Time-symmetric tracking of yeast cells and other objects, mitigation of benchmark positional bias

Gergely Szabó

PhD dissertation

supervisor:
Dr. András Horváth

Pázmány Péter Catholic University
Faculty of Information Technology and Bionics
Roska Tamás Doctoral School of Sciences and Technology

Budapest, 2024

Abstract

Temporal forward-tracking has been the dominant approach for multi-object tracking and segmentation for several decades, and this remains unchanged even with the advent of machine-learning-based solutions. While this approach is well-suited for live tracking applications, it imposes significant architectural limitations in scenarios where inference speed is less critical. Therefore, in this dissertation, the primary aim is to showcase a novel tracking architecture developed specifically to circumvent these limitations. Initially, I designed and tuned the architecture for the detection, instance segmentation, and tracking of budding yeast cells in videomicroscopic recordings, where it exhibited superior performance compared to other evaluated tools. Following this, I extended the architecture to accommodate a broader range of image modalities, data variations, and computational demands, enabling a comprehensive evaluation across varied synthetic and semi-synthetic scenarios. These evaluations highlight the advantages and limitations of the architecture when confronted with specific challenges.

Additionally, during the early stages of developing the architecture, I identified a common bias in widely used image processing benchmarks for convolutional neural networks, particularly concerning object positioning. I conducted an in-depth investigation across multiple benchmarks and prediction tasks to validate this bias, revealing severe but often difficult-to-detect consequences stemming from its presence. To address this issue, I experimented with various data manipulation techniques and architectural adjustments, several of which successfully eliminated the bias almost entirely. These findings strongly suggest that the bias is inherently linked to the boundary conditions used in convolutional operations.

Acknowledgement

I would like to express my sincere gratitude to András for his continuous supervision since my early BSc years. He helped me learn the fundamentals of machine learning and computer vision, supported me in exploring my own ideas beyond well-known solutions despite the uncertainty and potential for failure, was always available in times of emergency, responded to my untimely phone calls, assisted with administrative tasks, and encouraged my professional growth even outside of the university.

I am deeply thankful to Zsófi for always being there to discuss and critique my wildest research ideas, to delve into the theoretical and technical intricacies of all our projects, and for supporting my pursuit of a PhD despite its many challenges.

I would like to thank Andrea and Paolo for the opportunity to work with such high-quality data on a state-of-the-art research project with direct applicability to their research group at IFOM, while still allowing me to serve as the primary designer of the architecture. Their invaluable insights from a biological perspective greatly enhanced the project. Additionally, I want to thank Matyi for his work on the application UI, which hopefully transforms the project from a not-so-trivial system into a user-friendly application accessible to laboratories in the near future.

I am grateful to Pázmány Péter Catholic University for accepting me as a PhD student and providing the computational resources essential for my research. I would especially like to thank Dr. Vida Tivadarné for her kindness and support, even when I made administrative missteps.

Finally, I would like to thank my family, who tolerated my highly unconventional work schedule (sorry for frequently walking through the house at 3 a.m.) and supported my aspirations to become a researcher despite the financial challenges.

Contents

Chapter 1

Background and motivation

1.1 Multi-object tracking

Automated detection and tracking of multiple objects remains an important yet challenging task across many loosely related fields. The greatest advancements, however, have primarily been made in the context of autonomous navigation, particularly in real-time applications where predictions must be made immediately after receiving new data. My primary objective was to develop a machine learning-based architecture for the detection, segmentation, and tracking of budding yeast cells in videomicroscopic recordings, intended for public use and specifically for the International Foundation of Medicine (IFOM). Unlike conventional tracking tasks, where predictions are typically made as data is acquired, in this context, tracking is only required after the complete image sequence has been recorded. This approach makes prediction speed less critical and eliminates the data access constraints typically faced by conventional trackers that rely solely on past information. Moreover, in contrast to many other tracking applications, maintaining long-term, continuous tracking of all objects is of paramount importance in this scenario. Many biologically significant parameters — such as the duration of the cell division cycle, cell lifespan, division-based inheritance trees, and the effects of slower-acting chemicals [1, 2] — can only be accurately measured if the tracks remain fully continuous throughout the entire recording. [3, 4] While achieving perfect tracking results may not be feasible without expert supervision, reducing the number of manual corrections needed can substantially enhance the utility

of these predictions. Therefore, rather than using a combination of state-of-the-art methods designed and optimized for real-time tracking scenarios, I opted to develop a novel multi-object tracking architecture with minimal architectural assumptions and constraints, making it as data-driven as possible, and the main focus was placed on achieving stable continuous tracking instead of inference speed.

The concepts and results of this work are detailed in Chapter 2 and are based on my publication: [Ar1].

While the initial macro-architecture was designed based on the task of videomicroscopic yeast cell tracking, it was structured so that only the trained models carry any task-specific information, while the macro-architecture itself remains as general as possible. After successful training on the cell tracking dataset provided by IFOM and conducting comparative evaluations against other competing tools from various perspectives, the architecture was extended to handle both colored and grayscale images through the incorporation of necessary model changes. Object marking was made flexible between centroid and bounding box-based modalities, and the architecture was further optimized to handle long image sequences without memory overflow errors and with reasonable inference time. These enhancements enabled the architecture to be evaluated on various other tracking tasks after retraining the models, such as the MOTS dataset [5, 6, 7], utilizing synthetic training data for zero-shot knowledge transfer learning [8, 9, 10]. Furthermore, due to the relative lack of high-quality multi-object tracking and segmentation benchmark datasets, comparative evaluations of the model were performed in various synthetic scenarios specifically designed to showcase different challenges and aspects of the multi-object tracking and segmentation task, including comparisons against the widely used Kalman filter [11] and ablated variants of the proposed tracker model.

