity, necessitating negative clicks around all sides to distinguish between them effectively. An illustration of the erosion, dilation, and initial clicks process is provided in Figure 4.5.

The refinement process begins by initializing the refinement network $RN$ with the instance mask $d.mask$ detected by Mask R-CNN, a step crucial for maintaining result stability, as demonstrated in our evaluation section 4.5.5. The initial input to $RN$ includes the current frame $i$, the instance mask $d.mask$ and the center point $p_{center}$ as the first positive click (line 9 of Algorithm 4.1). For the subsequent points we alternate between the positive and negative clicks (line 10 to 13 of Algorithm 4.1). Upon completing the refinement, we assess the Intersection over Union (IoU) between the original instance mask $d.mask$ and the refined mask $mask_{refined}$ (line 14 of Algorithm 4.1). This evaluation ensures that only significant and accurate refinements replace the original mask, preventing the acceptance of potentially incorrect refinement outcomes. If the change in the instance mask $d.mask$ is too drastic, we retain the original mask from Mask R-CNN. To facilitate this decision, we introduce a threshold $\tau_{iou}$, reflecting the training quality of the underlying Mask R-CNN and the reliability of its instance masks. We set $\tau_{iou} = 0.7$ for our scenario. Surpassing this threshold means the original mask is substituted with $mask_{refined}$, necessitating adjustments to the bounding box $d.box$ to align with the new mask (line 15 to 16 of Algorithm 4.1).

## 4.4 Tracking with SWIFT

The second part of SWIFT aims to track the identified and refined detections from the instance segmentation stage, assigning a unique track ID to each instance mask. This allows for the individual tracking of each animal within a video. To accomplish this, we have designed a tracking algorithm that utilizes various filtering steps to refine the tracking process. These steps involve either deleting existing tracklets or merging them to form larger, more comprehensive tracklets.

### 4.4.1 SWIFT Tracking Algorithm

Our SWIFT Tracking Algorithm is shown in Algorithm 4.2. In the following, we explain all individual steps of the algorithm in detail. The inputs to our algorithm include a video $V$, composed of frames $f_0, f_1, ..., f_n$, and the corresponding refined detections $D_0, D_1, ..., D_n$ across these frames. Additionally, there are various threshold values utilized in the different filtering steps that will be explained in the corresponding filtering step section. A Re-Identification (Re-ID) network $ReIDN$ is also required to re-identify individual animals following periods of occlusion or when they exit and re-enter the scene. The algorithm's outputs are the tracks $T$. As long as a track is not finalized, it is generally called a tracklet.

**Algorithm 4.2:** SWIFT Tracking Algorithm

**Input :** Video $V$ consisting of frames $f_0, f_1, ..., f_n$; all detections $D_0, D_1, ..., D_n$ from refinement algorithm in frames $f_0, f_1, ..., f_n$; threshold values $\tau_{score}$,$\tau_{cost}$,$\tau_{contain}$, $\tau_{iou}$, $\tau_{reid}$,$\tau_{reidg}$, $\tau_{length}$ and $\tau_{tscore}$; Re-ID network $ReIDN$

**Output:** Tracks $T$

`// Tracking with particle filter and IoU segmentation mask matching (Algorithm 4.3)`

1 $T \leftarrow$ ParticleMaskTracking($V$,$D$,$\tau_{score}$,$\tau_{cost}$)

`// Filtering Step 1: Partial detection check (Algorithm 4.4)`

2 $T \leftarrow$ FilteringStep1($T$,$\tau_{contain}$)

`// Filtering Step 2: Tracklet matching based on position estimation and ReID-Net (Algorithm 4.5)`

3 $T \leftarrow$ FilteringStep2($T$,$ReIDN$,$\tau_{iou}$,$\tau_{reid}$)

`// Filtering Step 3: Global tracklet matching based on ReID-Net (Algorithm 4.6)`

4 $T \leftarrow$ FilteringStep3($T$,$ReIDN$,$\tau_{reidg}$)

`// Filtering Step 4: Deletion of short and low score tracklets (Algorithm 4.7)`

5 $T \leftarrow$ FilteringStep4($T$,$\tau_{length}$, $\tau_{tscore}$)

6 Return $T$

### 4.4.2 Particle Filter and IoU Segmentation Mask Matching

Explanation of Algorithm 4.3: Our SWIFT Tracking Algorithm begins by forming basic tracklets through a combination of particle filter tracking (Gordon et al., 1993) and Intersection over Union (IoU) matching with instance masks (line 1 of Algorithm 4.2), shown in detail in Algorithm 4.3. This initial step is inspired by the simple, but very successful SORT tracking algorithm (Bewley et al., 2016). In contrast, we employ the Intersection over Union (IoU) values of instance masks instead

of the bounding boxes for detections. This adjustment is particularly beneficial in scenarios involving crowded scenes, such as deer moving within a group. Furthermore, we replace the Kalman filter (Welch et al., 2006) with a particle filter (Gordon et al., 1993) as our motion model. While both filters are designed to predict an animal's next frame position based on previous detections, the Kalman filter is limited to linear motion predictions. In contrast, a particle filter is capable of accommodating both linear and non-linear movements, offering a more versatile solution for tracking the varied motions of animals. The efficacy and comparison of these filters are further explored in our experiments Section 4.5.7.

We begin with an empty set of tracklets $\emptyset$ (line 1 of Algorithm 4.3) and sequentially process the video $V$, frame by frame (line 2 of Algorithm 4.3). Initially, detections $d$ within a frame that have a score $d.score$ below the threshold $\tau_{score}$ are filtered out (line 3 to 7 of Algorithm 4.3). The threshold $\tau_{score}$ is set at the commonly accepted value of 0.5 for instance segmentation and tracking tasks. Subsequently, we assess which tracklets to deactivate based on the absence of matches with new detections over the last 4 frames (line 8 to 10 of Algorithm 4.3). Through experiments we determined 4 as the optimal value for this threshold. This parameter ensures that a tracklet can be continued even when in one frame a detection is missed. This threshold is not intended to account for extended periods of occlusion or to address re-identification (Re-ID) challenges.

The particle filter $pf$ predicts the new position of the bounding box $t.predBbox$ for each tracklet $t$ (line 12 of Algorithm 4.3). Going beyond the SORT algorithm's bounding box prediction, we also predict the position of the instance mask $t.predMask$ within this predicted bounding box (line 13 of Algorithm 4.3) and introduce a weighting function to include prediction scores. To predict the instance mask, we resize the current bounding box (which contains the instance mask) to match the dimensions of the predicted bounding box, adjusting the instance mask's size accordingly (This stretches and compresses the instance mask accordingly). This simple and efficient approach integrates instance mask predictions into the particle filter's prediction. Given that our videos typically have a high frame rate, the changes in instance masks from one frame to the next is generally minimal. Additionally, our research (Schindler and Steinhage, 2021a) has demonstrated that more complex methods of predicting instance mask movement, such as using optical flow techniques (e.g., FFGA (Zhu et al., 2017)), are not consistently reliable for articulated objects with complex movements like animals.

To match existing tracklets with new detections, a cost matrix is constructed. This matrix is generated by calculating the Intersection over Union (IoU) for all possible pairs of the predicted instance masks $t.predMask$ and the new detections $d.mask$ within the current frame (line 17 of Algorithm 4.3). Tracklets that have been deactivated are excluded from this cost matrix. We weight the IoU values using a custom weighting function (Equation 4.1), designed to favor matches between new detections with high score values $d.score$ and existing tracklets whose most recent

**Algorithm 4.3:** Particle mask tracking

**Input :** Video $V$ consisting of frames $f_0, f_1, ..., f_n$; all detections $D_0, D_1, ..., D_n$ found by Mask R-CNN in frames $f_0, f_1, ..., f_n$; threshold values $\tau_{score}$ and $\tau_{cost}$

**Output:** Tracklets $T$

$T \leftarrow \emptyset$ ▷ // initialize tracklets with empty set
**for** $f_i \in V$ **do**
  **for** $d \in D_i$ **do**
    **if** $d.score < \tau_{score}$ **then**
      del $d$ ▷ // delete unreliable detections
    **end**
  **end**
  **for** $t \in T$ **do**
    deactivate tracklet $t$ if not matched for 4 frames
  **end**
  **for** $t \in T$ **do**
    $t.predBbox \leftarrow t$.pf.predict() ▷ // particle filter predicts new bounding box position
    $t.predMask \leftarrow$ predictMask($t.predBbox$)
  **end**
  **for** $t \in T$ **do**
    **for** $d \in D_i$ **do**
      `// compute the costmatrix using the scores of the tracklets and the detections and the overlap of the predicted mask with the detected mask`
      $costmatrix[t, d] \leftarrow$ $(-1) \cdot w(t.lastScore, d.score) \cdot$IoU($t.predMask, d.mask$)
    **end**
  **end**
  $m, unmatchedT, umatchedD \leftarrow$ HungarianAlgorithmMatching($costmatrix$)
  filter out matches $m$ with cost value $< \tau_{cost}$
  $T$.update($m, unmatchedT, unmatchedD$) ▷ // update existing tracklets and initiate new tracklets
**end**
Return $T$

detection also has a high score value $d.lastScore$. By squaring the mean of these two scores, the function effectively penalizes matches between low-scoring detections. To align with the Hungarian Algorithm's (Kuhn, 1955) preference for the lowest values (used for the matching process), the cost matrix values are multiplied by $-1$. So the Hungarian Algorithm matches the lowest not the highest values. Matching means

that a one-to-one correspondence is built so that exactly one detection is associated with one existing tracklet.

$$w(t.lastScore, d.score) = \left(\frac{t.lastScore + d.score}{2}\right)^2 \tag{4.1}$$

The Hungarian Algorithm outputs a list of matches $m$, along with lists of unmatched tracklets $unmatchedT$ and unmatched detections $unmatchedD$ (line 20 of Algorithm 4.3). To ensure the reliability of these matches, we apply a threshold $\tau_{cost}$ to filter out unreliable matches, as the Hungarian Algorithm does not incorporate a threshold by default (line 21 of Algorithm 4.3). We set $\tau_{cost}$ to 0.3, opting for a strict threshold to minimize the occurrence of erroneous matches. This is crucial since our subsequent filtering steps do not split existing tracklets but instead either merge them or delete them entirely. Matches with a cost value below $\tau_{cost}$ are removed from the matches list, reallocating the involved tracklets and detections to $unmatchedT$ and $unmatchedD$, respectively. The algorithm then updates all tracklets with this new information (line 22 of Algorithm 4.3). Matches are added to existing tracklets. Unmatched tracklets count internally how many times they fail to match and unmatched detections initiate a entirely new tracklets. The result of the particle mask tracking algorithm are tracklets that build a basis for the following filtering steps.

### 4.4.3 Filtering Step 1: Partial Deletion Check

Explanation of Algorithm 4.4: In the initial step of our filtering process (line 2 of Algorithm 4.2), we assess whether any tracklets are either fully spatially enclosed by another tracklet or entirely encompass one. This scrutiny is necessary because detection processes often capture not just the targeted animal but also parts of it, such as a leg or head, as separate detections. Despite these partial detections typically scoring lower, their score might still exceed the $\tau_{score}$ threshold (from Algorithm 4.3), leading them to be mistakenly acknowledged as valid detections. Similarly, a detection might erroneously include additional elements like a tree or another animal in close proximity, due to color similarities, resulting in larger detections that might fully enclose or be enclosed by another tracklet. The tracking with the particle filter has accordingly built short tracklets from these detections, which either lie completely within another tracklet or completely enclose another tracklet. These tracklets are faulty and therefore should be deleted.

To address this, the computeContainment function (line 3 of Algorithm 4.4) evaluates whether one tracklet $t_1$ is temporally contained within another $t_2$. If so, it calculates a containment value $c(t_1, t_2)$ by measuring, for each frame $f$ within $t_1$, the intersection of instance masks between $t_1$ and $t_2$ relative to the area of the instance

mask of $t_1$ are summed up. This sum is then averaged over the length $len(t_1)$ of tracklet $t_1$:

$$c(t_1, t_2) = \frac{\sum_{f \in t_1} \left( \frac{t_1[f].mask \cap t_2[f].mask}{t_1[f].mask} \right)}{len(t_1)} \tag{4.2}$$

The final value should identify tracklets that are either incorrectly segmented parts of an animal or erroneous detections that mistakenly include other objects. These incorrect tracklets are marked for deletion to refine the tracking outcomes.

Additionally, we compute $c(t_2, t_1)$ to account for the ambiguity regarding which tracklet is contained within the other. The computeContainment function then determines the greater value between $c(t_1, t_2)$ and $c(t_2, t_1)$ and returns it. A high $c(t_1, t_2)$ indicates that $t_1$ is a partial detection of $t_2$, such as when $t_1$ captures only the animal's head and $t_2$ represents the full animal correctly detected. Conversely, a high $c(t_2, t_1)$ suggests that $t_1$ erroneously includes $t_2$, for example, $t_1$ might be a mistaken detection that combines the animal with a nearby tree, whereas $t_2$ accurately detects the animal alone. If the containment value $c$ exceeds the threshold $\tau_{contain}$, the tracklet $t_1$ is deleted (line 4 to 5 of Algorithm 4.4). Based on our experimental findings, setting $\tau_{contain} = 0.85$ proved to be the most effective for identifying and eliminating improperly contained tracklets.

**Algorithm 4.4:** SWIFT Filtering Step 1

```
   Input  : Tracklets T; threshold value τ_contain
   Output: Tracklets T
1  for t_1 ∈ T do
2  |  for t_2 ∈ T do
3  |  |  c ← computeContainment(t_1,t_2)   ▷ // check how much one tracklet
   |  |    overlaps with the other
4  |  |  if c > τ_contain then
5  |  |  |  del t_1          ▷ // if there is a significant overlap, one tracklet is a
   |  |  |    partial detection of another tracklet and should be deleted
6  |  |  end
7  |  end
8  end
9  Return T
```

### 4.4.4 Filtering Step 2: Tracklet Matching based on Position Estimation and ReID-Net

Explanation of Algorithm 4.5: The Filtering step 2 (line 3 of Algorithm 4.2) merges tracklets based on a position estimation and a Re-ID network that is used to re-

identify animals in the same video. We sort the tracklets by their length in descending order before proceeding through each one. Initially, we extract all tracklets $T_{nO}$ that do not have a temporal overlap with the selected tracklet $t_1$ (line 12 of Algorithm 4.2). The velocity for the last bounding box of $t_1$ is then determined by averaging the velocities of its last 10 bounding boxes (line 4 of Algorithm 4.5). This simplistic motion model is preferred over complex models like the Kalman filter, which may alter the bounding box size significantly, potentially leading to its disappearance over time, especially since no new measurements are available to update either a Kalman or particle filter. We keep the bounding box fixed and only estimate the new position based on this fixed velocity.

While the last frame $n$ of the video is not reached and the tracklet $t_1$ was not already successfully matched (line 5 of Algorithm 4.5) we estimate the new bounding box $t_1.bboxEst$ position of $t_1$ using the constant velocity $v_1$ (line 6 of Algorithm 4.5). We refrain from predicting the mask, as this would be unreliable over an extended period. A potential match with any tracklet from $T_{nO}$ is considered possible if a tracklet $t_{nO}$ starts in the current frame $curFrame$ and its initial bounding box overlaps with $t_1$'s estimated bounding box, surpassing the $\tau_{iou}$ threshold (line 8 of Algorithm 4.5). Upon meeting this criterion, we apply the Re-ID network $ReIDN$ to the tracklets for matching. The network calculates the Re-ID features of the bounding box that is in the middle of each tracklet. This provides more reliable re-identification than using the last bounding box of the first tracklet and the first bounding box of the second tracklet. Often, the first and last frames of a tracklet display only partial views of an animal (such as a single leg of an animal) due to occlusion or poor visibility. The Euclidean distance between the Re-ID features of $t_1$ and $t_{nO}$ is computed to determine if they represent the same animal. If $ReIDN(t_1, t_{nO})$ is below the $\tau_{reid}$ threshold, indicating a high likelihood of a match, the tracklets are merged (line 9 of Algorithm 4.5). This threshold $\tau_{reid}$ is crucial to prevent merging tracklets that track different animals.

### 4.4.5 Filtering Step 3: Global Tracklet Matching based on ReID-Net

Explanation of Algorithm 4.6: In the third filtering step (line 4 of Algorithm 4.2), we utilize the Re-ID network $ReIDN$ to match tracklets based solely on Re-ID features, bypassing the spatial constraints and position estimations from the previous step. This approach addresses scenarios not covered by spatial analysis, such as when an animal exits and re-enters the scene or changes direction while obscured, for instance, by a tree. We call this phase global tracklet matching due to its lack of spatial limitations on re-identification. Initially, Re-ID features are computed for all tracklets, with $ReIDN$ extracting features from the bounding box in the middle of each tracklet (line 1 to 3 of Algorithm 4.6). A distance matrix is then generated using these features, employing the Euclidean distance (line 4 of Algorithm 4.6). Since the matrix is symmetrical, we only need to consider the upper triangular

**Algorithm 4.5:** SWIFT Filtering Step 2

**Input :** Tracklets $T$; threshold values $\tau_{iou}$ and $\tau_{reid}$; Re-ID network $ReIDN$
**Output:** Tracklets $T$

```
1  sort T by tracklet length (longest first)
2  for t_1 ∈ T do
3      T_nO ← get all t with no temporal overlap to t_1
4      estimate velocity v_1 of t_1.bbox
5      while curFrame< n and tracklets not merged do
6          t_1.bboxEst ← estimate new bbox position of t_1 with velocity v_1
7          for t_nO in T_nO do
               // a match is possible when (1) the non overlapping
                  tracklet starts in curFrame and (2) the instance
                  mask of this tracklet significantly overlaps with
                  the estimated bbox of t
8              if (curFrame = t_nO[0].frame) and
                IoU(t.bboxEst, t_nO[0].bbox) > τ_iou then
                   // Re-ID network checks, if it is (probably) the
                      same animal
9                  if ReIDN(t_1, t_nO) < τ_reid then
10                     merge t_1 and t_nO
11                 end
12             end
13         end
14         curFrame ← curFrame + 1
15     end
16 end
17 Return T
```

matrix. Distances for temporally overlapping tracklets are set to $\infty$, preventing a merging of these tracklets. The values on the diagonal are also set to $\infty$, since merging a tracklet with itself is impossible (line 5 of Algorithm 4.6). Tracklets with the smallest distance value in the matrix are merged, provided this value falls below the threshold $\tau_{reidg}$ (line 6 to 9 of Algorithm 4.6). Following each merge operation, the distance matrix is updated to reflect the newly combined tracklet, with its entries taking the minimum value from the corresponding entries of the two original tracklets. This process continues as long as the smallest distance remains below the established threshold, ensuring that only tracklets likely representing the same animal are merged.

**Algorithm 4.6:** SWIFT Filtering Step 3

**Input :** Tracklets $T$; threshold value $\tau_{reidg}$; Re-ID network $ReIDN$
**Output:** Tracklets $T$

**for** $t \in T$ **do**
  $featuresReID \leftarrow ReIDN(t)$ ▷ // extract Re-ID features from all tracklets with the Re-ID network
**end**
$distanceMatrix \leftarrow$ computeDistMatrix($featuresReID$,Euclidean) ▷ // compute the distance between all Re-ID features to each other
set entries of temporally overlapping tracklets in $distanceMatrix$ to $\infty$
**while** min($distanceMatrix$) $< \tau_{reidg}$ **do**
  merge tracklets from $T$ where min($distanceMatrix$) ▷ // if Re-ID features have a small distance to each other they are probably the same animal
  update $distanceMatrix$
**end**
Return $T$

### 4.4.6 Filtering Step 4: Deletion of Short and Low Score Tracklets

Explanation of Algorithm 4.7: In the fourth and final step of our filtering process (line 5 of Algorithm 4.2) we eliminate tracklets that are either too short in duration or have a low average score. Each tracklet is checked if its length falls below the threshold $\tau_{length}$, set at 5 for our scenario. This threshold varies based on the frame rate of the videos being analyzed, with lower frame rates necessitating a reduced $\tau_{length}$. The score of a tracklet, $score(t)$, is calculated as the average of the detection scores for all instance masks within the tracklet. Tracklets characterized by a low average score are deemed to consist predominantly of unreliable detections and thus should be deleted. Through experimental validation, we have established a score threshold $\tau_{tscore}$ of 0.75 to decide which tracklets should be deleted, ensuring that only tracklets with sufficiently reliable detections are retained for further analysis.

**Algorithm 4.7:** SWIFT Filtering Step 4

**Input :** Tracklets $T$; threshold values $\tau_{length}$ and $\tau_{tscore}$
**Output:** Tracklets $T$

**for** $t \in T$ **do**
  **if** $len(t) < \tau_{length}$ *or* $score(t) < \tau_{tscore}$ **then**
    del $t$ ▷ // delete short and unreliable tracklets
  **end**
**end**
Return $T$

Finally, the SWIFT Tracking Algorithm returns the found tracks $T$ (line 6 of Algorithm 4.2).

## 4.5 Results

In this section we evaluate our SWIFT approach. First, we introduce the metrics for instance segmentation, tracking and the combined task. Especially the COCO (Common Objects in COntext) metrcis for instance segmentation are complex. Therefore, we explain them in detail. In the following, we analyze the instance segmentation quality of SWIFT. We then analyze the tracking capability of SWIFT. Finally, we look at the joint task of tracking and instance segmentation.

### 4.5.1 COCO Metrics

To evaluate SWIFT's performance, we employ the standard COCO (Common Objects in COntext) metrics designed for instance segmentation evaluation. In the following, we base our explanation on (Schindler and Steinhage, 2021a,b; Padilla et al., 2021; Everingham et al., 2010).

We calculate the average precision $AP^{mask}$ for instance masks as an average across IoU threshold values from 0.5 to 0.95 in steps of 0.05. Specific average precision metrics, $AP^{mask}_{0.5}$ and $AP^{mask}_{0.75}$, are determined at IoU thresholds of 0.5 and 0.75, respectively, with $AP^{mask}_{0.75}$ serving as a stricter criterion compared to $AP^{mask}_{0.5}$. The metrics $AP^{mask}_{small}$, $AP^{mask}_{medium}$, and $AP^{mask}_{large}$ assess the average precision for objects of varying sizes. Small objects are all objects with a size smaller than $32 \times 32$ pixels. Medium sized objects lie in the range between $32 \times 32$ pixels and $96 \times 96$ pixels. Objects that are bigger than $96 \times 96$ pixels are considered as large objects. In our application scenario most of the objects are considered as large objects. Small objects are almost non existent, only representing very small animal parts, while they enter or leave the scene. Additionally, the average recall $AR^{mask}$ mirrors the average precision by averaging over IoU thresholds from 0.5 to 0.95, also in steps of 0.05, but for recall values.

Each prediction from Mask R-CNN, whether a bounding box or segmentation mask, is accompanied by a confidence score that reflects the reliability of the prediction. The Intersection over Union (IoU), a crucial component in metric computation, is defined as

$$IoU(B_p, B_{gt}) = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{4.3}$$

where $B_p$ represents the area of the predicted bounding box or segmentation mask, and $B_{gt}$ refers to the area of the ground truth bounding box or segmentation mask,

with the definition being analogous for both bounding boxes and segmentation masks.

A detection is considered a true positive (TP) if

- the confidence score exceeds a specific threshold, such as 0.5, indicating the detection's reliability.
- the Intersection over Union (IoU) surpasses a certain threshold, for instance, 0.5, ensuring the detection is accurately positioned.
- the predicted class matches the actual class of the object.

A detection is classified as a false positive (FP) if its confidence score exceeds the threshold, but either the class prediction is incorrect, the IoU is below the desired threshold, or both conditions apply. A false negative (FN) occurs, when the confidence score of a detection is lower than the threshold for a ground truth object that should be detected. Precision $p$ and Recall $r$ are calculated using their standard definitions:

$$p = \frac{TP}{TP + FP} \qquad \text{and} \qquad r = \frac{TP}{TP + FN} \tag{4.4}$$

Varying the confidence score threshold leads to distinct pairs of precision and recall values. When recall is plotted on the x-axis and precision on the y-axis, this produces a precision-recall curve. Gradually lowering the confidence threshold from 1 to 0 results in a monotonically decreasing recall. While precision may fluctuate, it generally decreases as the confidence threshold is reduced. In this precision-recall curve, a precision at the x-coordinate $r$ can therefore be regarded as $p(r)$.

The average precision (AP) serves as a numerical metric for comparing precision-recall curves across various detectors, calculating the mean precision across different recall levels for a single detector. The precision-recall curve is constructed through the interpolation of multiple (recall, precision) points. Specifically, the interpolated precision $p_{interp}$ at a given recall level $r$ is defined as

$$p_{interp}(r) = \max_{r' \geq r} p(r') \tag{4.5}$$

This implies that $p_{interp}$ represents the highest precision value obtained for a recall level $r'$ that is greater than or equal to $r$. The recall levels $r$ are chosen at equal intervals. For the COCO metrics neighbouring recall levels differ by 0.01. Consequently, 101 recall levels $r$ ranging from 0.0 to 1.0 with a step size of 0.01 are evaluated in the calculation. Thus, for COCO datasets, the Average Precision (AP) for a class $i$ at an IoU threshold $th$ is calculated as

$$AP_{i,th} = \frac{1}{101} \sum_{r \in \{0.0 : 0.01 : 1.0\}} p_{interp}(r) \tag{4.6}$$

The AP is averaged over all classes, commonly referred to as mean Average Precision (mAP). In COCO evaluation metrics, this mean average precision is also denoted by AP, too. We follow the naming convention from COCO datasets. For $K$ classes, each with its respective $AP_{i,th}$, the mean $AP_{th}$ at an IoU threshold $th$ is computed as

$$AP_{th} = \frac{\sum_{i=1}^{K} AP_{i,th}}{K} \tag{4.7}$$

The AP for the COCO evaluation metric, excluding $AP_{0.50}$ and $AP_{0.75}$, is determined by averaging the AP across various IoU thresholds, specifically from 0.50 to 0.95 in increments of 0.05. Thus, the AP represents an average across both all classes and the specified range of IoU thresholds, calculated as

$$AP = AP_{0.50:0.05:0.95} = \frac{AP_{0.50} + AP_{0.55} + ... + AP_{0.95}}{10} \tag{4.8}$$

In addition to the overall calculation, we specifically examine $AP_{0.50}$ and $AP_{0.75}$ as distinct components of our evaluation metrics. $AP_{0.50}$, corresponding to an IoU threshold of 0.50, is also recognized as the Pascal VOC metric. Meanwhile, $AP_{0.75}$, associated with an IoU threshold of 0.75, is referred to as the strict metric within the context of COCO evaluations.

The average recall (AR) is calculated across various IoU thresholds, with the COCO datasets specifically considering the 10 IoU thresholds from 0.50 to 0.95, in increments of 0.05, mirroring the approach used for AP calculation (4.8). Consequently, the AR for class $i$ at an IoU threshold $th$ is calculated as

$$AR_{i,th} = \frac{r_{0.50} + r_{0.55} + ... + r_{0.95}}{10} \tag{4.9}$$

The AR is averaged across all $K$ classes, which can also be referred to as mean average recall (mAR). However, within the COCO metrics, this aggregate measure mAR is simply denoted as AR. Thus, the $AR_{th}$ at a specific IoU threshold $th$ is computed as

$$AR_{th} = \frac{\sum_{i=1}^{K} AR_{i,th}}{K} \tag{4.10}$$

We evaluate two variations of average recall: $AR_{max=1}$ and $AR_{max=10}$, which are calculated based on allowing a maximum of 1 detection and a maximum of 10 detections

per image, respectively. Both versions of AR are averaged across the aforementioned 10 IoU thresholds:

$$AR = \frac{AR_{0.50} + AR_{0.55} + ... + AR_{0.95}}{10} \tag{4.11}$$

### 4.5.2 MOT Metrics

The accuracy of our tracking approach is assessed using the Multiple Object Tracking (MOT) metrics, with further details available in the works of Schindler and Steinhage (2021b); Bernardin and Stiefelhagen (2008); Milan et al. (2016). We consider the accuracy $MOTA$ and precision $MOTP$ of our tracking outcomes. For the calculation of the MOT metrics, the false positives $FP$, false negatives $FN$ and the id switches $IDSW$ have to be considered. Ground truth tracks are categorized into Mostly Tracked (MT), Partially Tracked (PT), and Mostly Lost (ML) based on the tracking algorithm's performance in identifying each track.

Similar to the COCO metrics, the Multiple Object Tracking (MOT) metrics comprise various measures. MOTA, which stands for Multi-Object Tracking Accuracy, is defined as

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t} = \frac{\sum_t (TP_t - FP_t - IDSW_t)}{\sum_t GT_t} \tag{4.12}$$

where $t$ represents the frame index, $FN$ the number of false negatives, $FP$ the number of false positives, $IDSW$ the number of identity switches, and $GT$ the total number of ground truth objects. The classification into true positives ($TP$), false negatives ($FN$), and false positives ($FP$) follows the same process as outlined for bounding boxes in Section 4.5.1, using the IoU value for determination. An identity switch is identified when a ground truth target $i$ is matched to track $j$, whereas its previous match was $k \neq j$. Notably, MOTA can result in a negative value if the errors produced by the tracking algorithm surpass the number of actual tracks, placing the MOTA value within the range of $(-\infty, 100)$.

The Multi-Object Tracking Precision (MOTP) quantifies the average precision between the bounding boxes of true positive detections and their corresponding ground truth bounding boxes.

$$MOTP = \frac{\sum_{t,i} d_{t,i}^{bbox}}{\sum_t c_t} \tag{4.13}$$

In this context, $d_{t,i}^{bbox}$ represents the IoU value of bounding box $i$ with its assigned ground truth, while $c_t$ denotes the number of matches in frame $t$. Thus, MOTP

calculates the average overlap between all correctly matched detections and their ground truths. So MOTP is a measure for the localisation accuracy. The range of MOTP values is between 50 and 100, reflecting that an IoU of 50 is the minimum threshold for a detection to qualify as a true positive.

Ground truth tracks are categorized into three distinct types: Mostly Tracked (MT), Partially Tracked (PT), and Mostly Lost (ML), based on the effectiveness of the tracking algorithm in identifying the track throughout its duration. A track is classified as MT if it is successfully identified for at least 80% of its lifespan. Conversely, a track is deemed ML if less than 20% of its lifespan is matched correctly. The PT category encompasses all other scenarios not meeting the criteria for MT or ML.

The identification metric $IDF1$ computes a bijective mapping between the ground truth tracks and the generated tracks. The difference to the MOTA metric is that MOTA considers matches on the level of single detections (in a frame) and $IDF1$ considers matches on a track level.

The $IDF1$ metric is defined as

$$IDF1 = \frac{IDTP}{IDTP + 0.5 \cdot IDFN + 0.5 \cdot IDFP} \tag{4.14}$$

The $IDTP$ is defined as the (identity) true positives, which are the matches from the overlapping parts of matched tracks. The overlap is defined in the same way as for MOTA that means the IoU threshold for the bounding boxes is set to 0.5. The (identity) false negatives $IDFN$ are the ground truth detections from the parts of the matched tracks that do not overlap and from the remaining unmatched tracks. In the same way the (identity) false positives $IDFP$ are the remaining detected tracks in these both scenarios.

### 4.5.3 MOTS Metrics

We use the MOTS metrics introduced by Voigtlaender et al. (2019) for the combined task of instance segmentation and tracking. These metrics extend the traditional MOT metrics by utilizing instance masks in place of bounding boxes for their calculations. The mask-based metrics, $MOTSA$ and $MOTSP$, assess the system's accuracy and precision, respectively. So $MOTSA$ and $MOTSP$ are defined in the same way as the corresponding MOT metrics in Section 4.5.2 with the only difference that for the IoU matching masks are used instead of bounding boxes.

$$MOTSA = 1 - \frac{\sum_t (FN_t^{mask} + FP_t^{mask} + IDSW_t)}{\sum_t GT_t} = \frac{\sum_t (TP_t^{mask} - FP_t^{mask} - IDSW_t)}{\sum_t GT_t} \tag{4.15}$$

$$MOTSP = \frac{\sum_{t,i} d_{t,i}^{mask}}{\sum_t c_t^{mask}} \tag{4.16}$$

Additionally, the $sMOTSA$ (soft Multi-Object Tracking and Segmentation Accuracy) metric is a soft variant of $MOTSA$. For the calculation of $sMOTSA$ the $TP_t^{mask}$ are replaced by $\sum_{t,i} d_{t,i}^{mask}$. As a result of this replacement, not the number of correct matches is counted, but the accuracy of the instance masks is included in the calculation. In this way $sMOTSA$ should measure segmentation and detection and tracking quality at once.

$$sMOTSA = \frac{\sum_t (\sum_{t,i} d_{t,i}^{mask} - FP_t^{mask} - IDSW_t)}{\sum_t GT_t} \tag{4.17}$$

### 4.5.4 Implementation Details

All software is developed using Python 3 and utilizes PyTorch (Paszke et al., 2019) for the construction and training of networks. The Mask R-CNN model we employ is based on PyTorch's Mask R-CNN detection framework. Our refinement network's implementation draws from the official GitHub repository provided by Sofiiuk et al. (2021). Within this setup, the HR-Net is pre-trained on both the COCO dataset (Lin et al., 2014) and the LVIS dataset (Gupta et al., 2019), ensuring a robust foundation for our applications. The training and testing of the networks was performed with GPUs with 24 GB graphic memory, either a GeForce RTX 3090, a QUADRO RTX 6000 or a NVIDIA RTX A5000.

### 4.5.5 Evaluation Studies on SWIFT

We conduct several evaluation studies for SWIFT to investigate the efficiency and functionality of each component. First, we consider instance segmentation and then tracking. Within the tracking evaluation, we will also examine underlying motion model for forming basis tracklets in our tracking algorithm (Algorithm 4.2). In each section, we compare our approaches to known competitive approaches to the respective task.

All datasets are split into training and testing sets, ensuring that approximately 20% of each class is allocated to the test set. Given the limited number of videos for each dataset, we do not create a separate validation set. Therefore, we execute a stratified 5-fold cross-validation for each dataset. The term "stratified" means that in each fold 20% of the data from each class is designated as test data. This ensures that the test sets accurately represent the animal class distribution of the entire dataset.

### 4.5.6 Instance Segmentation Evaluation

We assess the quality of instance segmentation of SWIFT by employing COCO metrics. Specifically, we compare the performance of our refinement algorithm (Algorithm 4.1) with the outcomes from the baseline Mask R-CNN (He et al., 2017), which operates without any refinement enhancements. Moreover, we analyze our choice of refinement network. Therefore, we compare the HR-Net (Sofiiuk et al., 2021), with the f-BRS (feature-Backpropagating Refinement Scheme) approach (Sofiiuk et al., 2020), which also uses positive and negative points for interactive segmentation. For the HR-Net, we specifically analyze two distinct operational modes. Additionally, we use the new instance segmentation foundation model Segment Anything Model SAM (Kirillov et al., 2023) as the refinement network in our algorithm. Here, too, we examine different ways of using the model. As mentioned in Section 3, the approach of Sofiiuk et al. (2021) was successfully used for the interactive annotation of the dataset.

We employ Stochastic Gradient Descent (SGD) as the optimization technique for training our Mask R-CNN model. Training starts with an initial learning rate of 0.0005, momentum of 0.9, and a weight decay of 0.0005. Altering the momentum and weight decay parameters did not yield improvements based on our evaluations. To enhance mask quality, we adjust the resizing parameters of the network to match the frame resolution of our dataset. This adjustment slows down the training process and increases GPU memory requirements compared to standard settings, but it results in higher quality masks.

Additionally, we compute the mean and standard deviation of each dataset's images to tailor normalization specifically for our needs. The network is trained over 3 epochs with learning rate reductions by a factor of $\gamma = 0.1$ after the first and second epoch to further reduce loss. Extending the training duration or further reducing the learning rate beyond this point does not enhance results.

The refinement networks HR-Net and the f-BRS are pre-trained on the COCO and LVIS datasets. The foundation model SAM has been trained on the huge SA-1B dataset. The HR-Net includes a feature called *setmask* that allows for the initialization of the refinement process using a given instance mask. We evaluate the performance of HR-Net both with and without the use of this initialized mask. For the SAM model, we investigate two different ways of predicting masks. The first mode also uses positive and negative clicks in the same way as the other two refinement networks. The second mode uses a bounding box as input, in which the best suitable mask is then searched for.

The results of the instance segmentation are presented in Table 4.1. The best results for each metric in each dataset is marked in blue. We consider 5 of the COCO metrics presented in Section 4.5.1. We use the $AP^{bbox}$ to analyze the accuracy of the bounding boxes. Our focus is on the evaluation of the instance segmentation

| Dataset | Method | $AP^{bbox}$ | $AP^{mask}$ | $AP^{mask}_{0.50}$ | $AP^{mask}_{0.75}$ | $AR^{mask}$ |
|---|---|---|---|---|---|---|
| Rolandseck Daylight | Mask R-CNN | 0.547 | 0.435 | 0.803 | 0.431 | 0.503 |
| | SWIFT f-BRS | 0.501 | 0.426 | 0.803 | 0.385 | 0.509 |
| | SWIFT HR-Net w/ sm | 0.526 | 0.452 | 0.804 | 0.429 | 0.541 |
| | SWIFT HR-Net w sm | 0.562 | 0.485 | **0.809** | 0.517 | 0.559 |
| | SWIFT SAM | 0.548 | 0.441 | 0.802 | 0.431 | 0.519 |
| | SWIFT SAM bbox | **0.583** | **0.497** | 0.806 | **0.545** | **0.579** |
| Rolandseck Nighttime | Mask R-CNN | 0.573 | 0.489 | 0.836 | 0.532 | 0.563 |
| | SWIFT f-BRS | 0.552 | 0.470 | 0.834 | 0.463 | 0.567 |
| | SWIFT HR-Net w/ sm | 0.585 | 0.502 | 0.835 | 0.514 | 0.598 |
| | SWIFT HR-Net w sm | 0.622 | **0.537** | **0.840** | **0.579** | **0.622** |
| | SWIFT SAM | 0.591 | 0.493 | 0.836 | 0.527 | 0.577 |
| | SWIFT SAM bbox | **0.648** | 0.535 | 0.835 | 0.574 | 0.501 |
| Bavarian Forest Daylight | Mask R-CNN | 0.539 | 0.558 | 0.834 | 0.665 | 0.668 |
| | SWIFT f-BRS | 0.514 | 0.530 | 0.835 | 0.610 | 0.659 |
| | SWIFT HR-Net w/ sm | 0.567 | 0.576 | 0.835 | 0.638 | 0.711 |
| | SWIFT HR-Net w sm | **0.621** | **0.609** | **0.841** | **0.700** | **0.729** |
| | SWIFT SAM | 0.545 | 0.547 | 0.834 | 0.634 | 0.671 |
| | SWIFT SAM bbox | 0.605 | 0.574 | 0.834 | 0.653 | 0.704 |
| Bavarian Forest Nighttime | Mask R-CNN | 0.244 | 0.290 | 0.585 | 0.254 | 0.460 |
| | SWIFT f-BRS | 0.260 | 0.288 | 0.585 | 0.239 | 0.462 |
| | SWIFT HR-Net w/ sm | 0.249 | 0.286 | **0.586** | 0.217 | 0.380 |
| | SWIFT HR-Net w sm | **0.285** | **0.315** | **0.586** | **0.305** | **0.490** |
| | SWIFT SAM | 0.275 | 0.300 | **0.586** | 0.255 | 0.477 |
| | SWIFT SAM bbox | 0.25 | 0.297 | 0.583 | 0.224 | 0.484 |
| Wildlife Crossings | Mask R-CNN | 0.146 | 0.196 | **0.455** | 0.105 | 0.415 |
| | SWIFT f-BRS | 0.178 | 0.216 | **0.455** | 0.142 | 0.442 |
| | SWIFT HR-Net w/ sm | 0.201 | 0.231 | **0.455** | 0.165 | **0.468** |
| | SWIFT HR-Net w sm | 0.206 | 0.232 | 0.452 | **0.192** | 0.463 |
| | SWIFT SAM | 0.169 | 0.211 | **0.455** | 0.135 | 0.436 |
| | SWIFT SAM bbox | **0.208** | **0.236** | **0.455** | 0.181 | 0.458 |

**Table 4.1:** Instance segmentation comparison: We compare the results of the base Mask R-CNN with our SWIFT Refinement Algorithm using different refinement networks. We use the f-BRS, the HR-Net and SAM as the refinement model. For the HR-Net we analyze the two different modes with setmask (w sm) and without setmask (w/ sm), which describe whether the HR-Net is initialized with or without the Mask R-CNN mask. For the Segment Anything Model we compare the type of refinement. SWIFT SAM uses the positive and negative clicks like the other approaches. The SWIFT SAM bbox approach uses the bounding box of the mask from Mask R-CNN to determine the refined mask. The depicted metrics are the standard COCO metrics. The best results for each dataset are shown in blue.