The concepts and results of this work are detailed in Chapter 3 and are based on my publication: [Ar2].

1.2 Benchmark object positional bias

During the initial design phase of the multi-object tracking architecture, an image tiling solution was considered, where training would be performed in such a way that each individual object was centered. However, it was uncertain whether the positional bias introduced by this approach could significantly decrease prediction performance. Although this object-centering tiling method was later discarded due to the adoption of a Mask R-CNN-like model for object detection and instance segmentation [12] and the development of a novel time-symmetric local tracking approach using a different perspective, the question of object centering bias remained relevant, as it was hypothesized that it might have implications for other tasks and datasets. Therefore, an initial test was conducted on the MNIST dataset [13], where objects are naturally centered. These tests revealed unusual fluctuations in predictive performance even with slight object offsets, hinting at the potential presence of a significant bias.

Following the initial tests, the presence of the bias was measured in popular datasets with known object positions, such as MS-COCO [14], and it was found that object positioning is indeed heavily favored towards the center of the field of view. Due to the presence of this bias in widely used benchmark datasets and the initial MNIST tests indicating a potentially substantial impact on predictive performance, evaluation scenarios using modified variants of multiple popular benchmarks were prepared to showcase the severity and locality of the bias. Additionally, saliency (also referred to as activation) maps [15, 16] were analyzed to determine whether the bias was related only to the max-pooling operation [17] present in the architectures used or if it was a more general issue inherent to convolutional neural networks. The evaluation of these saliency maps also helped pinpoint the locality of the bias.

Following these measurements, multiple solutions were proposed to address the bias, including options for both dataset manipulation and architectural modifications. The best-performing solution, which utilized alternative boundary conditions such as toroidal boundary conditions, demonstrated that the likely source of the bias is the widely used zero-boundary condition, which influences the model in an unnatural way if it is not trained to handle object positioning. While

this finding is not entirely new, as other publications have reported correlations between zero-boundary conditions and better localization, and toroidal boundary conditions with more tolerant object positioning, the depth of the analysis and the strength of the findings, even with shifted objects fully present in the images, represent a novel contribution.
The concepts and results of this work are detailed in Chapter 4 and are based on my publication: [Ar3].

Chapter 2

Videomicroscopic yeast tracking

2.1 Author Contributions

While all aspects of the macro-architecture, model design, training, and user interaction functionalities were designed and developed by me, and I wrote the manuscript along with all tables and figures, which serve as the basis for this chapter, the project itself was a collaborative effort. Due to data restrictions preventing access to the test data to maintain its integrity, Paolo Bonaiuti from the International Foundation of Medicine (IFOM) conceptualized the evaluation methods, generated comparative results for the assessed tools, and contributed to manuscript preparation, particularly in Sec. 2.4.1. Andrea Ciliberto from IFOM provided valuable theoretical insights and constructive feedback from a biological perspective, while András Horváth contributed essential theoretical insights and critiques from a machine learning standpoint.

2.2 Introduction

For several decades, tracking of solid objects in video recordings has been a partially solved problem. Classical image processing-based pattern matching approaches, such as Scale-Invariant Feature Transform (SIFT) [18] combined with Random Sample Consensus (RANSAC) [19], have the potential to track objects with fixed shapes invariantly to shift, scale, rotation, and partially to lighting conditions. Furthermore the extension of such methods with variants of the Kalman filter [11, 20, 21] can introduce object velocity or other degrees of freedom

into the estimation, which can substantially reduce the position estimation noise, particularly when the introduced additional parameters tend to be continuous. However, these methods base their estimation on prior assumptions, which greatly reduces their capabilities when the assumptions do not hold true. These limitations are especially apparent in live cell tracking on videomicroscopy recordings, as cells tend to change shape and morphological description rapidly, unpredictably, and in a highly nonlinear manner. Additionally, cells can grow and divide with time, have no real velocity, and often exhibit unpredictable behavior as a living, non-rigid object. Furthermore, cells on the same recordings tend to be very similar, making the task even more challenging. Therefore, the most promising solution to address these issues is using machine learning, where the feature importances and other estimates are learned by the model and depend mainly on the distribution of the training dataset.

In recent years, several cell tracking software and libraries have switched to machine-learning-based solutions, mainly utilizing convolutional neural networks (CNNs), as they generally perform well on image processing tasks. Some of these tools, such as CellTracker [22], Usiigaci [23], and YeaZ [24], separate segmentation and tracking and use deep learning only for high-quality segmentation. However, they use classical methods, such as the Kalman filter, bipartite graph matching, or other methods for tracking which ultimately rely on a rigid metric system. In contrast, other tools, such as CellTrack R-CNN [25], and Ilastik [26], adopt deep learning or other forms of machine learning for tracking either in an end-to-end manner or as a separate segmentation followed by tracking. This approach shows advantages both theoretically and empirically, as demonstrated by the often superior performance of machine learning based and other data-driven methods when sufficient amount of training data is available. [27, 28] However, to the best of our knowledge, all of these tools rely on a frame-by-frame identity matching approach, utilizing only consecutive frames for tracking. This approach is a major simplification and disregards a large amount of useful information that could be present in the temporal neighborhood of the given video frames.