**(a)** Rolandseck Daylight


**(b)** Rolandseck Nighttime

**Figure 4.6:** Exemplary instance segmentation results with SWIFT from the Rolandseck datasets: The bounding box and mask are colored in the respective instance class color, green for fallow deer and red for red deer. Moreover, the animal class is written above the corresponding bounding box with the detection score value.

accuracy. Accordingly, we consider $AP^{mask}$, $AP^{mask}_{0.50}$, $AP^{mask}_{0.75}$ as well as the $AR^{mask}$. Other sub-metrics of the COCO metrics, which take object size into account, do not play a major role here, as small and medium-sized objects as defined by the COCO metrics only occur very rarely in our datasets.

**(a)** Bavarian Forest Daylight

**(b)** Bavarian Forest Nighttime

**Figure 4.7:** Exemplary instance segmentation results with SWIFT from the Bavarian Forest datasets: The bounding box and mask are colored in the respective instance class color, blue for roe deer and red for red deer. Moreover, the animal class is written above the corresponding bounding box with the detection score value.

The best results are achieved by SWIFT in combination with the HR-Net and the setmask option. For the datasets Rolandseck Nighttime, Bavarian Forest Daylight and Bavarian Forest Nighttime this combination achieves the highest results for all metrics. For the Rolandseck Daylight and Wildlife Crossings datasets the SAM model with the bounding box prompt is slightly better than the HR-Net. The f-BRS refinement shows that the choice of the refinement network is very important because the f-BRS worsens the instance mask AP in comparison to the basis Mask R-CNN detections. The possibility to initialize the HR-Net with the Mask R-CNN detection with the setmask functionality is very beneficial. For all datasets this leads to improved results. Without the initialization the pure click based refinement can lead to faulty masks, especially if there are other animals or objects with the same color nearby. The Segment Anything model works in our scenario better with the bounding box prompt instead of the click based prompt. Especially for the datasets Rolandseck Daylight and Wildlife Crossings the SAM bbox approach even outperforms the HR-Net. Wildlife Crossings is the smallest of all 5 considered

datasets with the worst video quality. Therefore, the Mask R-CNN results show the lowest AP values and therefore they do not work as a reliable basis as for the other datasets. The Rolandseck Daylight dataset consists of color images (frames) and has a high resolution. Therefore, this dataset is most similar to the training data of the SAM model. As a consequence, the results of SAM can outperform the HR-Net. In general, the $AP^{mask}_{0.75}$ shows the most improvements of the refinement. This is because this metric takes into account the finer details of the instance masks. For example, for the Rolandseck Daylight dataset the HR-Net with setmask improves the $AP^{mask}_{0.50}$ only from 0.803 to 0.809, but the stricter $AP^{mask}_{0.75}$ from 0.431 to 0.517 from the Mask R-CNN basis results. In general, it is clear that refinement with both HR-Net and SAM achieves a great improvement in mask accuracy for all datasets compared to the basic Mask R-CNN detections.

We show exemplary instance segmentation results with SWIFT for the Rolandseck datasets in Figure 4.6. Moreover, we present results with SWIFT for the Bavarian Forest datasets in Figure 4.7 and for the Wildlife Crossings dataset in Figure 4.8.

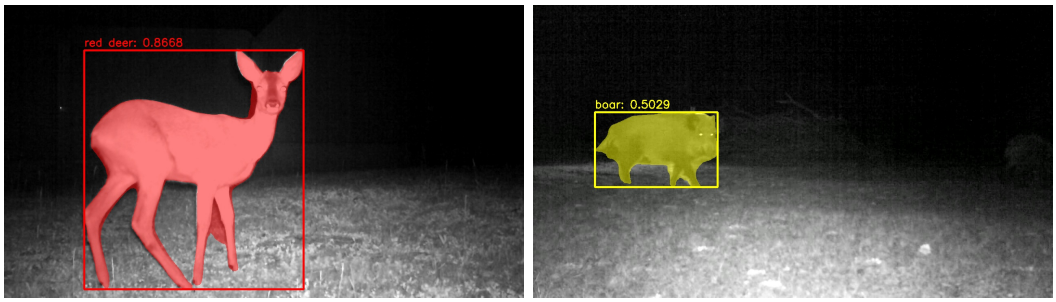



**Figure 4.8:** Exemplary instance segmentation results with SWIFT from the Wildlife Crossings dataset: The bounding box and mask are colored in the respective instance class color, red for red deer and yellow for boar. Moreover, the animal class is written above the corresponding bounding box with the detection score value.

### 4.5.7 Tracking Evaluation

To assess the tracking performance of SWIFT, we employ the MOT metrics that we described in Section 4.5.2 and compare our system with the established and effective BYTE approach (Zhang et al., 2021a). Additionally, we compare the tracking outcomes using original Mask R-CNN detections versus those using refined masks to evaluate the impact of our refinement process. We also explore the effectiveness of different motion models for creating the basis tracklets.

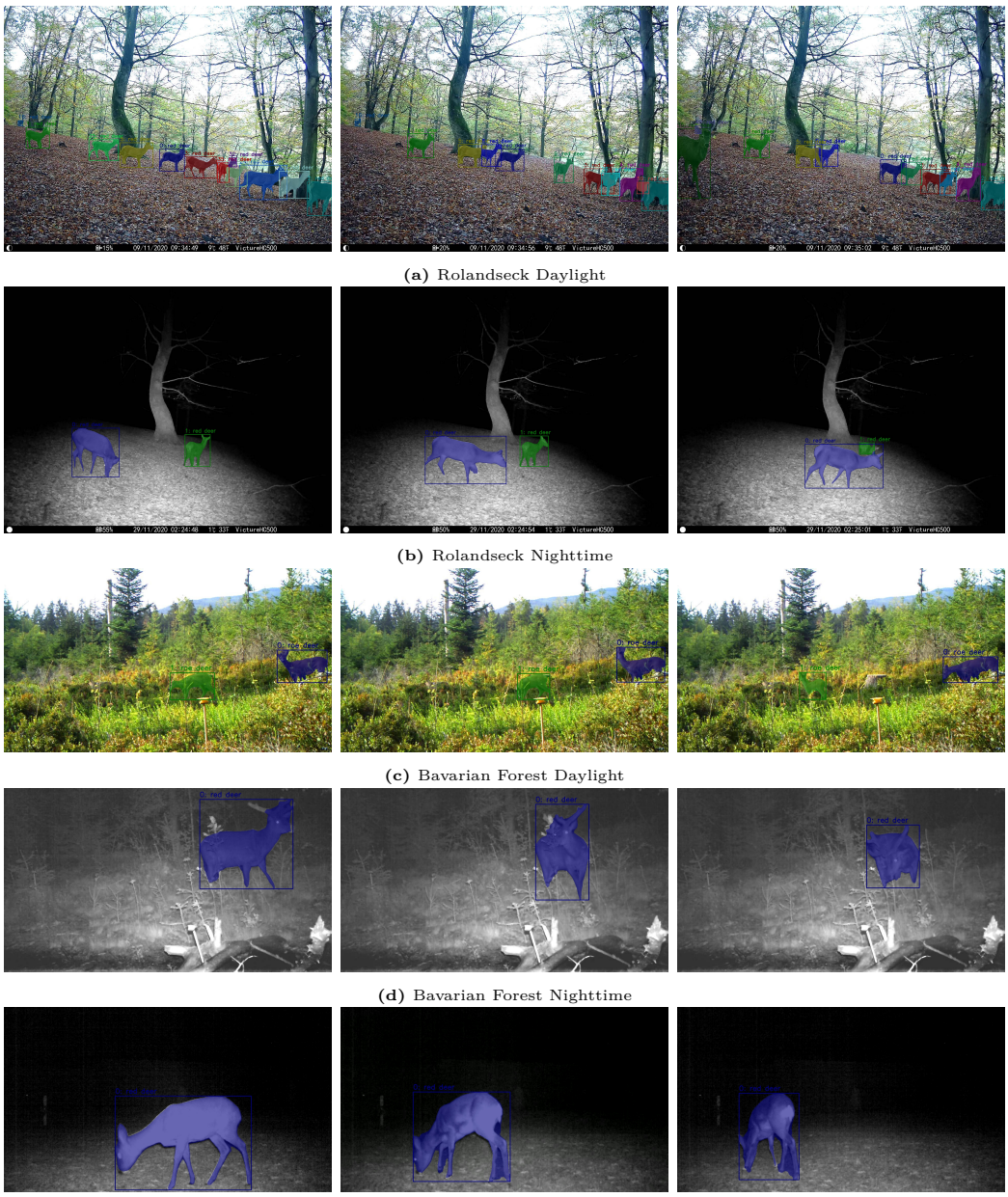
Within our tracking algorithm, we utilize a Re-ID network to facilitate re-identification. We use the OS-Net (Omni-Scale Network) (Zhou et al., 2019), trained over 10 epochs with a learning rate of 0.0003 using frames extracted around the animals. For training, we extract a bounding box with a fixed aspect ratio around each animal, resizing it to 256x128 pixels. This horizontal format better suits the animals in our

dataset compared to the typical vertical 128x256 pixel format used in person Re-ID, prompting us to adjust the network inputs accordingly. The OS-Net achieves nearly perfect mAP scores, and the Rank-1 accuracy is almost 100.0%. These excellent Re-ID results can be attributed to the nature of the task, which involves recognizing individual animals within a single video. Since the Re-ID network operates on an image basis, it benefits from receiving many similar images due to the high frame rate of our videos, particularly when the animals move minimally. This similarity greatly aids in the re-identification process. However, recognizing animals across different videos remains a difficult challenge.

In Table 4.2 we present the tracking results of SWIFT using the MOT metrics. We analyze the BYTE tracker (Zhang et al., 2021a) and compare SWIFT using the Mask R-CNN detections and the refined detections. The MOT accuracy MOTA is slightly higher for SWIFT with the refined detections in comparison to the Mask R-CNN detections. The only exception forms the Rolandseck Daylight dataset, where SWIFT with Mask R-CNN detections is slightly better. The relatively poor tracking results for wildlife crossings can be explained by the correspondingly inaccurate detection results for this dataset. Since both SWIFT and BYTE are tracking-by-detection methods, the tracking accuracy is highly dependent on the quality of the detections. The BYTE tracker performs worse than SWIFT. This can be explained by the fact that BYTE is very dependent on the quality of the detections. BYTE does not delete any detections and also considers low-confidence detections as possible starting points for new tracklets. This shows how important the several filtering steps in our tracking algorithm, in which tracklets are deleted and merged, are for the wildlife monitoring application scenario.

The Rolandseck Daylight dataset contains 200 GT (ground truth) tracks. For SWIFT with refined detections 154 tracks are MT (mostly tracked), 35 tracks are PT (partially tracked) and 11 tracks are ML (mostly lost). The Rolandseck Nighttime dataset consists of 99 GT, where SWIFT with refined detections achieves 76 MT, 16 PT, and 7 ML. The Bavarian Forest Daylight dataset contains only 60 GT tracks, where SWIFT with refined detections achieves 47 MT, 10 PT, and 3 ML. The Bavarian Forest Nighttime dataset consists of 33 GT and SWIFT with refined detections achieves 11 MT, 11 PT, and 11 ML. Finally, the Wildlife Crossings dataset has 88 GT, where SWIFT with refined detections delivers 4 MT, 8 PT, and 76 ML.

We illustrate exemplary tracking results achieved by SWIFT for all five datasets in Figure 4.9. For each example, we consider three frames. For the Rolandseck datasets there are 200 frames between the shown frames, for the Bavarian Forest Daylight dataset 150 frames, for the Bavarian Forest Nighttime dataset 90 frames and for the Wildlife Crossings dataset 30 frames. We color each detected track in a different color. Therefore, the color does not correspond to the animal class in this visualization. As it can be seen, the more complex tracking tasks in the Rolandseck datasets are handled quite well. The other three datasets offer mainly easier tracking scenarios because there are often only single animals in the videos.



**(a)** Rolandseck Daylight

**(b)** Rolandseck Nighttime

**(c)** Bavarian Forest Daylight

**(d)** Bavarian Forest Nighttime

**(e)** Wildlife Crossings

**Figure 4.9:** Visualization of exemplary multi-object tracking and segmentation results of SWIFT for all five datasets: Three frames of a video are displayed. For the Rolandseck datasets there are 200 frames between the frames shown, for Bavarian Forest Daylight there are 150 frames between the frames shown, for Bavarian Forest Nighttime there are 90 frames between the frames shown and for Wildlife Crossings there are 30 frames between the frames shown. Each detected track is colored differently (the color does not correspond to the object class).

| Dataset | Method | MOTA ↑ | MOTP ↑ | IDF1 ↑ |
|---|---|---|---|---|
| Rolandseck Daylight | BYTE | 4.3% | 81.2% | 13.4% |
| | **SWIFT** with Mask R-CNN | **76.3%** | **83.4%** | **73.9%** |
| | **SWIFT** with refinement | 75.1% | 83.9% | 73.7% |
| Rolandseck Nighttime | BYTE | 15.8% | 14.4% | 30.3% |
| | **SWIFT** with Mask R-CNN | 42.7% | 76.8% | 57.3% |
| | **SWIFT** with refinement | **77.8%** | **86.4%** | **76.1%** |
| Bavarian Forest Daylight | BYTE | 13.6% | 83.4% | 53.2% |
| | **SWIFT** with Mask R-CNN | 83.7% | 85.1% | **81.2%** |
| | **SWIFT** with refinement | **84.6%** | **88.7%** | 80.8% |
| Bavarian Forest Nighttime | BYTE | −36.2% | 64.8% | 58.8% |
| | **SWIFT** with Mask R-CNN | 51.8% | 77.5% | **61.0%** |
| | **SWIFT** with refinement | **53.1%** | **80.1%** | 60.7% |
| Wildlife Crossings | BYTE | −33.8% | 36.7% | 20.3% |
| | **SWIFT** with Mask R-CNN | 17.3% | 83.6% | 28.8% |
| | **SWIFT** with refinement | **17.4%** | **90.1%** | **29.1%** |

**Table 4.2:** Tracking comparison: We compare the tracking results of BYTE with our SWIFT Tracking Algorithm. For SWIFT we differentiate between Mask R-CNN detections and the refined detections from our refinement algorithm. The depicted metrics are the MOT metrics. The best results for each dataset are shown in blue.

| Dataset | Motion Model | MOTA ↑ | MOTP ↑ | IDF1 ↑ |
|---|---|---|---|---|
| Rolandseck Daylight | Kalman filter | **75.6%** | **84.0%** | **74.4%** |
| | UKF | 75.5% | 83.9% | 74.0% |
| | particle filter | 75.1% | 83.9% | 73.7% |
| Rolandseck Nighttime | Kalman filter | 76.8% | **86.4%** | **76.8%** |
| | UKF | **78.0%** | **86.4%** | 75.6% |
| | particle filter | 77.8% | **86.4%** | 76.1% |
| Bavarian Forest Daylight | Kalman filter | **85.0%** | **88.7%** | 80.8% |
| | UKF | **85.0%** | **88.7%** | **80.9%** |
| | particle filter | 84.6% | **88.7%** | 80.8% |
| Bavarian Forest Nighttime | Kalman filter | **53.1%** | **80.1%** | **60.7%** |
| | UKF | 51.7% | **80.1%** | 58.8% |
| | particle filter | **53.1%** | **80.1%** | **60.7%** |
| Wildlife Crossings | Kalman filter | 17.3% | **90.1%** | 29.0% |
| | UKF | **17.5%** | **90.1%** | **29.3%** |
| | particle filter | 17.4% | **90.1%** | 29.1% |

**Table 4.3:** SWIFT Motion model comparison: We compare the Kalman filter, the UKF and the particle filter as a motion model for our tracking algorithm. The best results for each dataset are shown in blue.

### Analysis of the Motion Model

In this section we analyze the motion model that we use in the first step of our tracking algorithm 4.3 to detect the basis tracklets. Table 4.3 presents the performance of our tracking algorithm using various motion models to generate the initial tracklets. We evaluate the effectiveness of the Kalman filter, the Unscented Kalman filter (UKF), and the particle filter. The results indicate that all three variants achieve similar overall tracking results. The reason for this is that the subsequent filtering steps in our algorithm, which delete and combine tracklets, are essential for the final tracks. Of course, the tracking scenarios under consideration also play a decisive role in this. Many videos do not contain very large groups of animals, which means that the theoretical advantages of the particle filter do not always apply in practice.

Generally, particle filters are more adept for multi-object tracking compared to Kalman filters and UKFs. This advantage arises because Kalman filters are limited to linear systems, and UKFs, while capable of handling non-linear systems, are not multimodal and struggle with occlusions (Labbe, 2014). Additionally, research by Marron et al. (2007) highlights that particle filters outperform Kalman filters in complex scenarios such as tracking multiple objects.

| Dataset | Method | sMOTSA ↑ | MOTSA ↑ | MOTSP ↑ |
|---|---|---|---|---|
| Rolandseck Daylight | **SWIFT** with Mask R-CNN | 35.3% | 56.1% | 73.1% |
| | **SWIFT** with refinement | **42.6%** | **58.4%** | **78.3%** |
| Rolandseck Nighttime | **SWIFT** with Mask R-CNN | −35.1% | −24.2% | 68.3% |
| | **SWIFT** with refinement | **54.3%** | **68.0%** | **82.6%** |
| Bavarian Forest Daylight | **SWIFT** with Mask R-CNN | 34.1% | 49.5% | 79.2% |
| | **SWIFT** with refinement | **41.3%** | **51.3%** | **86.3%** |
| Bavarian Forest Nighttime | **SWIFT** with Mask R-CNN | 11.2% | 21.3% | 75.1% |
| | **SWIFT** with refinement | **14.9%** | **22.4%** | **82.4%** |
| Wildlife Crossings | **SWIFT** with Mask R-CNN | 7.0% | 11.2% | 70.8% |
| | **SWIFT** with refinement | **9.3%** | **11.3%** | **86.3%** |

**Table 4.4:** MOTS comparison: The MOTS metrics for SWIFT with the Mask R-CNN detections and with the refined detections are shown. The best results for each dataset are shown in blue.

### 4.5.8 Multi-Object Tracking and Segmentation Evaluation

Finally, we analyze the MOTS metric results for SWIFT using the Mask R-CNN detections in comparison to the refined detections in Table 4.4. This comparison highlights a notable enhancement in the MOTS (Multi-Object Tracking and Segmentation) task when utilizing the refined instance masks in SWIFT, as opposed to relying solely on basic Mask R-CNN detections. While the improvement in the MOTA scores for all datasets in Table 4.2 is small, there is a bigger increase in the MOTSA values. Here, the MOTSA value for Rolandseck Daylight is also higher for the refined detections compared to the Mask R-CNN detections, where the MOTA value in Table 4.2 was slightly higher for the Mask R-CNN detections. This demonstrates that refining the instance masks within the SWIFT framework yields more accurate tracking outcomes than using unrefined Mask R-CNN detections alone. Additionally, the sMOTSA value, which assesses the combined quality of segmentation and tracking, also sees a notable difference between unrefined and refined detections, further underscoring the benefits of using SWIFT with refined detections.

Exemplary MOTS results of SWIFT are shown in Figure 4.9. As SWIFT uses instance masks for the tracking process, the results of multi-object tracking and MOTS are visually identical. In general, bounding boxes without the instance masks would be sufficient for the multi-object tracking.

### 4.5.9 Domain Adaptation from Rolandseck to Bavarian Forest

When training and evaluating the deep learning models on the different datasets, the question arises as to whether a model that has been trained on one dataset can

be directly transferred to another. This is generally a complex problem for deep learning applications and is known as domain shift. This means that the (source) distribution for the training set is different to the (target) distribution of the test set (Kouw and Loog, 2018). There are also many works that attempt to perform domain adaptation in the area of detection and tracking (Chen et al., 2018b; He and Zhang, 2019; Zhang and Zhang, 2021). Domain shift is a common problem in applications with medical data, for example, when different scanners and sensors are used to record the data (Stacke et al., 2019; Sheikh and Schultz, 2020). This problem is also present in wildlife monitoring, as different camera models with different resolutions and FPS rates are used to record the videos. Different camera traps were also used for our different datasets.

There are various techniques that can help to perform a domain adaptation. A particularly basic one is the use of many and strong data augmentations. By applying different data augmentations, a deep learning model generalizes better and can thus be transferred more easily to another dataset (Sheikh and Schultz, 2020). Early stopping during training can also help to prevent overfitting on the data (Zheng and Yang, 2022). Another option is to fine-tune the network by transfer learning on selected samples of the target dataset (Zhuang et al., 2020). However, this requires already annotated data for the desired target dataset.