Therefore, we developed a complete tracking pipeline that separates segmentation and tracking because end-to-end training with our method was unfeasible. Our method uses deep learning for both stages utilizing the temporal neighborhood

of consecutive frames for prediction, makes skipping of frames possible up to the size of the temporal neighborhood making the tracking substantially more robust and uses metrics only between different predictions for the same cell on the same frame minimizing the introduction of methodical assumptions via the choice of the similarity metric. The most similar method in the literature that we know of is DeepTrack [29], which was recently developed for tracking cars. However, while this method utilizes the local temporal neighborhood via encoding objects using Temporal Convolutional Networks (TCNs), the network architecture and the metric system for object matching is substantially different, making our solution architecturally novel.

2.3 Methods

2.3.1 Instance Segmentation

In case of tracking pipelines not trained in an end-to-end manner, segmentation is commonly employed as a preliminary stage before tracking. The reason being that the detection of the objects is mandatory for tracking, and detection via segmentation can provide essential information about the objects present, simplifying and improving the manageability of the tracking task. In the context of cell tracking, high-quality segmentation of cells is also important from a biological perspective as it offers valuable information regarding the size and shape of cells, as well as serving as a quality check for the tracking process.

From both the target and solution architecture perspectives, segmentation can be classified into semantic segmentation and instance segmentation. In the realm of CNN-based image processing, semantic segmentation is typically considered the easier task and can be accomplished using relatively simpler network architectures like SegNet [30], U-Net [31], and others. However, accurately separating objects of the same category after semantic segmentation can be highly challenging and may necessitate the utilization of classical image processing methods, which we previously discussed as being less desirable. On the other hand, recent advancements in deep learning, particularly the introduction of Mask R-CNN-like architectures [12], have made it possible to achieve instance segmentation

solely based on deep learning techniques. Although these architectures tend to be more complex and rigid due to the combination of convolutional, fully connected, and potentially other network connection types, they are highly valuable when it is essential to separate instances of objects belonging to the same category.

In our approach, we employ a distinct step of Mask R-CNN-based instance segmentation prior to tracking, grounded in the previously described arguments. This step serves two purposes: to provide accurate segmentation results and to serve as initialization for the tracker. As high quality environments are publicly available for Mask R-CNN architeture training, instead of focusing on developing a novel architecture, we trained an instance segmentation model using the Detectron2 [32] environment. To improve the performance and robustness of the model, we employed multiple data augmentation techniques during the training process.

The specific model we chose was a Mask R-CNN architecture with ResNet-50 feature pyramid backbone pretrained on the COCO instance segmentation dataset [14] with 128 ROI heads for the single "cell" object type. The model was trained through 120,000 iterations with a base learning rate of 0.0025, minibatch size of 4 and otherwise the deafult parameters of the Detectron2 environment. For artificial data augmentation, we employed various transformation options provided by the Detectron2 environment and the Albumentations library [33]. The chosen transformations and their respective parameters are summarized in Tab. 2.1. The transformations and their parameters were determined in a qualitative manner to ensure that the resulting outcomes closely resemble biologically plausible samples, while maximizing variance. However, conducting an extensive data-specific parameter search could potentially yield even better results.

The trained instance segmentation model showed exceptional performance in both object detection and shape recognition. Therefore we decided to not only use its results as inputs for the subsequent tracking method in the pipeline but also utilize them as the final segmentation output. This decision was made in contrast to the option of using segmentation estimates of the tracking method. Examples of such instance segmentation results can be seen in Fig. 2.1. Using these initial instance segmentation predictions as segmentation instances in tracking effectively combines the strengths of both stages of the pipeline: the segmentator module produces more accurate segmentations, while the tracker connects them.

Detectron2 transformations	
Transformation	Parameters
Random Brightness	Intensity Min = 0.7, Intensity Max = 1.3, P = 0.2
Random Contrast	Intensity Min = 0.7, Intensity Max = 1.3, P = 0.2
Random Flip	P = 0.1
Random Extent	Intensity Min = 0.7, Intensity Max = 1.3, No Shift, P = 0.2

Albumentations transformations	
Transformation	Parameters
Grid Distortion	Num Steps = 10, Distort Limit = 0.2, P = 0.2
Elastic Transformation	Alpha = 991, Sigma = 8, Alpha Affine = 50, P = 0.2

Table 2.1: **Employed augmentation techniques**
Data augmentation transformation parameters for Detectron2 and Albumentations-based instance segmentation training

2.3.2 Tracking

While deep-learning based segmentation has gained popularity in modern cell tracking software, tracking often relies on comparing consecutive frames using metrics like Euclidean distance or Intersection over Union (IoU). In contrast, more advanced but still rigid methods, such as variants of the Kalman filter, consider motility patterns along with position and shape for tracking. Following such frame-to-frame similarity measures, various techniques, including variants of the Hungarian method [34] can be employed for optimal assignment without repetition.

However, these methods are sensitive to several hyper-parameters, artifacts, and unusual cases. Even with perfect segmentation and correct assignment, frame-to-frame metrics are not expected to yield a perfect match unless the object moves in a completely predictable way, with estimable process and measurement noises. [20, 21] This sensitivity arises due to the movement, speed, shape changes, and other descriptors of the object, as well as environmental factors such as lighting conditions. Consequently, manual fine-tuning of parameters on a per-recording basis, or even adopting different parameters within a single record, is often necessary.