We try out basic domain adaptation techniques to transfer SWIFT trained on the Rolandseck Daylight dataset to the Bavarian Forest Daylight dataset. In the same way, we also investigate SWIFT trained on the Rolandseck Nighttime dataset to be transferred to the Bavarian Forest Nighttime dataset. We choose the Rolandseck datasets as source datasets because they are larger and have a higher resolution (cf. the dataset Chapter 3). For the domain shift, we only analyze the object detection, in particular the instance segmentation, since the tracking is directly dependent on the detection results and therefore no further influence of the domain shift plays a role there. The target datasets, the Bavarian Forest datasets, were recorded with different camera traps and show a lower resolution and lower FPS. Most importantly, different animal species occur in the Bavarian Forest datasets and in the Rolandseck datasets. Fallow deer and red deer occur in Rolandseck, whereas red deer and roe deer occur in Bavarian Forest (cf. Figure 4.10). This makes a direct transfer of the classification part of the detector impossible. In order to enable a domain shift, we generalize the animal classes to an animal class deer. Moreover, the vegetation in Rolandseck and in the Bavarian Forest National Park is very different. In the Bavarian Forest, significantly more shrubs, bushes and grasses can be seen in the videos. As a result, there are more occlusions of the animals, while at the same time the animals stand out more easily from the mostly green background, which makes them easier to detect. In Rolandseck, the surroundings are characterized by brown tones, such as leaves lying on the ground. This makes it more difficult to detect the animals, whose fur color is similar to the color tones of the surroundings.

Furthermore, we are extending our existing data augmentation techniques and are

| Dataset | Method | $AP^{bbox}$ | $AP^{mask}$ | $AP^{mask}_{0.50}$ | $AP^{mask}_{0.75}$ | $AR^{mask}$ |
|---|---|---|---|---|---|---|
| Bavarian Forest Daylight | SWIFT trained on Bavarian Forest Daylight | **0.621** | **0.609** | 0.841 | **0.700** | **0.729** |
| | SWIFT trained on Rolandseck Daylight | 0.559 | 0.538 | **0.843** | 0.612 | 0.608 |
| Bavarian Forest Nighttime | SWIFT trained on Bavarian Forest Nighttime | 0.285 | **0.315** | 0.586 | **0.305** | **0.490** |
| | SWIFT trained on Rolandseck Nighttime | **0.292** | 0.305 | **0.617** | 0.224 | 0.394 |
| Rolandseck Daylight | SWIFT trained on Rolandseck Daylight | **0.583** | **0.497** | **0.806** | **0.545** | **0.579** |
| | SWIFT trained on Bavarian Forest Daylight | 0.310 | 0.269 | 0.445 | 0.280 | 0.312 |
| Rolandseck Nighttime | SWIFT trained on Rolandseck Nighttime | **0.622** | **0.537** | **0.840** | **0.579** | **0.622** |
| | SWIFT trained on Bavarian Forest Nighttime | 0.378 | 0.317 | 0.536 | 0.304 | 0.411 |

**Table 4.5:** Domain adaptation: The column *Dataset* describes the target dataset of the domain adaptation. The source datasets are mentioned in the *Method* column as *trained on*. The depicted metrics are the standard COCO metrics. The best results are shown in blue.

also introducing a Gaussian blur and random cropping. This should make the Mask R-CNN even more robust. For the refinement we use SAM with the bounding box input. This has achieved the best results on the Rolandseck Daylight dataset and can create more accurate masks in this transfer scenario due to the greater degree of freedom (in relation to setting the positive and negative clicks) in order to compensate for the difficulty of the non-occurring roe deer samples in the training dataset. Our results for the domain shift are presented in Table 4.5.

**Figure 4.10:** Exemplary frames from the source dataset Rolandseck Daylight (left) and the target dataset Bavarian Forest Daylight (right) show the difficulties for performing a successful domain adaptation since Rolandseck contains fallow deer and Bavarian Forest contains roe deer. Moreover, there is huge difference in the vegetation of both locations.

With the domain adaptation techniques described above, SWIFT trained on the Rolandseck Daylight dataset can almost achieve the accuracy of the network trained on the Bavarian Forest Daylight dataset. In the $AP^{mask}_{0.50}$ metric, this network is even slightly better. In order to achieve a better transfer, samples from the target dataset could now be taken for further transfer learning. Since this is only a basic experiment to show the adaptability of our model, we do not use deeper and more complex domain adaptation techniques. Domain adaptation for the Nighttime dataset is in general easier, as the videos in Rolandseck and the Bavarian Forest do not differ that much in color. Nevertheless, the videos were created with different camera traps and recording techniques. In the Rolandseck Nighttime recordings, it can be seen that the central area is illuminated more brightly by the infrared light and the sides of the image are darker. We show this exemplary in Figure 4.11. In the Bavarian Forest, the night shots were taken with uniform lighting. The domain adaptation results show that the approach trained on Rolandseck Nighttime achieves similar good results as the original approach trained on the Bavarian Forest dataset. For the $AP^{mask}_{0.50}$ metric the network trained on the Rolandseck Nighttime dataset even surpasses the results of the other approach. We also investigate the domain shift from the other perspective, where the source and target datasets are switched. As already theoretically explained, this domain adaptation is not as successful as the domain adaptation with the source dataset Rolandseck. But with the described domain shift techniques the network trained on the Bavarian Forest datasets achieves good results for this difficult task.

In order to achieve a better adaptation to the target datasets, fine-tuning would

**Figure 4.11:** Exemplary frames from the source dataset Rolandseck Nighttime (left) and target dataset Bavarian Forest Nighttime (right) show the difficulties for performing a successful domain adaptation. The camera traps in Rolandseck Nighttime use an infrared flash, which leads to a brighter center and darker sides of the scene. The Bavarian Forest Nighttime shots are darker but more evenly lit.

have to be carried out with the help of transfer learning. To do this, samples would have to be selected from the respective target dataset on which the Mask R-CNN is trained further. Successful transfer learning has been achieved in other areas of computer vision in the work of (Zhuang et al., 2020; Hu and You, 2020), for example. This would certainly improve the accuracy of the night datasets, where the recording methods are very different. Since we limit ourselves here to basic experiments and investigate the general transferability of our approach to an unknown dataset, we do not carry out any further transfer learning experiments.

## 4.6 Discussion

Our findings indicate that SWIFT excels not only in the MOTS task for wildlife camera trap videos but also in the subtasks of instance segmentation and multi-object tracking. By integrating the robust Mask R-CNN system (He et al., 2017) with a subsequent refinement step, we significantly enhanced instance segmentation. However, for application areas requiring (near) real-time recognition, such as pedestrian detection in autonomous driving, this additional refinement step would be impractical due to the extra time it demands. Consequently, SWIFT is an offline tracking approach and not suited for real-time detection and (online) tracking applications.

In the context of wildlife monitoring, the urgency for real-time video analysis is generally low. Often, data from camera traps are reviewed only months later due to the sheer volume of video collected. More precise instance masks are particularly beneficial in distinguishing individual animals within larger groups, such as herds of red deer. In scenarios where simple bounding boxes might encapsulate multiple animals, our refined instance masks ensure that each animal is distinctly recognized. This improvement in action detection reliability is invaluable for ecologists, enhancing their ability to analyze animal behavior. This leads to more accurate behavioral

analyses and insights, which are critical for conservation and ecological studies (like (van Beeck Calkoen et al., 2021)).

We discovered that semi-automatic instance segmentation methods yield highly accurate instance masks for annotating wildlife data. This insight formed the foundation for our approach, which aims to automate these methods by replacing manual user inputs (such as clicks for object segmentation) with masks generated by a reliable automatic instance segmentation method. Our SWIFT Refinement Algorithm facilitates the automatic setting of "clicks", using the detection results from Mask R-CNN as input instead of manually annotated ground truth masks. To limit the uncertainty caused by the automation of clicks and to ensure reliable results, we use two techniques in particular: First, by automatically adjusting the scale (zooming in and out by erosion and dilation) of the Mask R-CNN instance masks to reliably set clicks inside and outside the detected animal; second, by utilizing the setmask function from the approach of Sofiiuk et al. (2021) to initialize the HR-Net with the Mask R-CNN mask, thereby avoiding significant alterations that could result from erroneous clicks.

Our experiments demonstrated that the choice of refinement technique is critical. For instance, refinement using the f-BRS approach (Sofiiuk et al., 2020) actually reduced the quality of instance masks due to its instability and lack of integration with Mask R-CNN's segmentation results. In contrast, initializing HR-Net with the Mask R-CNN mask via the setmask function provides a stable and reliable refinement process.

We have also examined the increasingly popular and successful foundation models as possible refinement approaches. Here it was shown that the Segment Anything Model (SAM) even achieved better results than the HR-Net for two of the five datasets. For night images in particular, the results of the foundation model were worse than those of the HR-Net. This shows that when using the foundation models, the data on which the networks were trained and the application data must be taken into account. Our experiments also show that SAM achieves better results with a bounding box as input compared to the input of positive and negative clicks. This is due to the fact that the foundation model has more degrees of freedom when generating masks with the bounding box. In the future, it is expected that foundation models will continue to improve and will probably be the better choice for refinement. However, it must be taken into account what the data looks like in the application context (e.g. night images). Scenarios that are foreign to the foundation model may also result in less accurate masks.

Accurately creating instance masks for animals that are partially in or out of the scene presents particular challenges. For example, when only a leg or parts of the head are visible, Mask R-CNN may struggle to differentiate animal parts from environmental features like branches or tree parts, leading to unreliable detections. Additionally, accurately detecting the legs of animals in a group where they may be

obscured by other animals is challenging, as these parts may incorrectly be assigned to the wrong animal.

For multi-object tracking, we demonstrated that SWIFT outperforms the BYTE tracker (Zhang et al., 2021a) in tracking accuracy. Additionally, we conducted a thorough examination of the individual components within our tracking algorithm.

We found that the choice of the motion model to form basic tracklets is not that important. The particle filter, Kalman filter and unscented Kalman filter (UKF) generate almost the same final tracking accuracy. This shows that the following filtering stages are more important for the tracking. In these stages tracklets are deleted and combined. Especially this functionality is very important for the application of the tracker because there are false detections or partial detections of animals. This is one main difference to the BYTE tracker, which performs poorly on all datasets. The particle filter can handle non-linear movements, which are more characteristic of animal behaviors than the linear models assumed by the Kalman filter (Marron et al., 2007). However, this did not play a major role in the generation, as often only a few animals appear in a video or only a few fast movements take place in larger groups. For other scenarios or other animal classes, such as birds or fish, this advantage will be more important.

Furthermore, our experiments revealed that the refined instance masks produced by our SWIFT Refinement Algorithm significantly enhanced tracking accuracy compared to the basic Mask R-CNN detections. This improvement was particularly notable in the sMOTSA metrics for multi-object tracking and segmentation, as detailed in Table 4.4.

SWIFT operates under the tracking-by-detection paradigm, meaning the efficacy of its tracking is inherently linked to the quality of its instance segmentation detections. This relationship is clearly demonstrated in our evaluations presented in Table 4.4, where SWIFT shows superior tracking performance using refined detections as opposed to basic Mask R-CNN detections. Consequently, future enhancements in instance segmentation are likely to yield further improvements in tracking performance. An additional benefit of SWIFT is its flexibility; the SWIFT Tracking System is designed to potentially utilize detection results from other instance segmentation methods, thereby enabling tracking based on a variety of segmentation outputs. This modularity and adaptability enhances the applicability of SWIFT across different tracking scenarios and datasets.

Both instance segmentation and tracking metrics, including MOTS, can generally be improved through the annotation of additional training material. We created five datasets comprising overall 194 videos, which collectively contain 82,768 individual frames. For context, the well-known tracking datasets MOT16 (Milan et al., 2016) features 14 videos with 11,235 frames, while MOT20 (Dendorfer et al., 2020) includes 8 videos totaling 13,410 frames (both datasets do not contain instance masks, just

bounding boxes and track IDs). This comparison highlights the extensive nature of our datasets.

SWIFT enhances the ability of ecologists to manage and analyze the vast amounts of data generated by camera traps by automating the detection and tracking of animals within video footage. Manual review of such data is immensely time-consuming, necessitating days or even weeks of labor, thus highlighting the need for automation through artificial intelligence, as noted by Green et al. (2020). To prepare SWIFT for recognizing new species in videos, training data must first be generated, which involves manually annotating a substantial number of video sequences. Like all deep learning approaches, SWIFT requires a large, diverse set of training data to perform accurately on unseen data. In our domain adaptation study we showed that this is not an easy task even for similar looking species. If, in the future, a dataset is built up containing videos from different locations and recorded with different camera traps, this will facilitate transferability.

The use of instance masks in conjunction with video data offers deeper and more comprehensive insights compared to traditional image-based bounding box detection approaches like those by (Verma and Gupta, 2018; Beery et al., 2018, 2019; Falzon et al., 2020; Bonneau et al., 2020; van der Zande et al., 2021). Unlike Yang et al. (2019b), who employ bounding box detection for gorillas in camera trap videos, our approach incorporates both instance masks and tracking. Without tracking, linking detections from one frame to another to facilitate action detection would be challenging. Existing image-based instance segmentation studies such as those by Ter-Sarkisov et al. (2018); Salau and Krieter (2020); Bello et al. (2021); Hu et al. (2021) are typically confined to indoor environments with overhead cameras, making them less relevant to the dynamic and uncontrolled conditions of wildlife monitoring. Our datasets represent a more typical setup for camera trap studies and poses greater challenges. Furthermore, while Zeppelzauer (2013) track elephant segments through videos, they do not provide metrics for the accuracy of their segments as instance masks, nor do they compute MOT metrics to measure tracking accuracy, making their study less comparable to ours in terms of methodological complexity and the detailed analysis required for effective wildlife monitoring.

SWIFT operates without any user intervention during the detection and tracking processes, which is a notable advantage over methods like the one proposed by Xue et al. (2021), where users must manually set a guidance instance mask in each video frame to facilitate instance segmentation and subsequent tracking.

## 4.7 Conclusions

For the most accurate instance segmentation possible, we combine a reliable and popular CNN, the Mask R-CNN, to detect and classify the animal instances, and a refinement network to further increase the quality of the instance masks. We

have also shown how important it is to select a suitable refinement network and at the same time we have examined the new foundation models in this context. In particular, we utilize the instance mask information in our designed tracking algorithm. We use several filtering stages in which basic tracklets are combined and deleted if necessary. In this way, we reduce the influence of false detections on the tracking accuracy. We were able to show the improvement in tracking accuracy through the instance masks by applying the MOTS metrics.

# Chapter 5

# Action Recognition and Action Detection

In this chapter, we outline our strategy for action recognition and action detection. Action recognition, also called action classification, aims to identify the action performed by a single actor in a short sequence. On the other hand, action detection identifies all actors within a video and subsequently recognizes the actions of each detected actor. We introduce our innovative action recognition network, MAROON (Mask guided Action RecOgnitiON), in Section 5.3. Prior to that, we describe our comprehensive system for action detection, which is based on SWIFT and MAROON. The main results of this chapter are published in (Schindler et al., 2024).

## 5.1 Introducing Action Classes, Action Recognition and Action Detection

Action detection aims to detect all actors and to classify their performed actions in a video. To perform action detection the general tasks of (object) detection and action recognition are combined. We have already described how detection works, in particular the instance segmentation, in Section 4.1. The task of action recognition is to assign an action class $c$ to a video sequence of fixed length $t$, where only one actor is present. Usually $t$ is 8, 16 or 32 frames. Longer sequences require more GPU memory for training neural networks. In addition, longer sequences can become too long for temporally short actions. In this case, different actions would be represented in one sequence, which would make classification impossible. In Figure 5.1 we show visually the differences between action recognition and action detection.

The usual input size of the frames for an action recognition network is square, for example $m \times m$. A common value for $m$ is 256 pixels. This means that either a frame has to be resized or a square patch has to be cut out from the whole frame. The second strategy is the common approach for action detection. The object detector localizes the actor and the bounding box of the detection forms the basis for a square


**(a)** Action recognition **(b)** Action detection

**Figure 5.1:** Comparison between action recognition and action detection: The action recognition (a) only classifies one action from one actor in a frame. The action class is written in the top left corner. If there are multiple actors present, the outcome would be unclear. The action detection (b) localizes each actor and describes its action. The predicted action is written above each bounding box. A classification and tracking of actors is not necessarily included in the action detection.

cutout. The action of the cutout actor is then classified by the action recognition approach.

## 5.2 Action Detection System

The workflow of our action detection approach is shown in Figure 5.2. First, our instance segmentation system SWIFT (Schindler and Steinhage, 2022) detects and tracks all animals in the video data. We already explained and evaluated SWIFT in the previous Chapter 4. This represents the detection part of the action detection. Utilizing the bounding boxes and track IDs from the detected animals, we extract them from the video as square cutouts to ensure they are suitable for action recognition without being distorted (e.g. compressed or stretched). Subsequently, the instance masks from SWIFT are used to cut out the exact animal's shape. This step is crucial, particularly when multiple animals are in close proximity, ensuring each individual is distinctly separated in the cutout. Moreover, isolating the animal from its background enhances the system's ability to generalize across various unseen environments. These cutouts are then resized to match the input dimensions required by our action recognition network MAROON. The action recognition process then identifies the action class, culminating in a comprehensive action detection result that includes the action class, instance mask, bounding box, species classification, confidence score, and track ID.

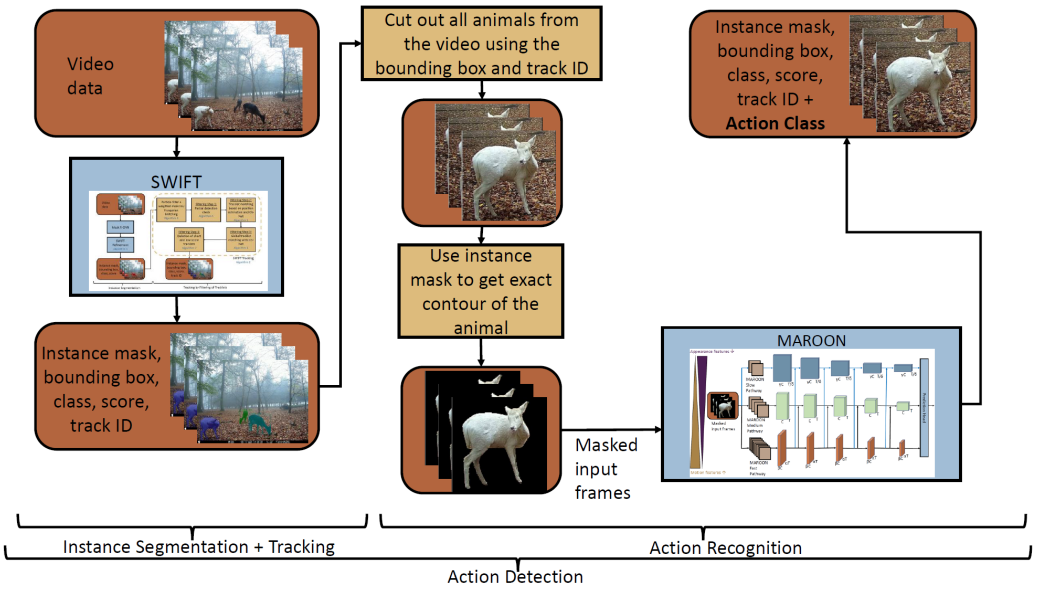



**Figure 5.2:** Our action detection system, a combination of SWIFT and MAROON. The instance segmentation and tracking are performed by SWIFT that we described in Chapter 4. The bounding boxes and instance masks found are then used to cut out the animals. These frames are then fed as input into the MAROON action recognition network, which will be explained in the following section. The result from the action recognition network is an action label for the input frames.

## 5.3 Action Recognition with MAROON

Here, we explain in detail the architecture and idea of our action recognition network MAROON - Mask guided Action RecOgnitiON. In Figure 5.3 we present the architecture of MAROON. We highlight in blue the pathways and lateral connections that we newly introduce in comparison to the base model of SlowFast (Feichtenhofer et al., 2019). Our network introduces two key innovations: (1) the employment of masked input frames, and (2) the implementation of a triple-stream approach. As previously mentioned, the input frames of the action recognition network are first cutout from the overall video frame using the bounding boxes detected by SWIFT and then masked by the instance mask of the animal. This technique enables MAROON to concentrate solely on the actor during feature extraction, preventing the network from learning irrelevant background information.