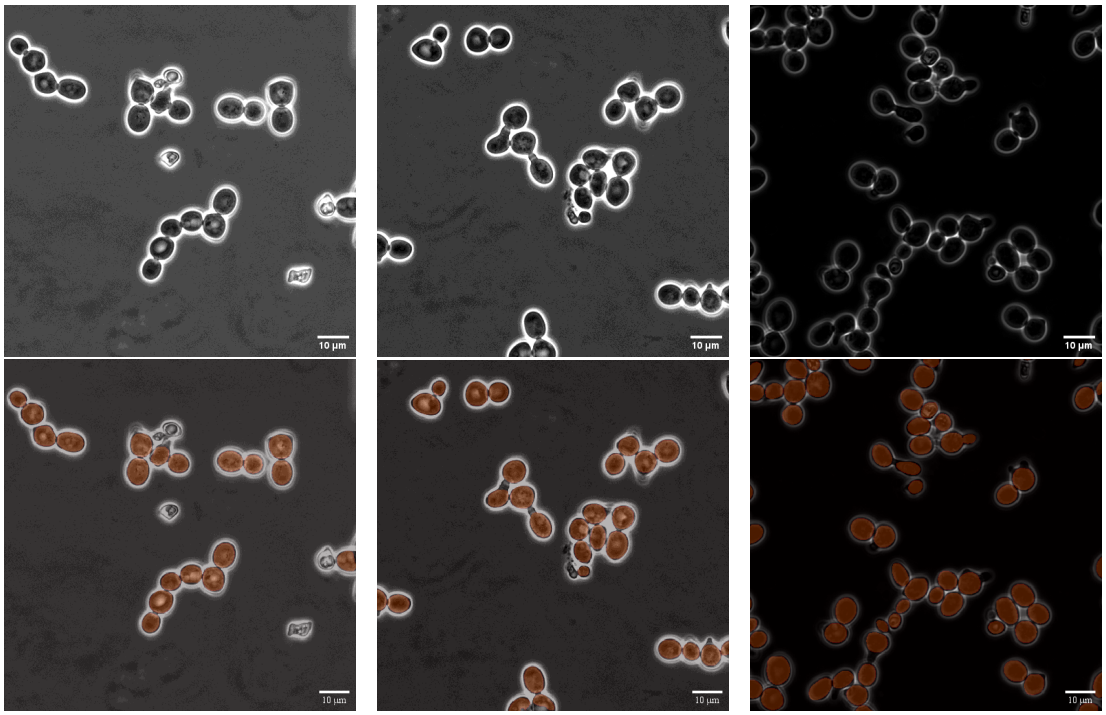


Figure 2.1: **Segmentation sample results**
A showcase of robust segmentation results, depicting accurate cell detection and segmentation under varying lighting conditions, while effectively avoiding detection of obviously dead cells. The upper row displays the raw input samples, while the lower row shows the predicted segmentation instances overlaid.

Machine-learning based techniques offer an alternative to address these challenges, as they can learn the movement patterns of the tracked object and adapt to environmental changes when provided with diverse training data. Artificial data augmentation can also be used to enhance data variety or simulate specific anomalies. However, in popular cell tracking software, these solutions are usually implemented on a frame-to-frame level, potentially leading to sensitivity to unexpected temporally local anomalies, such as changes in lighting conditions or administration of drugs that alter cell movement patterns or morphology.

Instead we propose a novel multi-frame based assignment method that predicts the position of a marked cell on several consecutive frames using a state-of-the-art semantic segmentation architecture. Our method is based on the concept that, although the direction of time is important in cell tracking, certain morphological

changes such as cell growth generally occur in the positive temporal direction. Therefore, having information about the future position of the cell can be advantageous for tracking. As a result, we have developed a time-symmetric tracking method that utilizes the local temporal neighborhood in both the negative and positive temporal directions. Based on this architecture, the model learns to capture the direction of time and temporally unidirectional changes from the data, instead of relying on a temporally biased architecture. Our approach draws notable inspiration from the 2019 study of X. Wang [35]. However, instead of using cycle consistency losses, we sought to create an architecture that remains invariant to the direction of time itself. After the predictions are generated by this architecture, the resulting local tracks can be optimally matched and linked together based on prediction information within their temporal overlaps using the Hungarian method.

Local Tracking

The local tracking around a single frame is performed on all cells individually distinguished by the previous instance segmentation step via a neural network designed for semantic segmentation. The channel parameters of this network can be defined the tracking range (TR) parameter. The segmentator model input for a single cell instance consists of $(TR + 1 + TR) + 1$ images as channels of the single input where $TR + 1 + TR$ images are raw video frames centered around the frame on which the given cell is to be tracked, while on the additional 1 channel a single white dot marks the cell to be tracked at its centroid and the rest of the image is black. The determination of the centroid coordinates (x_c, y_c) is based on the segmentation instance for the given cell using image moments. The equations are as follows:

$$x_c = \frac{\sum\sum x \cdot I(x,y)}{\sum\sum I(x,y)} \tag{2.1}$$

$$y_c = \frac{\sum\sum y \cdot I(x,y)}{\sum\sum I(x,y)} \tag{2.2}$$

The target outputs for the local tracker consist of $n + 1 + n$ frames, containing the segmentation of the cell marked on the last channel of the input. An illustration of this local tracking architecture is presented in Fig. 2.2.

The reason for using the centroid of the segmentation and not the segmentation itself in the input is that according to our experiments if the segmentation information would be present there, the local tracker would not learn how to perform semantic segmentation. Instead it would copy the input segmentation to all outputs, providing false segmentations to all outputs except the middle one. However via using the centroid as a marker, the local tracker is forced to learn how to perform segmentation for the single marked cell on all temporal instances present, and thus it will be able to perform tracking via segmentation. Furthermore, if a cell instance is temporally close to the beginning or end of the recording, such that it is within n frames of the boundary, direct forward or backward tracking using a kernel size of n+1+n is not feasible. Nevertheless, the number of channels, which determines the local tracking distance, is a fixed parameter of the network architecture. To address this issue, we incorporate temporal padding by repeating the first or last frame. Additionally, we ensured that such instances are included during training to make the network capable of handling such edge cases and maintain its temporal invariance.

Based on empirical evidence, we found that among the tested semantic segmentation models, variants of the DeepLabV3+ [36] architecture from Pytorch Segmentation Models [37] yielded the best performance for this task. The models were trained through 20 epochs with a minibatch size of 10 in all experiments using stochastic gradient descent optimizer with cosine annealing learning rate scheduling having warm restarts during the first 15 epochs, and a gradual cooldown during the last 5 epochs. The model output had sigmoid activation with binary crossentropy loss.

Artificial data augmentation of the local tracking samples was performed using the torchvision transformations library [38]. For positional transformations random horizontal flip [p=0.5] and random affine [degrees=(-40, 40), scale=(0.7, 1.3), p=0.4] transformations were uniformly applied to the input and ground truth data. For color transformations color jitter [brightness=0.5, contrast=0.5, p=0.4] transformation was applied for each input frame individually to improve model robustness for unexpected lighting artefacts.

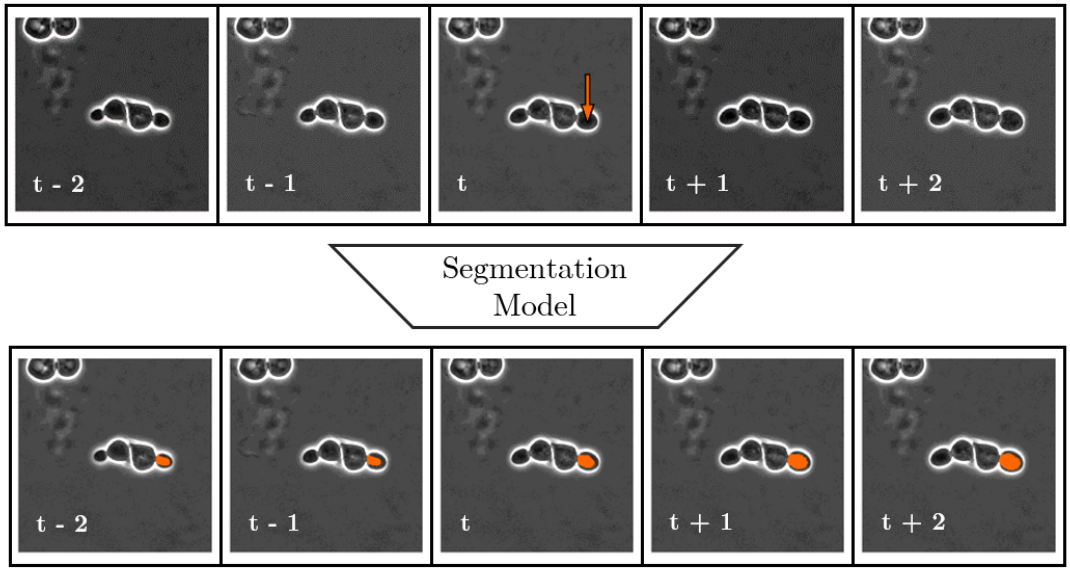


Figure 2.2: **Local tracking structure**
Depiction of input and output information structure of local tracking via segmentation with local tracking range $(TR) = 2$. Notice that while the left to right positive temporal direction of the data is recognizable from cell growth, and the model learns this during training, the architecture itself has no built in directional preference, resulting in a completely data-driven estimation for local tracking.

Global Tracking

As the output of local tracking, a single cell instance marked by the video frame and its position is tracked forwards and backwards through TR frames. This local tracking output can be obtained for each cell instance detected by the instance segmentation step, serving as the input for global tracking. Therefore the aim of global tracking is to match local tracks based on the semantic segmentation predictions in their overlapping areas, and chain them together based on ID matching creating full cell tracks based on a globally optimal consensus.

In practice, at first, we obtain this global consensus on a t to $t + 1$ level, as usually the cells can be tracked from one frame to the next. However, if a cell instance did not receive any matches above the minimal requirement threshold, the matching can be repeated for larger temporal distances. This secondary matching is performed after all t to $t + 1$ matching is done, and only between

the remaining candidates. Secondary matching is also performed hierarchically in terms of temporal difference meaning that temporally closer instances get matched first and only the remaining ones can get matched later with larger temporal differences. The maximal possible matching distance is $2TR$, but matches over TR are unreliable as they are based on relatively few segmentation predictions, and only indirectly contain information about the segmentation of the central cell instances to be matched. Therefore we believe, using TR as the maximal temporal distance between local track matching is logically the most sound choice, but in practice other choices could have minor benefits depending on the dataset.