Our network architecture builds on the idea of SlowFast (Feichtenhofer et al., 2019). SlowFast introduces a two-stream network, consisting of two pathways, a slow pathway and a fast pathway. The task of the fast pathway is to extract motion features (for example, the type of movement and speed of the animal) and the slow pathway focuses on the appearance features (for example, the color and the pose of the animal). To accomplish this, the approach involves adjusting the quantity of input frames, denoted as $T$, for each stream, and altering the number of feature channels, represented as $C$, of the convolutional layers of each respective stream. For this,

the parameters $\alpha$ and $\beta$ are introduced. For MAROON, we expand upon this concept by transitioning from a two-stream to a triple-stream approach, allowing us to extract motion and appearance features with varying levels of detail. In Figure 5.3 we present the architecture of MAROON. Accordingly, we name the three pathways MAROON slow pathway, MAROON medium pathway and MAROON fast pathway. We newly introduce the parameters $\gamma$ and $\delta$. With $\alpha$ and $\delta$ ($\alpha, \delta > 1$) we control the amount of frames for each of the three pathways. The MAROON fast pathway gets the most densely sampled $\alpha T$ frames. The MAROON medium pathway processes $T$ frames and the MAROON slow pathway handles a reduced number of $T/\delta$ frames. For instance, given an input sequence of 16 frames, the distribution might involve 16 frames for the fast pathway, 4 frames for the medium pathway, and 2 frames for the slow pathway. The second distinction among the three pathways lies in their channel capacity, $C$, which is regulated by two parameters, $\beta$ and $\gamma$. An increased number of channels facilitates the extraction of more detailed features, particularly useful when analyzing an animal's appearance in detail. The medium pathway has $C$ channels. The parameters $\beta$ and $\gamma$ ($\beta < 1, \gamma > 1$) adjust the channel quantity for the remaining two pathways, tailoring the level of feature detail they can extract. With $\beta C$ channels the fast pathway has less channels than the medium pathway allowing it to focus on extracting motion features. The slow pathway has with $\gamma C$ channels the highest number of channels of all three pathways. To summarize, the more frames are fed into the pathway the lower the channel capacity and vice versa.

Beyond the two-stream architecture, a key innovation of SlowFast (Feichtenhofer et al., 2019) involves the integration of both streams through lateral connections, allowing for the fusion of features from the fast pathway with those of the slow pathway. We build upon this concept by similarly integrating our third pathway, merging the fast pathway's features with those of our new MAROON slow pathway. The features from all three pathways are then combined in the prediction head, which produces the final action class prediction.

## 5.4 Results

In this section, we present our evaluation results of MAROON and compare them with the state-of-the-art approaches SlowFast (Feichtenhofer et al., 2019) and MViT (Fan et al., 2021). Furthermore, we conduct ablation studies to prove our architecture's functionality and the choice of our parameters.

### 5.4.1 Training and Testing Details

We selected SlowFast for comparison due to its notable success as a CNN approach, especially since MAROON draws inspiration from and extends SlowFast's concepts.

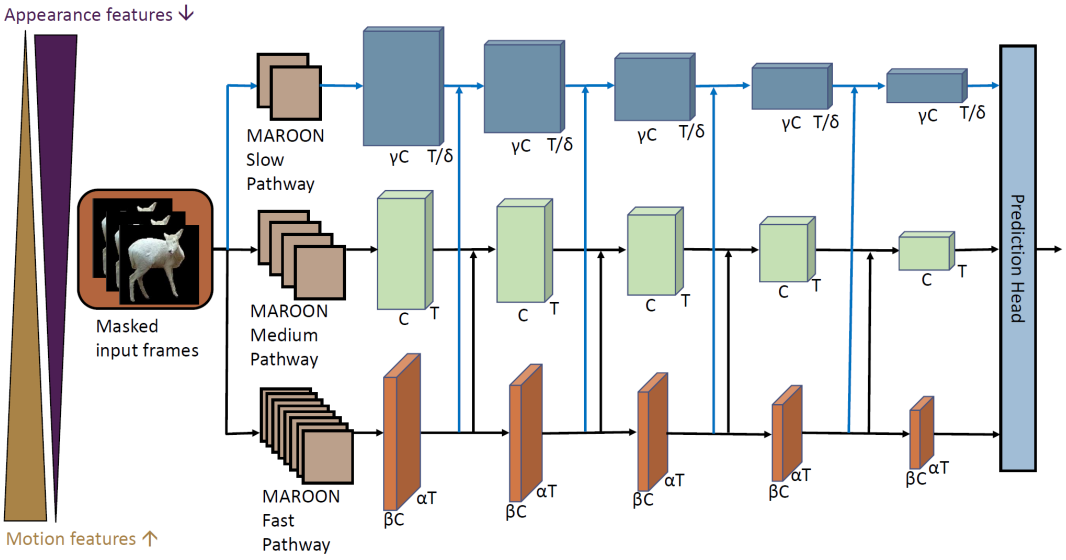



**Figure 5.3:** Our action recognition network MAROON uses three streams and masked input frames. Our improvements compared to the basic SlowFast architecture are marked in blue, i.e. the new MAROON Slow Pathway with the corresponding lateral connections and the input of masked input frames. The more input frames are entered into a path, the more motion features are extracted. However, if fewer frames are entered and there are more channels in the convolution blocks, mainly appearance features are extracted. This is shown by the colors of the convolution blocks. The orange blocks in the Fast Pathway mainly extract motion features, the green blocks in the Medium Pathway a mixture of motion and appearance features and the blue blocks in the Slow Pathway mainly appearance features.

MViT stands out as a prominent transformer model recognized for its effectiveness in action recognition with human subjects. The parameters for these models were determined based on recommendations from their respective publications and through extensive testing.

We base the maximum length of the input sequences on the action classes which describe temporally short actions such as head raising or sudden rush. For the Rolandseck Daylight and Rolandseck Nighttime datasets we choose an input length of 16 frames. Since this dataset has a high FPS of 30, fast actions are also well covered. For the other datasets Bavarian Forest Daylight, Bavarian Forest Nighttime and Wildlife Crossings we limit the input to 8 frames due to their lower FPS rates. The results of the Wildlife Crossings dataset should be evaluated with caution, as this dataset is not only the smallest of the five datasets and therefore has the lowest number of training samples, but it also contains very different animal classes, i.e. red deer, boar, hare and fox. For the sake of completeness, we also include this dataset in our action evaluations.

As shown in Section 3.4.1 the action class distributions in our datasets are long tailed distributions. To ensure accurate predictions, it's crucial to give special attention to underrepresented classes, which suffer from a scarcity of training samples. To enhance action recognition performance for these classes, we employ oversampling

during training. In each epoch we randomly sample from the smaller classes, so that each class represents at least 50% of the number of observations of the largest class (the experiments for determining this value are done in Section 5.4.5). For the already mentioned special case of the Wildlife Crossings dataset it is 100%. In general, the oversampling parameter depends on the dataset that is analyzed, especially on the distribution of the action classes, the size of the dataset and the variability of the data samples. Therefore, we determine this parameter through experiments. Another theoretical reason to restrict the resampling to 50% (and not 100% in general) is that our smallest classes have as few as 5 samples and the largest class more than 100 samples. Even with data augmentation strategies, these samples are presented to the network very frequently. By setting this limit, we aim to avoid overfitting to individual samples.

During training we enhance our dataset with random horizontal flips. Moreover, we perform temporal and spatial jittering as data augmentation techniques. In general, the action sequences are longer than the specified input length for the networks, which is either 16 or 8 frames in our cases. For temporal jittering, we randomly select a fixed-length sequence from the entire action sequence, ensuring the model is exposed to different segments of the action with each epoch to minimize overfitting. As described before, we cut out each animal by their bounding box. For a spatial jittering the box is cut out a little bit larger than the desired input size for the network. From this enlarged section, we then randomly crop to the exact input dimensions. This technique ensures the animal isn't always centered, thereby introducing more variation into the training data and enhancing the model's generalizability. For comparable testing, the inputs must remain uniform, so spatial jittering is omitted. However, to encompass various time points within an action sequence, we divide the sequence into equal parts for testing, a method known as ensemble view testing, specifically employing a 10-view testing approach. This strategy allows us to thoroughly evaluate the model's performance across different segments of an action sequence.

To evaluate the performance of networks in action recognition, we employ the common metric of top-1 and top-5 accuracy. Essentially, top-$k$ accuracy indicates a sample is correctly predicted if the network's $k$ highest-ranked predictions include the accurate action class for that sample. For the task of action detection, a universal metric is not established, though mean average precision (mAP) is occasionally utilized, as seen in certain studies (for example in (Chen et al., 2023; Biswas and Gall, 2020)). A limitation of mAP is that it primarily measures the accuracy of bounding boxes produced by the detector, neglecting aspects like instance masks, and thus heavily linking the action class prediction to the detector's performance. By evaluating the detector and action recognition components separately, each part is more equitably assessed, simplifying the decision process for potential replacements within the overall system.

All our models are trained for 200 epochs. For a fair comparison among the models,

we do not use pre-trained weights as initialization. For MAROON and also SlowFast we use the ResNet-50 as backbone. For the Rolandseck Daylight and Rolandseck Nighttime dataset the input sequence length is 16 frames. We set the parameters for MAROON for all datasets as $\alpha = 4, \beta = 1/8, \gamma = 4$ and $\delta = 8$. We determine these parameters through extensive experiments that are shown in the following ablation studies. The parameters $\alpha$ and $\delta$ lead to the input sizes 16 frames for the fast pathway, 4 frames for the medium pathway and 2 frames for the slow pathway for the Rolandseck Daylight and Nighttime dataset and respectively 8 frames for the fast pathway, 2 frames for the medium pathway and 1 frame for the slow pathway for the Bavarian Forest datasets and the Wildlife Crossings dataset.

### 5.4.2 Action Recognition Evaluation

We conduct evaluations of our models across all five datasets introduced earlier. These datasets are split into training and testing sets, ensuring that approximately 20% of each class is allocated to the test set. Given the limited volume of data, we opt not to establish a separate validation set. Instead, to demonstrate the superior performance of our model relative to benchmark models, we execute a stratified 5-fold cross-validation on every dataset. "Stratified" here implies that in each fold, 20% of the data from each class is designated as test data, ensuring that the composition of the test sets accurately reflects the action class distribution found within the entire dataset.

Our evaluation results, comparing the different models across different datasets, are summarized in Table 5.1.

MAROON outperforms the other models for all five datasets. On average, all models perform best on the Rolandseck datasets and worst on the Bavarian Forest datasets. This result can be explained by the fact that the Rolandseck datasets encompass the greatest number of sequences, thereby offering the broadest variety of scenarios for training. Conversely, the Bavarian Forest datasets contain the fewest sequences, limiting the diversity of training situations. For all datasets, MAROON improves the top-1 accuracy of the second best comparison model by at least 10 percentage points. This shows efficiency of our innovations for MAROON. The accuracies of the two comparison models are approximately the same. As already mentioned, the Wildlife Crossings dataset is a special case due to its small size and the very different animal species that occur. However, MAROON also achieves a high level of accuracy for this dataset.

In Figure 5.4 we present the top-1 accuracies for all action classes from the 5-fold cross-validation. In general, MAROON consistently surpasses the other models in performance for every class within all five datasets. Notably, MAROON maintains uniformly high top-1 accuracies across different action classes, a trend particularly

| Dataset | Model | Top-1 | Top-5 |
|---|---|---|---|
| Rolandseck Daylight | MAROON | **72.24** | **95.88** |
| | SlowFast | 42.05 | 89.66 |
| | MViT | 43.13 | 85.18 |
| Rolandseck Nighttime | MAROON | **82.33** | **97.15** |
| | SlowFast | 35.06 | 91.46 |
| | MViT | 41.61 | 91.48 |
| Bavarian Forest Daylight | MAROON | **54.27** | **97.59** |
| | SlowFast | 35.40 | 95.88 |
| | MViT | 35.05 | 94.17 |
| Bavarian Forest Nighttime | MAROON | **45.47** | 95.08 |
| | SlowFast | 35.49 | **95.68** |
| | MViT | 31.91 | 93.26 |
| Wildlife Crossings | MAROON | **73.11** | 98.40 |
| | SlowFast | 59.51 | **99.20** |
| | MViT | 53.97 | 96.03 |

**Table 5.1:** Evaluation results of our action recognition model MAROON compared for the five different datasets Rolandseck Daylight, Rolandseck Nighttime, Bavarian Forest Daylight, Bavarian Forest Nighttime and Wildlife Crossings with 5-fold cross validation. The top-1 and top-5 accuracies are depicted. The best value for each dataset is marked blue.

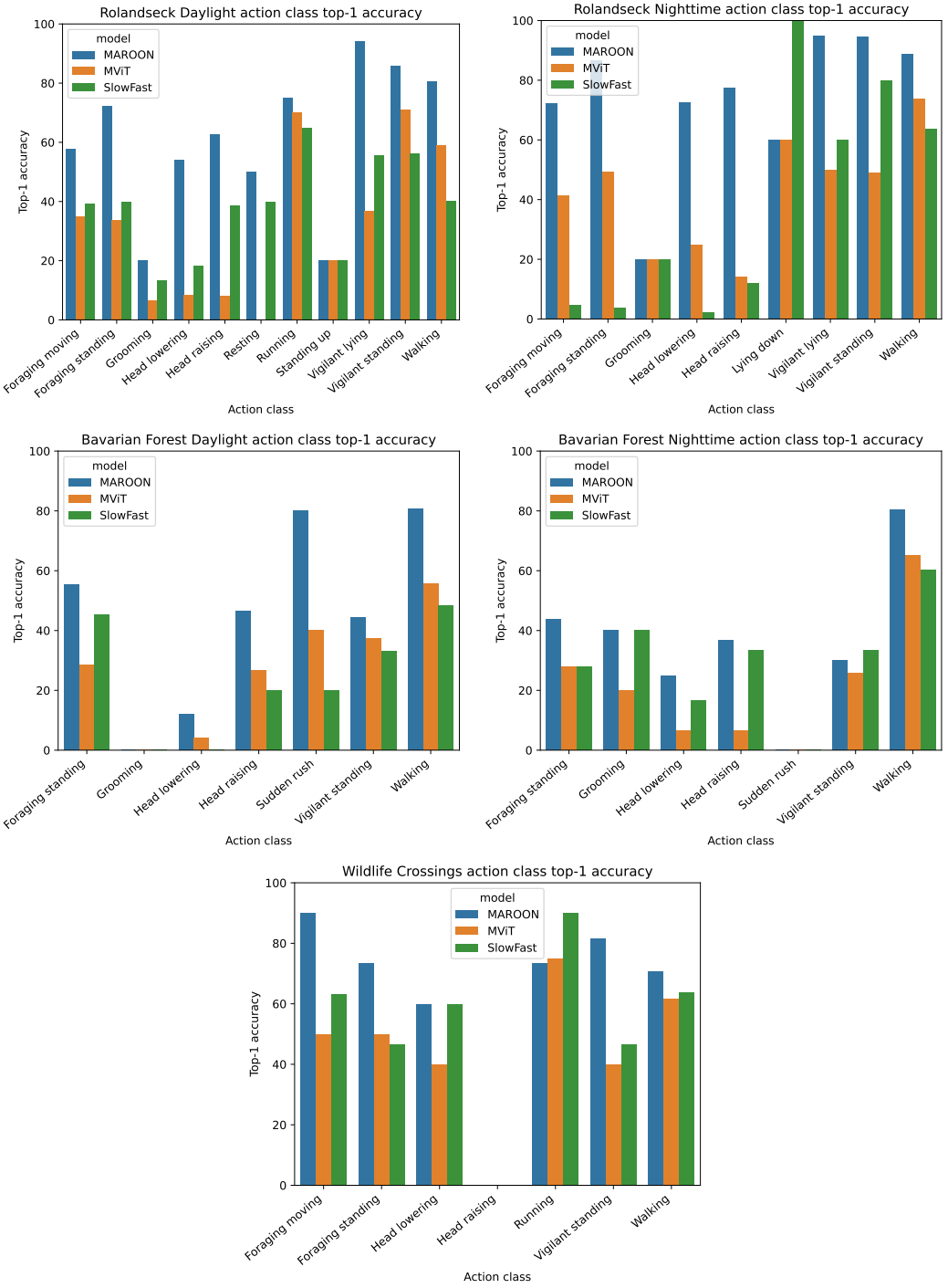



**Figure 5.4:** The action class top-1 accuracy of the cross validation of the five datasets for each action class with the models MAROON (blue), MViT (orange) and SlowFast (green).

evident in the Rolandseck Daylight dataset. This uniformity indicates that MAROON effectively learns to recognize actions not just in well-represented classes but also in those with fewer samples, ensuring a balanced performance across diverse action types.

Figure 5.5 shows exemplary SWIFT action detection results for all five datasets. The animal detections are ground truth detections.

In the following sections, we evaluate the functionality of different parts of our action recognition network. We always use the same 5-fold cross-validation as in this section.

### 5.4.3 The Impact of Masked Input Frames

The theory behind masking the input frames is that this allows the network to fully focus on the actors and the action they are performing and not be distracted by irritations in the background, particularly other animals that may be within the frame being viewed. Another advantage is that the removal of the background prevents the memorization of features from it, which could provide a false basis for the recognition of the actions.

In Table 5.2 we present the impact of the masked input. We evaluate all models with and without masked input frames. So, in comparison to our previous evaluations, we experiment with using masked input frames for both SlowFast and MViT models, and for our MAROON model, we employ regular (non-masked) input frames. The mask information enhances the results for all models on all datasets, yielding an average improvement of 10 percentage points. Notably, our action recognition network, MAROON, consistently outperforms or matches the results of the other methods, regardless of the presence of mask information. This improvement is particularly significant in the Rolandseck Daylight dataset, where mask information boosts MAROON's performance by over 15 percentage points compared to SlowFast and MViT. The MViT transformer model shows the smallest relative gain from mask information among the tested models. This outcome can be attributed to the operational mechanics of transformers, which utilize an attention mechanism focusing on small sections of the image. The introduction of mask information leads to larger areas of uniform blackness within the image, presenting minimal to no visual information about the animal to the transformer, thus limiting its effectiveness in these cases.

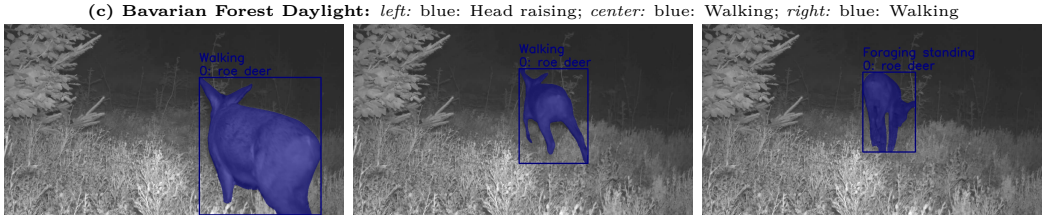
### 5.4.4 Lateral Connections

Additionally, we investigate the significance of the lateral connection between the fast and slow pathways within our MAROON model. Similar to how (Feichtenhofer et al., 2019) assessed the impact of lateral connections in their SlowFast model,

**(a) Rolandseck Daylight:** *left:* blue: Foraging standing, green: Walking, teal: Vigilant lying, red: Walking; *center:* blue: -, green: Walking, teal: Vigilant lying, red: Vigilant standing; *right:* blue: -, green: Head lowering, teal: Vigilant lying, red: Vigilant standing


**(b) Rolandseck Nighttime:** *left:* blue: Foraging moving; *center:* blue: Foraging standing; *right:* blue: Foraging moving, green: Foraging moving, teal: Walking


**(c) Bavarian Forest Daylight:** *left:* blue: Head raising; *center:* blue: Walking; *right:* blue: Walking




**(d) Bavarian Forest Nighttime:** *left:* blue: Walking; *center:* blue: Walking; *right:* blue: Foraging standing


**(e) Wildlife Crossings:** *left:* blue: -, green: Foraging standing, teal: Foraging standing; *center:* blue: -, green: Foraging standing, teal: Foraging standing; *right:* blue: -, green: Foraging standing, teal: Foraging standing

**Figure 5.5:** Visualization of exemplary action detection results of SWIFT (with ground truth detections) for all five datasets: Three frames of a video are displayed. Each detected track is colored differently (the color does not correspond to the object class). The predicted action class is written above the track and class prediction. If there is no action that means no prediction was possible. To make it easier to read, we have noted the action classes of the individual animals again under the frames.

| Dataset | Model | Top-1 (without mask) | Top-5 (without mask) | Top-1 (with mask) | Top-5 (with mask) |
|---|---|---|---|---|---|
| Rolandseck Daylight | MAROON | 54.65 | 90.96 | **72.24** | **95.88** |
| | SlowFast | 42.05 | 89.66 | 57.67 | 92.48 |
| | MViT | 43.13 | 85.18 | 49.41 | 89.57 |
| Rolandseck Nighttime | MAROON | 59.26 | 94.59 | **82.33** | **97.15** |
| | SlowFast | 35.06 | 91.46 | 48.71 | 95.44 |
| | MViT | 41.61 | 91.48 | 66.37 | 93.74 |
| Bavarian Forest Daylight | MAROON | 36.44 | 94.85 | **54.27** | **97.59** |
| | SlowFast | 35.40 | 95.88 | 45.35 | 96.57 |
| | MViT | 35.05 | 94.17 | 42.63 | 95.54 |
| Bavarian Forest Nighttime | MAROON | 35.04 | 91.42 | **45.47** | **95.08** |
| | SlowFast | 35.49 | 95.68 | 42.93 | 93.86 |
| | MViT | 31.91 | 93.26 | 40.62 | 92.03 |
| Wildlife Crossings | MAROON | 58.77 | 97.60 | **73.11** | **98.40** |
| | SlowFast | 59.51 | 99.20 | 63.54 | 97.60 |
| | MViT | 53.97 | 96.03 | 68.31 | 97.60 |

**Table 5.2:** Analysis of the impact of masked input frames on the action recognition accuracy. The best value for each dataset is marked blue.

our analysis encompasses various configurations. We explore four distinct scenarios: a connection from the fast to the slow pathway (which is ultimately selected for MAROON), a connection between the medium and slow pathways (both before and after integration with the fast pathway), and the absence of any lateral connections. The outcomes of these investigations are detailed in Table 5.3.