Based on this methodology, the global consensus of ID matching can be turned into a one-to-one assignment problem for $N \times M$ candidates, where N is the number of candidates in frame t and M is the number of candidates in frame $t + \Delta t$ where $1 < \Delta t \leq 2TR$. Such assignment problems can be optimally solved by variants of the Hungarian method in polynomial time even for non-square matrices, but this requires a sound metric choice to measure the goodness of the candidate matches. In our case these candidate matches are the $2TR + 1 - \Delta t$ segmentation pairs in the overlapping temporal region of the local tracks to be compared. Therefore in an ideal scenario where perfect segmentations are achieved in local tracking, any metrics that solely compare segments within the same frame and calculate the average of these comparisons could be employed. This is because segmentation instances belonging to the same cell should result in a perfect match. However in practice, the mean of Intersection over Union (IoU) turned out to be the best metric choice as it takes both the positional and morphological differences of the segmentations into account, and gives 0 similarity for segmentations with no overlap regardless of the distance. This process of calculating the metric similarity measure based on local tracks is depicted in Fig. 2.4.

After computing the selected metric between all global tracks of frame t and $t + \Delta t$, a threshold can be employed to establish a minimum required similarity based on the metric results. This step eliminates candidates that do not meet the desired level of similarity. Subsequently, the Hungarian method can be applied to determine the optimal global consensus for ID matching between frame t and $t + \Delta t$, as depicted in Fig. 2.5. Via performing these steps in the hierarchical approach as described earlier, complete global tracks are constructed based on

a global consensus of similarity, which solely relies on the prediction capabilities of the local tracker. The construction of global tracks is performed using the depth-first search graph algorithm [39] to cover all existing branches arising from connections with a higher temporal distance than one frame. An example of successful global tracking using this pipeline is presented in Fig. 2.3.

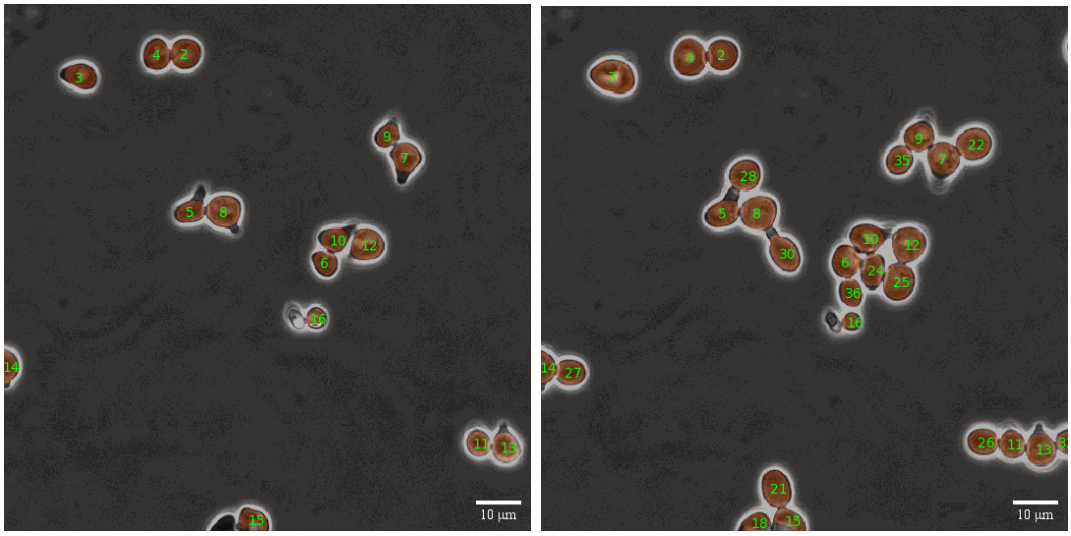


Figure 2.3: **Tracking sample results**
A side by side display of tracking results, demonstrating successful tracking of all cells in the same recording with a temporal difference of 8 frames. New cells were appropriately assigned new IDs while maintaining consistent tracking of existing cells.

Skipped Instance Interpolation

By employing the aforementioned methodology for global tracking, we can effectively address inconsistency errors in segmentation that commonly arise in realistic scenarios when $\Delta t > 1$. In such cases, although the resulting tracks will be complete, they may not be continuous, and segmentations for missed frames will be absent. To tackle this issue, we use a positional linear interpolation method, which is described by the following equations:

$$\Delta c(x,y) = \frac{(t - t_{\text{last}})c_{\text{next}}(x,y) + (t_{\text{next}} - t)c_{\text{last}}(x,y)}{t_{\text{next}} - t_{\text{last}}} \tag{2.3}$$

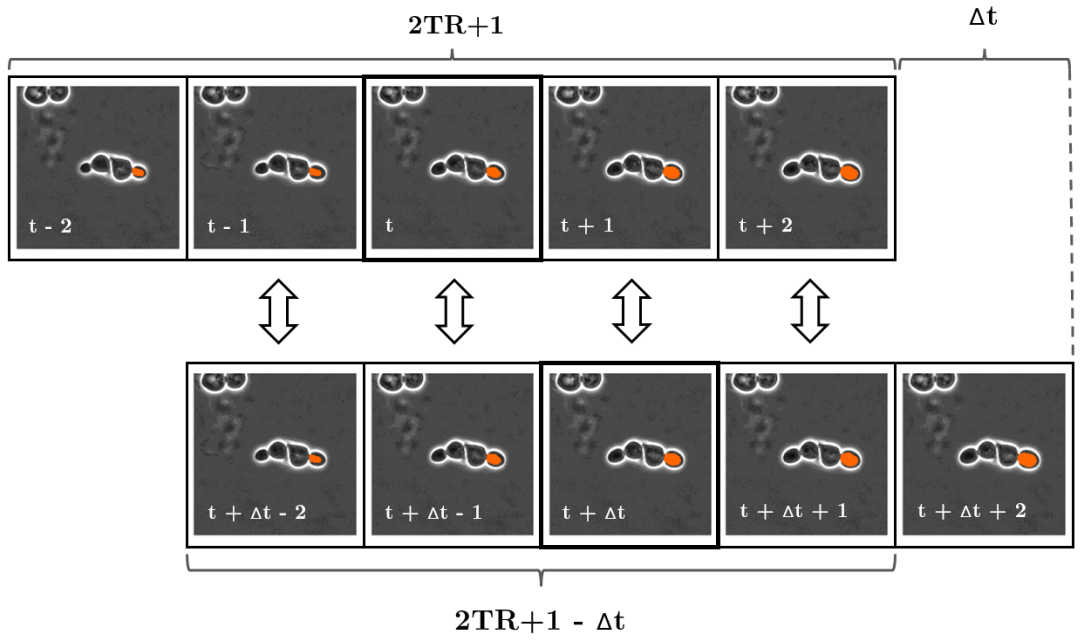


Figure 2.4: **Metric similarity structure**
Schematic structure of the metric similarity measurement step between $2TR+1$ long local tracks of different cell instances on frames with a temporal distance of Δt. The solid lines indicate the central segmentation instances with a Δt temporal distance to be matched, while the arrows indicate the similarity metric between the segmentation estimates for each time frame. Subsequently, the individual metric results are averaged to obtain a single measure of similarity.

$$S(x,y) = S_{\text{last}}(x,y) + c(x,y) \tag{2.4}$$

Here, $S(x,y)$ and t represent the interpolated segmentation and its corresponding time, respectively. $S_{\text{last}}(x,y)$, $c_{\text{last}}(x,y)$, and t_{last} denote the segmentation, centroid, and time of the last occurrence of the cell with the given ID, while $c_{\text{next}}(x,y)$ and t_{next} represent the centroid and time of the next occurrence of the cell with the given ID. Although this method only shifts the segmentation from the last occurrence to the linearly interpolated position without altering its shape, in practice the results have proven satisfactory. Still, in the future, incorporating a method that linearly interpolates the shape of the segmentation by considering both the last and next occurrences could potentially yield minor improvements.

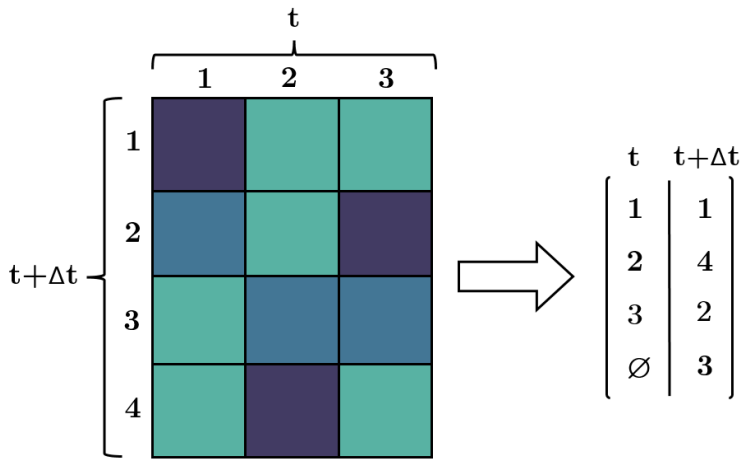


Figure 2.5: **Global assignment structure**
Illustration of the metric similarity based assignment step using the Hungarian method for non-square matrices. Newly unassigned cells, such as cell number 4 receive a new ID, while the previously assigned cells retain the ID of their corresponding previous instances.

Ancestry Assingment

Our architecture currently does not include an integrated solution for assigning newly born cells to their ancestors. However, the stability and continuity of the predicted cell tracks suggest the feasibility of implementing such a feature later if required. Initial tests show that a basic prediction method using Euclidean distance, combined with positional and morphological heuristics like maximal newborn cell size, can produce satisfactory results for ancestry assignment when the predicted tracks are unbroken. For example, a somewhat similar simple distance-based method was employed in the Cell Tracking Challenge submission by A. Arbelle (2021) [40, 41]. However, for more complex samples, additional morphological factors may need consideration, such as connectedness, or machine learning-based solutions could be utilized as a separate module if an ample amount of training data is available. Additionally, there is the option of integrated object instance classification of division, akin to the approach described in the publication by I. E. Toubal (2023) [42]. However, this would further increase the data intensity of architecture training and could introduce a potential weakness in the tracker

module due to false positive cell division predictions dissecting some of the tracks.