For all datasets the connection between the fast and slow pathway is the most successful choice. This configuration is for all datasets better by 3 to 4 percentage points than the second-best alternative. This choice also corresponds to our theoretical considerations in Section 5.3. However, the second-best choice varies for the different datasets.

| Dataset | Type of lateral connection | Top-1 | Top-5 |
|---|---|---|---|
| Rolandseck Daylight | fast to slow | **72.24** | 95.88 |
| | no connection | 65.50 | 95.70 |
| | medium (before merging) to slow | 66.74 | 94.32 |
| | medium (after merging) to slow | 68.26 | **96.00** |
| Rolandseck Nighttime | fast to slow | **82.33** | 97.15 |
| | no connection | 76.10 | **97.72** |
| | medium (before merging) to slow | 69.78 | 94.86 |
| | medium (after merging) to slow | 77.49 | 97.15 |
| Bavarian Forest Daylight | fast to slow | **54.27** | 97.59 |
| | no connection | 45.71 | **97.94** |
| | medium (before merging) to slow | 51.89 | 96.21 |
| | medium (after merging) to slow | 46.38 | 96.92 |
| Bavarian Forest Nighttime | fast to slow | **45.47** | 95.08 |
| | no connection | 42.35 | **96.93** |
| | medium (before merging) to slow | 39.18 | 95.10 |
| | medium (after merging) to slow | 41.70 | 95.09 |
| Wildlife Crossings | fast to slow | **73.11** | **98.40** |
| | no connection | 69.91 | 96.83 |
| | medium (before merging) to slow | 62.50 | 97.60 |
| | medium (after merging) to slow | 63.57 | 96.83 |

**Table 5.3:** Analysis of the lateral connections to the new MAROON Slow pathway. The best value for each dataset is marked blue.

### 5.4.5 Oversampling Analysis

In this section we analyze the degree of oversampling. As previously described, all datasets form long-tailed distributions in which some classes occur significantly more frequently than others. If training is now performed without oversampling, the data

samples are presented per epoch according to their distribution in the dataset. This means that rare classes are so rarely presented as training examples that they cannot be learned by the network or are ignored due to their rarity. Oversampling artificially increases the number of rare classes in the dataset. The oversampling threshold increases the samples of the small classes to such an extent that they contain at least $X\%$ of the largest class. That means that an oversampling threshold of 100% leads to classes of equal size. The new samples are randomly drawn and duplicated from the set of existing samples of the respective class and then augmented using the data augmentation techniques already described, such as horizontal flip, spatial and temporal jittering.

In Table 5.4 we present the analysis of the oversampling threshold for our five considered datasets. In all datasets except the Wildlife Crossings dataset, an oversampling threshold of 50% turns out to be the best choice. This is also a good choice from a theoretical point of view. With a lower value, the small classes still occur too rarely to be learned correctly by the network. However, if the oversampling threshold is set too high, this can lead to overfitting on the duplicated classes (even if these are varied by data augmentation). For the Rolandseck Daylight dataset, for example, an oversampling threshold of 100% would mean that the rarest class, standing up, with 5 samples would be expanded to the size of the largest class, vigilant standing, with 149 samples. For the Wildlife Crossings dataset, an oversampling threshold of 100% is the best choice. This can be explained by the fact that this dataset is the smallest dataset and therefore has the smallest number of samples and also contains the most different animal classes. This leads to a continued high diversity of the data even when oversampling to 100%.

### 5.4.6 Analysis of the Pathway Parameters

Our model MAROON introduces 2 new parameters, $\gamma$ and $\delta$ for our new third pathway. As already described in detail in Section 5.3, the parameter $\delta$ steers the amount of frames that are fed into the MAROON slow pathway. The parameter $\alpha$ from the original SlowFast architecture decides how many frames are fed into the fast pathway. The choice of the two parameters is limited by the given total number of frames of the sequences. In our case, this is 16 frames for the Rolandseck datasets and 8 frames for the other three datasets. With $\alpha = 4$, this results in the MAROON fast pathway receiving 16 or 8 frames depending on the dataset under consideration and the MAROON medium pathway receiving 4 or 2 frames respectively. This ratio also corresponds to the authors' choice of SlowFast (Feichtenhofer et al., 2019), as this proved to be the best choice in their experiments. The parameter $\beta$ controls the ratio of the feature channels between the MAROON medium and the MAROON fast pathway. This parameter has also already been introduced for SlowFast. The authors (Feichtenhofer et al., 2019) have shown that a value of $\beta = 1/8$ was the most successful. Our new parameter $\gamma$ controls the ratio of the channels of the

| Dataset | Oversampling threshold | Top-1 | Top-5 |
|---|---|---|---|
| Rolandseck Daylight | 0% | 69.95 | 95.25 |
| | 25% | 71.01 | 96.01 |
| | 50% | **72.24** | 95.88 |
| | 75% | 69.16 | **96.31** |
| | 100% | 68.39 | 94.94 |
| Rolandseck Nighttime | 0% | 77.19 | 98.00 |
| | 25% | 75.47 | **98.57** |
| | 50% | **82.33** | 97.15 |
| | 75% | 80.35 | 97.43 |
| | 100% | 80.90 | 98.01 |
| Bavarian Forest Daylight | 0% | 50.85 | 96.56 |
| | 25% | 47.45 | 96.92 |
| | 50% | **54.27** | **97.59** |
| | 75% | 46.39 | 97.24 |
| | 100% | 42.26 | 96.56 |
| Bavarian Forest Nighttime | 0% | 40.26 | 94.93 |
| | 25% | 41.13 | 93.22 |
| | 50% | **45.47** | 95.08 |
| | 75% | 43.05 | **96.33** |
| | 100% | 42.87 | 95.70 |
| Wildlife Crossings | 0% | 59.20 | 92.00 |
| | 25% | 60.34 | **99.20** |
| | 50% | 61.14 | 98.40 |
| | 75% | 63.54 | 97.60 |
| | 100% | **73.11** | 98.40 |

**Table 5.4:** Analysis of oversampling threshold for the MAROON model. Oversampling with 0% means that no oversampling takes place. At 100%, each class is represented equally frequently. The best value for each dataset is marked blue.

MAROON slow pathway to the MAROON medium pathway. The slow pathway has more feature channels than the medium pathway. The value of $\gamma$ is also limited by the available GPU memory capacity, as more feature channels also require correspondingly more memory.

In Table 5.5 we present our extensive experimental results for all five datasets. The parameter combination of $\gamma = 4$ and $\delta = 8$ is the best combination for all datasets. The parameter $\delta = 8$ in combination with $\alpha = 4$ results in the number of frames that are fed into the fast, medium and slow pathway as [16,4,2] for Rolandseck datasets and [8,2,1] for the other three datasets. The parameter $\gamma = 4$ and $\beta = 1/8$ result in feature channel capacities for the fast, medium and slow pathway as $[C/8, C, 4C]$ for all datasets.

## 5.5 Discussion

Our evaluation results indicate that MAROON outperforms other state-of-the-art models across various datasets. Each of our five datasets encompasses distinct scenarios and settings. The Rolandseck Daylight and Nighttime datasets, for example, often feature groups of animals, making the use of instance masks critical for accurately distinguishing individual animals and ensuring reliable action predictions. The Bavarian Forest Daylight dataset, on the other hand, includes videos with fewer animals, and is characterized by lower resolution and frame rates. The Bavarian Forest Nighttime dataset demonstrates MAROON's capability to predict actions in videos lacking color information and with very low frame rates. The Wildlife Crossings dataset shows not only videos with a low resolution and frame rate, but also different animal species like deer and boar. A straightforward method to enhance the accuracy across all models would be to increase the volume of video data, thus expanding the training and testing material. However, the process of annotating these videos is labor-intensive, and rarer action classes tend to be underrepresented, reflecting a common issue in wildlife ecology (Sakib and Burghardt, 2020) and the broader field of computer vision where action classes often exhibit long-tailed distributions (Zhang et al., 2023). Enriching these rare action classes with additional examples would improve model accuracy, but this is frequently constrained not just by annotation efforts but also by the scarcity of camera trap recordings of these actions. Such actions are typically of particular interest to researchers.

To address the imbalance in data representation, we employ oversampling, a widely used technique for more effective training with long-tailed distributions. Due to the very limited instances of some actions, undersampling is impractical as it would overly diminish the diversity of data for more frequent classes. To introduce more variety during training, we implement data augmentation techniques previously discussed in Section 5.4.1, such as horizontal flips, temporal jittering, and spatial jittering.

| Dataset | $\gamma$ | $\delta$ | Top-1 | Top-5 |
|---|---|---|---|---|
| Rolandseck Daylight | 2 | 4 | 68.41 | 95.55 |
| | 4 | 4 | 69.95 | 95.56 |
| | 8 | 4 | 60.73 | 92.49 |
| | 2 | 8 | 69.50 | 95.40 |
| | 4 | 8 | **72.24** | **95.88** |
| | 8 | 8 | 45.42 | 80.68 |
| | 2 | 16 | 69.18 | 95.25 |
| | 4 | 16 | 69.33 | 95.40 |
| | 8 | 16 | 48.46 | 84.81 |
| Rolandseck Nighttime | 2 | 4 | 81.20 | **98.01** |
| | 4 | 4 | 74.91 | 96.87 |
| | 8 | 4 | 74.61 | 97.44 |
| | 2 | 8 | 79.20 | 97.72 |
| | 4 | 8 | **82.33** | 97.15 |
| | 8 | 8 | 76.38 | 97.72 |
| | 2 | 16 | 79.17 | 97.15 |
| | 4 | 16 | 80.33 | 97.72 |
| | 8 | 16 | 78.06 | 97.72 |
| Bavarian Forest Daylight | 2 | 4 | 48.77 | 95.53 |
| | 4 | 4 | 53.27 | 95.53 |
| | 8 | 4 | 51.89 | 96.56 |
| | 2 | 8 | 51.17 | 96.91 |
| | 4 | 8 | **54.27** | **97.59** |
| | 8 | 8 | 54.25 | 97.25 |
| Bavarian Forest Nighttime | 2 | 4 | 36.19 | **96.93** |
| | 4 | 4 | 45.45 | 94.47 |
| | 8 | 4 | 44.83 | 93.88 |
| | 2 | 8 | 43.63 | 93.83 |
| | 4 | 8 | **45.47** | 95.08 |
| | 8 | 8 | 41.17 | 93.84 |
| Wildlife Crossings | 2 | 4 | 70.68 | 98.43 |
| | 4 | 4 | 63.54 | 99.20 |
| | 8 | 4 | 67.60 | 98.43 |
| | 2 | 8 | 71.57 | 98.40 |
| | 4 | 8 | **73.11** | 98.40 |
| | 8 | 8 | 69.08 | **99.23** |

**Table 5.5:** Analysis of the pathway parameters $\gamma$ and $\delta$ of MAROON. The best value for each dataset is marked blue.

If ample time and resources are allocated to a study, it's feasible to produce additional video recordings in zoos or wildlife enclosures to gather data on rare behaviors. Alternatively, augmenting the dataset with artificially generated data is another viable option. Although today's technology can produce high-quality artificial images, there remain challenges in transferring a network trained on these images to real-world scenarios, particularly for complex scenes like wildlife videos that feature varying weather conditions, exposure levels, and day-night cycles. Nonetheless, with continued improvements in the generation of artificial data, this method holds significant potential for enhancing the dataset in future studies.

Our action recognition network is currently designed to identify actions performed by individual animals, meaning it cannot detect interactions between multiple animals. In our dataset, instances involving animal interactions were extremely rare, thus were not a focus of our study. Typically, most action recognition models are developed to recognize the actions of a single actor. However, in other studies and datasets, particularly those involving different species, recognizing interactions between animals could be of interest. Existing action detection models, such as those documented in studies (Tang et al., 2020; Biswas and Gall, 2020), have explored action recognition among humans. To extend our model to capture animal interactions, it would require modifications to accept multiple animals masked as inputs and processed through the action recognition network, allowing for the detection of interactions between them.

Adapting our action detection system for use with other animal species, particularly those of similar size, is generally straightforward. Initially, it requires gathering and annotating data specific to the desired animal species and their actions. SWIFT and MAROON would then need to be retrained to recognize these new species and action classes. Given the robustness and reliability of the Mask R-CNN in SWIFT, which can detect a wide range of objects, transferring the system to new contexts is typically feasible. However, challenges may arise when dealing with smaller animals, such as birds, which might only occupy a very small area of the video frame (e.g., less than 50 x 50 pixels). Detection in such cases can be challenging, and action recognition may be further complicated by the lack of visual detail. Therefore, careful consideration of the camera setup is crucial to ensure that the recordings capture adequate detail for effective detection and action recognition.

The resolution of video recordings is crucial for effectively recognizing animals and their actions. Higher resolution is beneficial for instance segmentation with SWIFT, as it allows for the segmentation of more object details when the animal is closer to the camera and occupies more pixels. Consequently, higher image resolution facilitates easier recognition of action classes in the action recognition phase. However, the capability to recognize actions also depends on the specific action classes involved — namely, the level of detail required to distinguish the action and the potential for similar actions to be confused with one another. When considering the application of our system to other animal species, the video resolution must be

carefully considered. For smaller animals or birds, for instance, the existing camera settings might render the animals too small to accurately differentiate between various actions. Recordings from the Rolandseck dataset are already at a very high resolution, and enhancing video quality further can be challenging due to storage limitations on camera trap devices. Therefore, it is important to tailor recording settings to the specific needs of the targeted animal and action classes. Adjustments might include the proximity of the camera to the subject or the specific location of the recording, ensuring that the setup is optimized to capture the necessary detail for effective action recognition.

Building on the concept of SlowFast (Feichtenhofer et al., 2019) and incorporating three pathways into MAROON has markedly enhanced the accuracy of action recognition. However, this expansion of the model introduces more parameters compared to SlowFast, resulting in increased training times. Consequently, when working with very large datasets, it may be necessary to allocate more time for training or to employ additional hardware resources to manage the extended training process efficiently.

Introducing masked input sequences into our system has enhanced action prediction accuracy, with an average improvement of 10 percentage points. However, this approach requires that instance masks be available for the different animals. Typically, action recognition datasets focus solely on video sequences of the actor, while action detection datasets, such as the AVA dataset (Gu et al., 2018) for person action detection, often employ only bounding boxes. If no existing dataset with instance masks for the desired object type is available, new annotations must be created, which is generally more labor-intensive than creating bounding box annotations. Despite the challenges, the task of annotation is becoming more feasible with ongoing advancements in AI-assisted annotation tools (Sofiiuk et al., 2022; Liu et al., 2023a) and the development of foundation models like Segment Anything (Kirillov et al., 2023). These technologies are likely to simplify the process of generating instance masks in the future, making it less cumbersome than it currently is.

In Section 5.4.2, we detail the class-specific action recognition accuracies for our models across various datasets. There are two primary reasons why both our MAROON approach and the other two action recognition networks achieve lower accuracies for some classes compared to others. Firstly, as previously discussed, some classes such as resting, standing up, or sudden rush occur less frequently than others. Although oversampling partially mitigates this issue, there might still be insufficient diversity in the training data to enable successful generalization. Actively expanding the dataset with more examples of these less frequent behaviors, provided sufficient recordings are available, could enhance prediction accuracy. The second reason for the difficulty in predicting certain actions stems from variability in how they are performed and their similarity to other actions. For instance, the class foraging moving is distinguished from walking primarily because the animal is searching for food, typically with its head lowered. Similarly, foraging standing and vigilant standing

can appear alike, especially when the animal is searching for food near a bush without lowering its head. Increasing the number of samples for these nuanced classes could aid in better distinguishing between them. Nonetheless, it is observed that the action recognition accuracies for these more nuanced classes are generally higher than those for the classes affected by the first issue.

When applying an action detection system to videos where animals change their behavior — a common scenario — there are transition areas between different behaviors where the prediction shifts from one behavior to another. Insights from the analysis of behavioral sequences in ethology, as detailed in studies like (Chatfield and Lemon, 1970; Bels et al., 2022; Gygax et al., 2022) , could be valuable for enhancing the action prediction process in the future. These principles are also applicable to human behavior studies, as suggested by (Keatley, 2018), where certain behaviors are more likely to precede others. While these insights currently do not impact the evaluation of action recognition approaches — since evaluations typically focus on sequences showcasing a single behavior — they could be immensely useful for applying the system to new, unseen recordings. Incorporating this additional information could potentially increase the accuracy of action predictions or help validate the results of action recognition systems.

## 5.6 Conclusions

We presented an approach for action detection of wildlife in camera trap videos, which consists of the combination of our instance segmentation and tracking method SWIFT and our action recognition network MAROON. By exploiting the instance masks generated by SWIFT, MAROON can obtain masked input sequences and predict actions more accurately by focusing on the actor. In our ablation experiments, the great influence of the instance masks was particularly evident, as the action recognition accuracy improved enormously for all models and all datasets using the masked input sequences. The triple-stream approach allows a finer extraction of motion and appearance features than it is possible with the two-stream approach of SlowFast, for example. In our experiments on the 5 different datasets, we were able to show that MAROON beats the other state-of-the-art approaches in top-1 accuracy. In the ablation experiments, we also evaluated the individual aspects of MAROON.

# Chapter 6

# Conclusion

In this thesis, we presented the first instance segmentation, tracking and action detection system for the task of wildlife monitoring. Our system consists of two main parts: the detection and tracking part that we solve with SWIFT and the action recognition part that is done by our action recognition network MAROON. Our most important research result is that the use of instance masks enables a major improvement in multi-object tracking and action recognition.

Our novel MOTS (multi-object tracking and segmentation) pipeline, SWIFT, generates accurate instance masks combining a reliable instance segmentation network with a refinement algorithm. To form reliable tracks from the instance segmentation detections we introduce a tracking approach consisting of multiple filtering steps. Our system is the first approach, to the best of our knowledge, to track animals in wildlife monitoring videos with instance masks. The robust Mask R-CNN is able to detect the animals in the challenging and changing weather and recording conditions. By combining a robust instance segmentation approach such as Mask R-CNN with a refinement network, we solve the problem of smaller datasets in wildlife monitoring compared to general instance segmentation datasets. In this way, we can generate exact instance masks, even if the amount of training data is limited. Due to the modularity of our system, the network trained on animals benefits also from the further development of foundation models.

SWIFT improves the accuracy of instance masks and tracking compared to popular methods from the context of general object detection and person tracking that we trained on each of the five wildlife datasets. We achieve an average precision for the instance masks for the Rolandseck Daylight dataset of 0.485 compared to 0.435 of the Mask R-CNN baseline model. On average the instance mask quality for the different datasets is improved by 4 percentage points. Furthermore, SWIFT achieves a tracking accuracy of 84.6% for the Bavarian Forest Daylight dataset and achieves a sMOTSA score of 42.6% for its joint segmentation and tracking ability on the Rolandseck Daylight dataset. These findings demonstrate that the sequential deletion and combination of tracklets, guided by a particle filter motion

model and a Re-Identification (Re-ID) network, can effectively track animals in challenging conditions. The more accurate instance masks and improved tracking capability lead to longer and more accurate tracks of the animals. This leads to more accurate population estimations due to better detection capabilities and also forms an essential basis for the subsequent action recognition in our action detection system.

To perform action detection on camera trap videos, we combine SWIFT as the detector and tracker with our novel action recognition network MAROON. Our approach MAROON is the first action recognition approach that combines instance mask information for the input sequences with a feature extraction in a triple-stream approach for action recognition. By using the instance masks to mask the animals, actions of animals that are close to other animals can be better recognized, as only the actor is focused on by the network. In addition, cutting out the animals enables better generalization, as it removes the influence of the changing background on action recognition. In our experiments we have shown that MAROON improves the action recognition accuracy on all five datasets compared to other state-of-the-art computer vision approaches. For the dataset Rolandseck Daylight, MAROON achieves a top-1 accuracy of 72.24% in comparison to 43.13% from MViT and 42.05% from SlowFast. Also on the other datasets, our action recognition network achieves on average a 10 percent point higher top-1 accuracy compared to the other approaches. We perform a 5-fold cross validation to show that our approach generally works better than the other approaches.

In our work (Schindler et al., 2024), we have collaborated with ecologists and behavioral scientists from the University of Freiburg and the Bavarian Forest National Park to adapt our action detection system to the objectives of the application domain. Our systems will therefore significantly reduce the manual work of researchers and help them perform more complex behavioral analyses. We are therefore confident that SWIFT and MAROON will also provide important support for other ecological objectives and enable new insights to be gained in ecology.

## 6.1 Future Work

We have successfully applied our action detection system to various deer species. It would be interesting to apply our action detection system to datasets with other mammals that have been annotated accordingly. For a successful application, SWIFT would have to be retrained for the detection. It would be interesting to see how far action detection with MAROON can be transferred to different animal species. In this work, we looked at the actions of similar species, in particular red deer, roe deer and fallow deer. This could raise the question of whether the detection of general actions such as walking, vigilant standing or lying can be transferred from these animal species to other species such as lions, zebras or alpacas.

One focus of future research will be the detection of the animals. The classic approach of training and fine-tuning a detection model to a specific dataset may become obsolete if foundation models continue to develop as they have in recent years. We have already investigated this by integrating the Segment Anything Model into our refinement algorithm. It is possible that detection with instance masks may be completely replaced by an even more improved foundation model in the future. This could mean that further refinement is no longer necessary and a single network can take over this task. This would also significantly reduce the annotation work required for a transfer to previously unknown animal species.

In our SWIFT Tracking Algorithm, we use a Re-ID network to combine tracklets with other tracklets, where animals have been temporarily occluded by bushes or trees, for example, or animals have completely left and re-entered the scene. For the training of the Re-ID network, we had to limit ourselves to the recognition of animals within a video, as no ground truth exists for a cross-video Re-ID for the different deer species. It would therefore be interesting to see whether such a ground truth can be created in the future and whether a Re-ID network that has been trained on it can merge tracklets even better. An improvement could be expected, particularly for the cases, where an animal leaves the scene completely and then reappears.

Our action detection system can form the basis for further computer vision tasks in the context of wildlife monitoring in the future. For example, the re-identification of individual animal instances could be based on our detections and action predictions. The extracted appearance and motion features from the various steps of our system could be used accordingly in a re-identification network that uses video information. This would certainly be easier to implement on animals with special characteristics, for example, special individual fur patterns such as zebras or giraffes. In addition, a pose estimation of the animals could also be carried out based on the instance masks found. The keypoints found by the pose estimation could be helpful in better distinguishing complex actions. In order to perform a pose estimation, the data would first have to be annotated with the corresponding keypoints.

# List of Figures

# List of Tables

# Acronyms

The most important acronyms used in this thesis are listed here in alphabetical order:

| | |
|---|---|
| AP | Average Precision |
| AR | Average Recall |
| CNN | Convolutional Neural Network |
| COCO | Common Objects in COntext |
| f-BRS | feature-Backpropagating Refinement Scheme |
| FN | False Negative |
| FP | False Positive |
| FPS | Frames Per Second |
| GT | Ground Truth |
| HR-Net | High-Resolution Net |
| IDSW | ID SWitches |
| IoU | Intersection over Union |
| LVIS | Large Vocabulary Instance Segmentation (dataset) |
| mAP | mean Average Precision |
| mAR | mean Average Recall |
| MAROON | Mask guided Action RecOgnitiON |

| | |
|---|---|
| ML | Mostly Lost |
| MOT | Multi-Object Tracking |
| MOTA | Multi-Object Tracking Accuracy |
| MOTP | Multi-Object Tracking Precision |
| MOTS | Multi-Object Tracking and Segmentation |
| MOTSA | Multi-Object Tracking and Segmentation Accuracy |
| MOTSP | Multi-Object Tracking and Segmentation Precision |
| MT | Mostly Tracked |
| MViT | Multiscale Vision Transformer |
| OS-Net | Omni-Scale Network |
| PIR | Pyrocelectric InfraRed |
| PT | Partially Tracked |
| RGB | Red, Green and Blue (image) |
| RITM | Reviving Iterative Training with Mask guidance for interactive segmentation |
| SAM | Segment Anything Model |
| sMOTSA | soft Multi-Object Tracking and Segmentation Accuracy |
| SORT | Simple Online and Realtime Tracking |
| SWIFT | Segmentation WIth Filtering of Tracklets |
| TP | True Positive |

# Bibliography


Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. Bot-sort: Robust associations multi-pedestrian tracking. *arXiv preprint arXiv:2206.14651*, 2022.

Martin Ahrnbom, Mikael G Nilsson, and Håkan Ardö. Real-time and online segmentation multi-target tracking with track revival re-identification. In *VISIGRAPP (5: VISAPP)*, pages 777–784, 2021.

Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6836–6846, 2021a.

Anurag Arnab, Chen Sun, and Cordelia Schmid. Unified graph structured models for video understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8117–8126, 2021b.

Anurag Arnab, Xuehan Xiong, Alexey Gritsenko, Rob Romijnders, Josip Djolonga, Mostafa Dehghani, Chen Sun, Mario Lučić, and Cordelia Schmid. Beyond transfer learning: Co-finetuning for action localisation. *arXiv preprint arXiv:2207.03807*, 2022.

Ali Athar, Sabarinath Mahadevan, Aljosa Osep, Laura Leal-Taixé, and Bastian Leibe. Stem-seg: Spatio-temporal embeddings for instance segmentation in videos. In *European Conference on Computer Vision*, pages 158–177. Springer, 2020.

Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European conference on computer vision (ECCV)*, pages 456–473, 2018.

Sara Beery, Dan Morris, and Siyu Yang. Efficient pipeline for camera trap image review. *arXiv preprint arXiv:1907.06772*, 2019.

Rotimi-Williams Bello, Ahmad Sufril Azlan Mohamed, and Abdullah Zawawi Talib. Contour extraction of individual cattle from an image using enhanced mask r-cnn instance segmentation method. *IEEE Access*, 9:56984–57000, 2021.

Vincent L Bels, Jean-Pierre Pallandre, Eric Pelle, and Florence Kirchhoff. Studies of the behavioral sequences: The neuroethological morphology concept crossing ethology and functional morphology. *Animals*, 12(11):1336, 2022.

Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE transactions on pattern analysis and machine intelligence*, 33(9):1806–1819, 2011.

Oded Berger-Tal, Tal Polak, Aya Oron, Yael Lubin, Burt P Kotler, and David Saltz. Integrating animal behavior and conservation biology: a conceptual framework. *Behavioral Ecology*, 22(2):236–239, 2011.

Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 941–951, 2019.

Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.

Gedas Bertasius and Lorenzo Torresani. Classifying, segmenting, and tracking object instances in video with mask propagation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9739–9748, 2020.

Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.

Amlaan Bhoi. Spatio-temporal action recognition: A survey. *arXiv preprint arXiv:1901.09403*, 2019.

Sovan Biswas and Juergen Gall. Discovering multi-label actor-action association in a weakly supervised setting. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

Luigi Boitani. *Camera trapping for wildlife research*. Pelagic Publishing Ltd, 2016.

Mathieu Bonneau, Jehan-Antoine Vayssade, Willy Troupe, and Rémy Arquet. Outdoor animal tracking combining neural network and time-lapse cameras. *Computers and Electronics in Agriculture*, 168:105150, 2020.

Guillem Brasó and Laura Leal-Taixé. Learning a neural solver for multiple object tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6247–6257, 2020.

Otto Brookes, Majid Mirmehdi, Hjalmar Kühl, and Tilo Burghardt. Triple-stream deep metric learning of great ape behavioural actions. *arXiv preprint arXiv:2301.02642*, 2023.

A Cole Burton, Eric Neilson, Dario Moreira, Andrew Ladle, Robin Steenweg, Jason T Fisher, Erin Bayne, and Stan Boutin. Wildlife camera trapping: a review and recommendations for linking surveys to ecological processes. *Journal of Applied Ecology*, 52(3):675–685, 2015.

Jiarui Cai, Mingze Xu, Wei Li, Yuanjun Xiong, Wei Xia, Zhuowen Tu, and Stefano Soatto. Memot: Multi-object tracking with memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8090–8100, 2022.

Jintong Cai and Yujie Li. Realtime single-stage instance segmentation network based on anchors. *Computers and Electrical Engineering*, 95:107464, 2021.

Jiale Cao, Rao Muhammad Anwer, Hisham Cholakkal, Fahad Shahbaz Khan, Yanwei Pang, and Ling Shao. Sipmask: Spatial information preservation for fast image and video instance segmentation. *arXiv preprint arXiv:2007.14772*, 2020.

Jinkun Cao, Jiangmiao Pang, Xinshuo Weng, Rawal Khirodkar, and Kris Kitani. Observation-centric sort: Rethinking sort for robust multi-object tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9686–9696, 2023.

Anthony Caravaggi, Marco Zaccaroni, Francesco Riga, Stéphanie C Schai-Braun, Jaimie TA Dick, W Ian Montgomery, and Neil Reid. An invasive-native mammalian species replacement process captured by camera trap survey random encounter models. *Remote Sensing in Ecology and Conservation*, 2(1):45–58, 2016.

Anthony Caravaggi, Peter B Banks, A Cole Burton, Caroline MV Finlay, Peter M Haswell, Matt W Hayward, Marcus J Rowcliffe, and Mike D Wood. A review of camera trapping for conservation behaviour research. *Remote Sensing in Ecology and Conservation*, 3(3):109–122, 2017.

Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

Christopher Chatfield and Robert E Lemon. Analysing sequences of behavioural events. *Journal of Theoretical Biology*, 29(3):427–445, 1970.

Guobin Chen, Tony X Han, Zhihai He, Roland Kays, and Tavis Forrester. Deep convolutional neural network based species recognition for wild animal monitoring. In *2014 IEEE international conference on image processing (ICIP)*, pages 858–862. IEEE, 2014.

Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4974–4983, 2019a.

Lei Chen, Zhan Tong, Yibing Song, Gangshan Wu, and Limin Wang. Efficient video action detection with token dropout and context refinement. *arXiv preprint arXiv:2304.08451*, 2023.

Longtao Chen, Xiaojiang Peng, and Mingwu Ren. Recurrent metric networks and batch multiple hypothesis for multi-object tracking. *IEEE Access*, 7:3093–3105, 2018a.

Minghao Chen, Fangyun Wei, Chong Li, and Deng Cai. Frame-wise action representations for long videos via sequence contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13801–13810, 2022.

Ruilong Chen, Ruth Little, Lyudmila Mihaylova, Richard Delahay, and Ruth Cox. Wildlife surveillance using deep learning methods. *Ecology and Evolution*, 9 (17):9453–9466, 2019b. doi: 10.1002/ece3.5410. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/ece3.5410`.

Shoufa Chen, Peize Sun, Enze Xie, Chongjian Ge, Jiannan Wu, Lan Ma, Jiajun Shen, and Ping Luo. Watch only once: An end-to-end video action detection framework. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8178–8187, 2021.

Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollar. Tensormask: A foundation for dense object segmentation. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019c.

Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Domain adaptive faster r-cnn for object detection in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3339–3348, 2018b.

Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. Modular interactive video object segmentation: Interaction-to-mask, propagation and difference-aware fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5559–5568, 2021a.

Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. Rethinking space-time networks with improved memory coverage for efficient video object segmentation. *Advances in Neural Information Processing Systems*, 34, 2021b.

R Christoph and Feichtenhofer Axel Pinz. Spatiotemporal residual networks for video action recognition. *Advances in neural information processing systems*, 2: 3468—-3476, 2016.

Peng Chu, Heng Fan, Chiu C Tan, and Haibin Ling. Online multi-object tracking with instance-aware tracker and dynamic model refreshment. In *2019 IEEE winter conference on applications of computer vision (WACV)*, pages 161–170. IEEE, 2019.

Peng Chu, Jiang Wang, Quanzeng You, Haibin Ling, and Zicheng Liu. Transmot: Spatial-temporal graph transformer for multiple object tracking. *arXiv preprint arXiv:2104.00194*, 2021.

Gioele Ciaparrone, Francisco Luque Sánchez, Siham Tabik, Luigi Troiano, Roberto Tagliaferri, and Francisco Herrera. Deep learning in video multi-object tracking: A survey. *Neurocomputing*, 381:61–88, 2020.

Sharana Dharshikgan Suresh Dass, Hrishav Bakul Barua, Ganesh Krishnasamy, Raveendran Paramesran, and Raphael C-W Phan. Actnetformer: Transformer-resnet hybrid method for semi-supervised action recognition in videos. *arXiv preprint arXiv:2404.06243*, 2024.

Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. Mot20: A benchmark for multi object tracking in crowded scenes. *arXiv preprint arXiv:2003.09003*, 2020.

Caglayan Dicle, Octavia I Camps, and Mario Sznaier. The way they move: Tracking multiple targets with similar appearance. In *Proceedings of the IEEE international conference on computer vision*, pages 2304–2311, 2013.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Yunhao Du, Zhicheng Zhao, Yang Song, Yanyun Zhao, Fei Su, Tao Gong, and Hongying Meng. Strongsort: Make deepsort great again. *IEEE Transactions on Multimedia*, 2023.

Abhishek Dutta and Andrew Zisserman. The VIA annotation software for images, audio and video. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6889-6/19/10. doi: 10.1145/3343031.3350535. URL `https://doi.org/10.1145/3343031.3350535`.

Mark Everingham, Luc Van Gool, Christopher K.I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. In *Int J Comput Vis*, volume 88, page 303–338, 2010. doi: https://doi.org/10.1007/s11263-009-0275-4.

Greg Falzon, Christopher Lawson, Ka-Wai Cheung, Karl Vernes, Guy A Ballard, Peter JS Fleming, Alistair S Glen, Heath Milne, Atalya Mather-Zardain, and Paul D Meek. Classifyme: a field-scouting software for the identification of wildlife in camera trap images. *Animals*, 10(1):58, 2020.

Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6824–6835, 2021.

Hao-Shu Fang, Jianhua Sun, Runzhong Wang, Minghao Gou, Yong-Lu Li, and Cewu Lu. Instaboost: Boosting instance segmentation via probability map guided copy-pasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 682–691, 2019.

Gaspar Faure, Hughes Perreault, Guillaume-Alexandre Bilodeau, and Nicolas Saunier. Polytrack: Tracking with bounding polygons. *arXiv preprint arXiv:2111.01606*, 2021.

Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 203–213, 2020.

Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211, 2019.

Sandra Frey, Jason T Fisher, A Cole Burton, and John P Volpe. Investigating animal activity patterns and temporal niche partitioning using camera-trap data: Challenges and opportunities. *Remote Sensing in Ecology and Conservation*, 3 (3):123–132, 2017.

Tsukasa Fukunaga, Shoko Kubota, Shoji Oda, and Wataru Iwasaki. Grouptracker: video tracking system for multiple animals under severe occlusion. *Computational biology and chemistry*, 57:39–45, 2015.

Haiming Gan, Mingqiang Ou, Chengpeng Li, Xiarui Wang, Jingfeng Guo, Axiu Mao, Maria Camila Ceballos, Thomas D Parsons, Kai Liu, and Yueju Xue. Automated detection and analysis of piglet suckling behaviour using high-accuracy amodal instance segmentation. *Computers and Electronics in Agriculture*, 199:107162, 2022.

Naiyu Gao, Yanhu Shan, Yupei Wang, Xin Zhao, Yinan Yu, Ming Yang, and Kaiqi Huang. Ssap: Single-shot instance segmentation with affinity pyramid. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 642–651, 2019.

Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 244–253, 2019.

Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F-radar and signal processing*, volume 140, No.2, pages 107–113. IET, 1993.

Siân E Green, Jonathan P Rees, Philip A Stephens, Russell A Hill, and Anthony J Giordano. Innovations in camera trapping technology and approaches: The integration of citizen science and artificial intelligence. *Animals*, 10(1):132, 2020.

Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6047–6056, 2018.

Wenchao Gu, Shuang Bai, and Lingxing Kong. A review on 2d instance segmentation based on deep neural networks. *Image and Vision Computing*, page 104401, 2022.

Agrim Gupta, Piotr Dollar, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Lorenz Gygax, Yvonne RA Zeeland, and Christina Rufener. Fully flexible analysis of behavioural sequences based on parametric survival models with frailties—a tutorial. *Ethology*, 128(2):183–196, 2022.

Leonard Hacker, Finn Bartels, and Pierre-Etienne Martin. Fine-grained action detection with rgb and pose information using two stream convolutional networks. *arXiv preprint arXiv:2302.02755*, 2023.

Abdul Mueed Hafiz and Ghulam Mohiuddin Bhat. A survey on instance segmentation: state of the art. *International journal of multimedia information retrieval*, 9(3):171–189, 2020.

Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018.

Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

Zhenwei He and Lei Zhang. Multi-adversarial faster-rcnn for unrestricted object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6668–6677, 2019.

Maik Henrich, Mercedes Burgueño, Jacqueline Hoyer, Timm Haucke, Volker Steinhage, Hjalmar S Kühl, and Marco Heurich. A semi-automated camera trap distance sampling approach for population density estimation. *Remote Sensing in Ecology and Conservation*, 2023. doi: https://doi.org/10.1002/rse2.362.

Roberto Henschel, Laura Leal-Taixé, Daniel Cremers, and Bodo Rosenhahn. Fusion of head and full-body detectors for multi-object tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1428–1437, 2018.

Miran Heo, Sukjun Hwang, Seoung Wug Oh, Joon-Young Lee, and Seon Joo Kim. Vita: Video instance segmentation via object token association. *Advances in Neural Information Processing Systems*, 35:23109–23120, 2022.

Miran Heo, Sukjun Hwang, Jeongseok Hyun, Hanjung Kim, Seoung Wug Oh, Joon-Young Lee, and Seon Joo Kim. A generalized framework for video instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14623–14632, 2023.

Shun Hongo, Yoshihiro Nakashima, Gota Yajima, and Shun Hongo. A practical guide for estimating animal density using camera traps: Focus on the rest model. *bioRxiv*, 2021. doi: https://doi.org/10.1101/2021.05.18.444583.

Man Hu and Fucheng You. Research on animal image classification based on transfer learning. In *Proceedings of the 2020 4th International Conference on Electronic Information Technology and Computer Engineering*, pages 756–761, 2020.

Zhiwei Hu, Hua Yang, and Tiantian Lou. Dual attention-guided feature pyramid network for instance segmentation of group pigs. *Computers and Electronics in Agriculture*, 186:106140, 2021.

De-An Huang, Zhiding Yu, and Anima Anandkumar. Minvis: A minimal video instance segmentation framework without video-based training. *Advances in Neural Information Processing Systems*, 35:31265–31277, 2022.

Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6409–6418, 2019.

Boyuan Jiang, MengMeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2000–2009, 2019.

David Keatley. *Pathways in crime: An introduction to behaviour sequence analysis*. Springer, 2018.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

Yu Kong and Yun Fu. Human action recognition and prediction: A survey. *International Journal of Computer Vision*, 130(5):1366–1401, 2022.

Wouter M Kouw and Marco Loog. An introduction to domain adaptation and transfer learning. *arXiv preprint arXiv:1812.11806*, 2018.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Heeseung Kwon, Manjin Kim, Suha Kwak, and Minsu Cho. Motionsqueeze: Neural motion feature learning for video understanding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, pages 345–362. Springer, 2020.

Roger Labbe. Kalman and bayesian filters in python. *Chap*, 7(246):4, 2014.

Trung-Nghia Le, Yubo Cao, Tan-Cong Nguyen, Minh-Quan Le, Khanh-Duy Nguyen, Thanh-Toan Do, Minh-Triet Tran, and Tam V Nguyen. Camouflaged instance segmentation in-the-wild: Dataset, method, and benchmark suite. *IEEE Transactions on Image Processing*, 31:287–300, 2021.

Dongho Lee, Jongseo Lee, and Jinwoo Choi. Cast: Cross-attention in space and time for video action recognition. *Advances in Neural Information Processing Systems*, 36, 2024.

Youngwan Lee and Jongyoul Park. Centermask: Real-time anchor-free instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13906–13915, 2020.

Guofa Li, Xin Chen, Mingjun Li, Wenbo Li, Shen Li, Gang Guo, Huaizhi Wang, and Hao Deng. One-shot multi-object tracking using cnn-based networks with spatial-channel attention mechanism. *Optics & Laser Technology*, 153:108267, 2022a.

Rui Li, Baopeng Zhang, Zhu Teng, and Jianping Fan. An end-to-end identity association network based on geometry refinement for multi-object tracking. *Pattern Recognition*, 129:108738, 2022b.

Wei Li, Yuanjun Xiong, Shuo Yang, Siqi Deng, and Wei Xia. Smot: Single-shot multi object tracking. *arXiv preprint arXiv:2010.16031*, 2020a.

Yan Li, Bin Ji, Xintian Shi, Jianguo Zhang, Bin Kang, and Limin Wang. Tea: Temporal excitation and aggregation for action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 909–918, 2020b.

Shijie Lian, Ziyi Zhang, Hua Li, Wenjie Li, Laurence Tianruo Yang, Sam Kwong, and Runmin Cong. Diving into underwater: Segment anything model guided underwater salient instance segmentation and a large-scale dataset, 2024.

Duohan Liang, Guoliang Fan, Guangfeng Lin, Wanjun Chen, Xiaorong Pan, and Hong Zhu. Three-stream convolutional neural network with multi-task and ensemble learning for 3d action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019.

Chung-Ching Lin, Ying Hung, Rogerio Feris, and Linglin He. Video instance segmentation tracking with a modified vae architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13147–13157, 2020.

Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7083–7093, 2019.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

Matthew Linkie, Yoan Dinata, Agung Nugroho, and Iding Achmad Haidir. Estimating occupancy of a data deficient mammalian species living in tropical rainforests: sun bears in the kerinci seblat region, sumatra. *Biological Conservation*, 137(1): 20–27, 2007.

Dongfang Liu, Yiming Cui, Wenbo Tan, and Yingjie Chen. Sg-net: Spatial granularity network for one-stage video instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9816–9825, 2021.

Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2):261–318, 2020.

Qin Liu, Zhenlin Xu, Gedas Bertasius, and Marc Niethammer. Simpleclick: Interactive image segmentation with simple vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22290–22300, 2023a.

Shu Liu, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. Sgn: Sequential grouping networks for instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3496–3504, 2017.

Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018.

Xiaolong Liu, Song Bai, and Xiang Bai. An empirical study of end-to-end temporal action detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20010–20019, 2022.

Zelin Liu, Xinggang Wang, Cheng Wang, Wenyu Liu, and Xiang Bai. Sparsetrack: Multi-object tracking by performing scene decomposition based on pseudo-depth. *arXiv preprint arXiv:2306.05238*, 2023b.

Yangxiao Lu, Yunhui Guo, Nicholas Ruozzi, Yu Xiang, et al. Adapting pre-trained vision models for novel instance detection and segmentation. *arXiv preprint arXiv:2405.17859*, 2024.

Run Luo, Zikai Song, Lintao Ma, Jinlin Wei, Wei Yang, and Min Yang. Diffusiontrack: Diffusion model for multi-object tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 3991–3999, 2024.

Kefeng Lv, Yongsheng Zhang, Ying Yu, Hanyun Wang, Lei Li, Huaigang Jiang, and Chenguang Dai. Contour deformation network for instance segmentation. *Pattern Recognition Letters*, 159:213–219, 2022.

Liqian Ma, Siyu Tang, Michael J Black, and Luc Van Gool. Customized multi-person tracker. In *Asian conference on computer vision*, pages 612–628. Springer, 2018.

M Marron, JC Garcia, MA Sotelo, M Cabello, D Pizarro, F Huerta, and J Cerro. Comparing a kalman filter and a particle filter in a multiple objects tracking application. In *2007 IEEE International Symposium on Intelligent Signal Processing*, pages 1–6. IEEE, 2007.

Jamie McCallum. Changing use of camera traps in mammalian field research: habitats, taxa and study types. *Mammal Review*, 43(3):196–206, 2013.

Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016.

Veronika Mitterwallner, Anne Peters, Hendrik Edelhoff, Gregor Mathes, Hien Nguyen, Wibke Peters, Marco Heurich, and Manuel J Steinbauer. Automated visitor and wildlife monitoring with camera traps and machine learning. *Remote Sensing in Ecology and Conservation*, 2023. doi: https://doi.org/10.1002/rse2.367.

Xun Long Ng, Kian Eng Ong, Qichen Zheng, Yun Ni, Si Yong Yeo, and Jun Liu. Animal kingdom: A large and diverse dataset for animal behavior understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19023–19034, 2022.

Mohammad Sadegh Norouzzadeh, Anh Nguyen, Margaret Kosmala, Alexandra Swanson, Meredith S. Palmer, Craig Packer, and Jeff Clune. Automatically

identifying, counting, and describing wild animals in camera-trap images with deep learning. *Proceedings of the National Academy of Sciences*, 115(25):E5716–E5725, 2018. ISSN 0027-8424. doi: 10.1073/pnas.1719367115. URL `https://www.pnas.org/content/115/25/E5716`.

E. Okafor, P. Pawara, F. Karaaba, O. Surinta, V. Codreanu, L. Schomaker, and M. Wiering. Comparative study between deep learning and bag of visual words for wild-animal recognition. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, Dec 2016. doi: 10.1109/SSCI.2016.7850111.

Rafael Padilla, Wesley L Passos, Thadeu LB Dias, Sergio L Netto, and Eduardo AB da Silva. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, 10(3):279, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Lorenzo Porzi, Markus Hofinger, Idoia Ruiz, Joan Serrat, Samuel Rota Bulo, and Peter Kontschieder. Learning multi-object tracking and segmentation from automatic annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6846–6855, 2020.

Jiyang Qi, Yan Gao, Xiaoyu Liu, Yao Hu, Xinggang Wang, Xiang Bai, Philip HS Torr, Serge Belongie, Alan Yuille, and Song Bai. Occluded video instance segmentation. *arXiv preprint arXiv:2102.01558*, 2021.

Alvaro Rodriguez, Hanqing Zhang, Jonatan Klaminder, Tomas Brodin, and Magnus Andersson. Toxid: an efficient algorithm to solve occlusions when tracking multiple animals. *Scientific reports*, 7(1):1–8, 2017.

Leonardo Rossi, Akbar Karimi, and Andrea Prati. Self-balanced r-cnn for instance segmentation. *Journal of Visual Communication and Image Representation*, page 103595, 2022.

Michael S Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: What can 8 learned tokens do for images and videos? *arXiv preprint arXiv:2106.11297*, 2021.

Faizaan Sakib and Tilo Burghardt. Visual recognition of great ape behaviours in the wild. *arXiv preprint arXiv:2011.10759*, 2020.

Jennifer Salau and Joachim Krieter. Instance segmentation with mask r-cnn applied to loose-housed dairy cows in a multi-camera setting. *Animals*, 10(12):2402, 2020.

Frank Schindler and Volker Steinhage. Identification of animals and recognition of their actions in wildlife videos using deep learning techniques. *Ecological Informatics*, 61:101215, 2021a.

Frank Schindler and Volker Steinhage. Saving costs for video data annotation in wildlife monitoring. *Ecological Informatics*, 65:101418, 2021b.

Frank Schindler and Volker Steinhage. Instance segmentation and tracking of animals in wildlife videos: Swift-segmentation with filtering of tracklets. *Ecological Informatics*, 71:101794, 2022.

Frank Schindler and Volker Steinhage. Swift - an efficient and effective application of instance segmentation and tracking in wildlife monitoring. In *Workshop Camera Traps, AI and Ecology, Jena*, 2023.

Frank Schindler, Volker Steinhage, Suzanne van Beeck Calkoen, and Marco Heurich. Action detection for wildlife monitoring with camera traps based on segmentation with filtering of tracklets (swift) and mask-guided action recognition (maroon). *Applied Sciences*, 14(2):514, 2024.

Stefan Schneider, Graham W Taylor, Stefan Linquist, and Stefan C Kremer. Past, present and future approaches using computer vision for animal re-identification from camera trap data. *Methods in Ecology and Evolution*, 10(4):461–470, 2019.

Rasha Sheikh and Thomas Schultz. Feature preserving smoothing provides simple and effective data augmentation for medical image segmentation. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part I 23*, pages 116–126. Springer, 2020.

Hao Sheng, Yang Zhang, Jiahui Chen, Zhang Xiong, and Jun Zhang. Heterogeneous association graph fusion for target association in multiple object tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(11):3269–3280, 2018.

Ivaxi Sheth. Three-stream network for enriched action recognition. *arXiv preprint arXiv:2104.13051*, 2021.

Bing Shuai, Andrew Berneshawi, Xinyu Li, Davide Modolo, and Joseph Tighe. Siammot: Siamese multi-object tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12372–12382, 2021.

Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *Advances in neural information processing systems*, 27:568—-576, 2014.

Konstantin Sofiiuk, Ilia Petrov, Olga Barinova, and Anton Konushin. f-brs: Rethinking backpropagating refinement for interactive segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8623–8632, 2020.

Konstantin Sofiiuk, Ilia A Petrov, and Anton Konushin. Reviving iterative training with mask guidance for interactive segmentation. *arXiv preprint arXiv:2102.06583*, 2021.

Konstantin Sofiiuk, Ilya A Petrov, and Anton Konushin. Reviving iterative training with mask guidance for interactive segmentation. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 3141–3145. IEEE, 2022.

Vivek Hari Sridhar, Dominique G Roche, and Simon Gingins. Tracktor: image-based automated tracking of animal movement and behaviour. *Methods in Ecology and Evolution*, 10(6):815–820, 2019.

Karin Stacke, Gabriel Eilertsen, Jonas Unger, and Claes Lundström. A closer look at domain shift for deep learning in histopathology. *arXiv preprint arXiv:1909.11575*, 2019.

Peize Sun, Jinkun Cao, Yi Jiang, Rufeng Zhang, Enze Xie, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack: Multiple object tracking with transformer. *arXiv preprint arXiv:2012.15460*, 2020.

Asif Shahriyar Sushmit, Partho Ghosh, Md Abrar Istiak, Nayeeb Rashid, Ahsan Habib Akash, and Taufiq Hasan. Segcodenet: Color-coded segmentation masks for activity detection from wearable cameras. *arXiv preprint arXiv:2008.08452*, 2020.

Jiajun Tang, Jin Xia, Xinzhi Mu, Bo Pang, and Cewu Lu. Asynchronous interaction aggregation for action detection. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16*, pages 71–87. Springer, 2020.

Aram Ter-Sarkisov, Robert Ross, John Kelleher, Bernadette Earley, and Michael Keane. Beef cattle instance segmentation using fully convolutional neural network. *arXiv preprint arXiv:1807.01972*, 2018.

MW Tobler, SE Carrillo-Percastegui, R Leite Pitman, R Mares, and G Powell. Further notes on the analysis of mammal inventory data collected with camera traps. *Animal Conservation*, 11(3):187–189, 2008.

Pavel Tokmakov, Jie Li, Wolfram Burgard, and Adrien Gaidon. Learning to track with object permanence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10860–10869, 2021.

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

Suzanne TS van Beeck Calkoen, Rebekka Kreikenbohm, Dries PJ Kuijper, and Marco Heurich. Olfactory cues of large carnivores modify red deer behavior and browsing intensity. *Behavioral Ecology*, 32(5):982–992, 2021.

Lisette van der Zande, Oleksiy Guzhva, T Bas Rodenburg, et al. Individual detection and tracking of group housed pigs in their home pen using computer vision. *Frontiers in Animal Science*, 2:10, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30:5998—-6008, 2017.

Juliana Vélez, William McShea, Hila Shamon, Paula J Castiblanco-Camacho, Michael A Tabak, Carl Chalmers, Paul Fergus, and John Fieberg. An evaluation of platforms for processing camera-trap data using artificial intelligence. *Methods in Ecology and Evolution*, 14(2):459–477, 2023.

Gyanendra K. Verma and Pragya Gupta. Wild animal detection using deep convolutional neural network. In Bidyut B. Chaudhuri, Mohan S. Kankanhalli, and Balasubramanian Raman, editors, *Proceedings of 2nd International Conference on Computer Vision & Image Processing*, pages 327–338, Singapore, 2018. Springer Singapore. ISBN 978-981-10-7898-9.

Alexander Gomez Villa, Augusto Salazar, and Francisco Vargas. Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks. *Ecological informatics*, 41:24–32, 2017.

Petra Villette, Charles J Krebs, and Thomas S Jung. Evaluating camera traps as an alternative to live trapping for estimating the density of snowshoe hares (lepus americanus) and red squirrels (tamiasciurus hudsonicus). *European Journal of Wildlife Research*, 63(1):1–9, 2017.

Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. Mots: Multi-object tracking and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7942–7951, 2019.

Gaoang Wang, Yizhou Wang, Haotian Zhang, Renshu Gu, and Jenq-Neng Hwang. Exploit the connectivity: Multi-object tracking with trackletnet. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 482–490, 2019.

Guiqin Wang, Peng Zhao, Yanjiang Shi, Cong Zhao, and Shusen Yang. Generative model-based feature knowledge distillation for action recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 15474–15482, 2024.

Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364, 2020a.

Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks for action recognition in videos. *IEEE transactions on pattern analysis and machine intelligence*, 41(11):2740–2755, 2018a.

Shaoru Wang, Yongchao Gong, Junliang Xing, Lichao Huang, Chang Huang, and Weiming Hu. Rdsnet: A new deep architecture forreciprocal object detection and instance segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, No.7, pages 12208–12215, 2020b.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018b.

Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. Solo: Segmenting objects by locations. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pages 649–665. Springer, 2020c.

Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. *Advances in Neural information processing systems*, 33:17721–17732, 2020d.

Yang Wang, Wanlin Zhou, Qinwei Lv, and Guangle Yao. Metricmask: Single category instance segmentation by metric learning. *Neurocomputing*, 500:896–908, 2022.

Yongxin Wang, Kris Kitani, and Xinshuo Weng. Joint object detection and multi-object tracking with graph neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13708–13715. IEEE, 2021.

Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. *arXiv preprint arXiv:2011.14503*, 2020e.

Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. Towards real-time multi-object tracking. In *European Conference on Computer Vision*, pages 107–122. Springer, 2020f.

Oliver R Wearn and Paul Glover-Kapfer. Snap happy: camera traps are an effective sampling tool when compared with alternative methods. *Royal Society open science*, 6(3):181748, 2019.

Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. *Siggraph Course*, 8:1–16, 2006.

Marco Willi, Ross T. Pitman, Anabelle W. Cardoso, Christina Locke, Alexandra Swanson, Amy Boyer, Marten Veldthuis, and Lucy Fortson. Identifying animal species in camera trap images using deep learning and citizen science. *Methods in Ecology and Evolution*, 10(1):80–91, 2019. doi: 10.1111/2041-210X.13099. URL `https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.13099`.

Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

Jialian Wu, Sudhir Yarram, Hui Liang, Tian Lan, Junsong Yuan, Jayan Eledath, and Gerard Medioni. Efficient video instance segmentation via tracklet query and proposal. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 959–968, 2022a.

Junfeng Wu, Yi Jiang, Wenqing Zhang, Xiang Bai, and Song Bai. Seqformer: a frustratingly simple model for video instance segmentation. *arXiv preprint arXiv:2112.08275*, 2(3):4, 2021.

Junfeng Wu, Qihao Liu, Yi Jiang, Song Bai, Alan Yuille, and Xiang Bai. In defense of online models for video instance segmentation. In *European Conference on Computer Vision*, pages 588–605. Springer, 2022b.

Tao Wu, Mengqi Cao, Ziteng Gao, Gangshan Wu, and Limin Wang. Stmixer: A one-stage sparse action detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14720–14729, 2023.

Wenqiang Xu, Haiyang Wang, Fubo Qi, and Cewu Lu. Explicit shape encoding for real-time instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5168–5177, 2019a.

Yihong Xu, Aljosa Osep, Yutong Ban, Radu Horaud, Laura Leal-Taixé, and Xavier Alameda-Pineda. How to train your deep multi-object tracker. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6787–6796, 2020a.

Yingkun Xu, Xiaolong Zhou, Shengyong Chen, and Fenfen Li. Deep learning for multiple object tracking: a survey. *IET Computer Vision*, 13(4):355–368, 2019b.

Zhenbo Xu, Wei Zhang, Xiao Tan, Wei Yang, Huan Huang, Shilei Wen, Errui Ding, and Liusheng Huang. Segment as points for efficient online multi-object tracking and segmentation. In *European Conference on Computer Vision*, pages 264–281. Springer, 2020b.

Tengfei Xue, Yongliang Qiao, He Kong, Daobilige Su, Shirui Pan, Khalid Rafique, and Salah Sukkarieh. One-shot learning-based animal video segmentation. *IEEE Transactions on Industrial Informatics*, 18:3799–3807, 2021.

Shen Yan, Xuehan Xiong, Anurag Arnab, Zhichao Lu, Mi Zhang, Chen Sun, and Cordelia Schmid. Multiview transformers for video recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3333–3343, 2022.

Fan Yang, Xin Chang, Chenyu Dang, Ziqiang Zheng, Sakriani Sakti, Satoshi Nakamura, and Yang Wu. Remots: Self-supervised refining multi-object tracking and segmentation. *arXiv preprint arXiv:2007.03200*, 2020.

Hanqing Yang, Liyang Zheng, Saba Ghorbani Barzegar, Yu Zhang, and Bin Xu. Borderpointsmask: One-stage instance segmentation with boundary points representation. *Neurocomputing*, 467:348–359, 2022.

Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5188–5197, 2019a.

Shusheng Yang, Yuxin Fang, Xinggang Wang, Yu Li, Chen Fang, Ying Shan, Bin Feng, and Wenyu Liu. Crossover learning for fast online video instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8043–8052, 2021.

Xinyu Yang, Majid Mirmehdi, and Tilo Burghardt. Great ape detection in challenging jungle camera trap footage via attention-based spatial and temporal feature blending. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019b.

Li Yuan, Yichen Zhou, Shuning Chang, Ziyuan Huang, Yupeng Chen, Xuecheng Nie, Tao Wang, Jiashi Feng, and Shuicheng Yan. Toward accurate person-level action recognition in videos of crowed scenes. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 4694–4698, 2020.

Soumaya Zaghbani and Med Salim Bouhlel. Mask rcnn for human motion and actions recognition. In *Proceedings of the 12th International Conference on Soft Computing and Pattern Recognition (SoCPaR 2020) 12*, pages 1–9. Springer, 2021.

Matthias Zeppelzauer. Automated detection of elephants in wildlife video. *EURASIP journal on image and video processing*, 2013(1):1–23, 2013.

Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022.

Hong-Bo Zhang, Yi-Xiang Zhang, Bineng Zhong, Qing Lei, Lijie Yang, Ji-Xiang Du, and Duan-Sheng Chen. A comprehensive survey of vision-based human action recognition methods. *Sensors*, 19(5):1005, 2019.

Yifan Zhang, Bingyi Kang, Bryan Hooi, Shuicheng Yan, and Jiashi Feng. Deep long-tailed learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. A simple baseline for multi-object tracking. *arXiv preprint arXiv:2004.01888*, 3(4): 6, 2020.

Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. *arXiv preprint arXiv:2110.06864*, 2021a.

Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision*, 129(11):3069–3087, 2021b.

Zhongzhou Zhang and Lei Zhang. Domain adaptive siamrpn++ for object tracking in the wild. *arXiv preprint arXiv:2106.07862*, 2021.

Jiaojiao Zhao, Yanyi Zhang, Xinyu Li, Hao Chen, Bing Shuai, Mingze Xu, Chunhui Liu, Kaustav Kundu, Yuanjun Xiong, Davide Modolo, et al. Tuber: Tubelet transformer for video action detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13598–13607, 2022a.

Zelin Zhao, Ze Wu, Yueqing Zhuang, Boxun Li, and Jiaya Jia. Tracking objects as pixel-wise distributions. In *European Conference on Computer Vision*, pages 76–94. Springer, 2022b.

Rongkun Zheng, Lu Qi, Xi Chen, Yi Wang, Kun Wang, Yu Qiao, and Hengshuang Zhao. Tmt-vis: Taxonomy-aware multi-dataset joint training for video instance segmentation. *Advances in Neural Information Processing Systems*, 36, 2024.

Zhedong Zheng and Yi Yang. Adaptive boosting for domain adaptation: Toward robust predictions in scene segmentation. *IEEE Transactions on Image Processing*, 31:5371–5382, 2022.

Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *Proceedings of the European conference on computer vision (ECCV)*, pages 803–818, 2018.

Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. Omni-scale feature learning for person re-identification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3702–3712, 2019.

Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*, pages 474–490. Springer, 2020.

Xingyi Zhou, Tianwei Yin, Vladlen Koltun, and Philipp Krähenbühl. Global tracking transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8771–8780, 2022.

Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 408–417, 2017.

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.