2.4 Results

For evaluation purposes, we opted to calculate F-scores based on binary decisions of correctness for segmentation and tracking, instead of utilizing continuous regression metrics. This approach offers a clear numerical evaluation of the capabilities of the compared tools and tool versions. F-scores for segmentation were computed using IoU, with a minimum similarity threshold of 0.5. We selected this threshold as it is strict enough to assess the accuracy of correct cell detection while disregarding minor segmentation discrepancies that hold minimal biological significance. In the case of tracking evaluation, F-scores were also calculated based on the segmentation IoU, with the additional inclusion of ID matching. True positive values, also referred to as "links" were registered if the segmentations matched on the same frame, and the IDs matched consecutively for both the prediction and the ground truth. This tracking evaluation method of link matching is described in the work of M. Primet (2011) [43].

2.4.1 Yeast Data Source

The training and validation data used to generate the primary results in this paper are videos of budding yeast cells dividing in a microfluidic device. Cells are *S. Cerevisiae* wild-type-like, with W303 genetic background. Yeast is a unicellular eukaryote, which shares many essential genes and phenotypes with multi-cellular eukaryotes, such as mammals. On the other hand, its relatively small genome ($\sim 12 \times 10^6$ base pairs), short doubling time and easy genetic tools allow to perform experiments in a short time and with limited costs. Such experiments would be very costly or even impossible to do in higher eukaryotes. Not surprisingly, many discoveries performed in higher eukaryotes were inspired by original studies performed in yeast.

In the experiments, cells were grown overnight at 30 °C in complete YPD medium and synchronized by the use of the pheromone α-factor. Imaging started when they were released from this stimulus. Cells were free to grow and duplicate, trapped in the microfluidic device to prevent them from moving in the field of

Tracking Range	Valid Local Tracks	Training Samples	Validation Samples
1	99019	88993	10026
2	64659	58059	6600
3	47940	43018	4922
4	37347	33487	3860
5	29888	26769	3119
6	24373	21803	2570
7	19931	17819	2112
8	16278	14533	1745
9	13301	11847	1454
10	10915	9709	1206

Table 2.2: **Yeast sample distribution**
Number of valid local tracking samples for model training in function of tracking range (TR), and the disribution of training and validation samples.

view. Images were acquired every 3 minutes for 3.5 hours using a DeltaVision Elite imaging system equipped with a phase-contrast objective. The field of view is a square of 111.1 μm size, resulting in a 512x512 pixels image. The duration of the cell-cycle of the analysed cells is $\sim$ 1.5 hours, while their size is $\sim 5\mu$m.

The segmentation and tracking models were trained on a dataset comprising of 314 movies, and for validation purposes during model design, 35 additional movies were used. These movies were acquired using the same microscope, objective, and image size, but with varying durations and time-lapses. On average, each movie contained 35.8 frames and 20.2 individual cells to be tracked. The ground truth labels were initially generated using Phylocell [44] in a semi-automated, semi-manual manner. They were then manually corrected and curated by multiple experts, who removed incorrect segmentations (such as dead cells, segmentations of empty space, and multiple segmentations per cell) and corrected cell tracking. The training and validation sample numbers of the segmentation model were independent of model parameters, while for the tracking model, the model parameter tracking range (TR) substantially influenced the number of available samples, as shown in Tab. 2.2. Further details on the model parameter TR will be discussed in Sec. 2.3.2. For final testing and evaluation of the tools and model parameters, 5 additional independent recordings were used, containing a total of 4629 identified cell instances.

2.4.2 Comparative Datasets Definition

While our architecture was initially designed for the segmentation and tracking of yeast cells, we also claim that it is capable of generalization to diverse data types, including those featuring faster-moving objects, given an adequate amount of training data. Additionally, in contrast to numerous other methods, our tracking architecture seamlessly integrates both morphological and kinematic information, making it entirely data-driven and retrainable without requiring any substantial architectural changes.

To support these claims, we aimed to train and evaluate it using microscopic recordings of other cell types. However, we found that such data is not readily available in the required quantity and quality. Our architecture demands extensive training data due to its size and complexity, as a trade-off for tracking quality and stability, which poses a challenge compared to simpler tracking solutions. For instance, the Cell Tracking Challenge [40] or the CTMC-v1 dataset [45] could have been suitable for evaluation purposes. However, they lack the necessary intra-class variance required for successful training of complex models without severe overfitting on the training data. This issue was also noted in the publication of I. E. Toubal (2023) [42].

To address this issue, we synthetically created multiple toy datasets comprising objects with various morphological and motility patterns. To introduce a level of difficulty and realism to the tracking tasks, each simulated recording includes a randomized background composed of 10 Gaussian density functions, with a maximum relative brightness level of 0.39, and a similar framewise-random noise with a maximum relative brightness level of 0.078.

Synthetic Arrows

The objects designated for tracking are triangular arrows, with a maximum size difference ratio of 6, an average speed equivalent to 40% of the mean object size, uncorrelated with the size of the individual object, allowing for speeds even greater than the object size. Additionally, these objects have a maximum angular rotation of 10° for each frame, allowing for object crossing and thus occlusion, and random uniform grayscale values exceeding a brightness level of 0.39. Furthermore, the

objects may also expand from one frame to the next by 1.6-10.0% of their original size, with a probability of 0.33 at each time step. The average density of objects in this scenario is ~ 11.39 per frame.

Synthetic Amoeboids

This object tracking task is similar to "Synthetic Arrows". However, the objects to be tracked are more complex amoeboids with semi-random initial shapes and shape changes from one frame to the next. Both the initial shapes of these objects and the iterative shape changes are generated using Perlin noise [46] in a 50-point polar coordinate system with an octave value of 6, a persistence value of 0.5, and a lacunarity value of 2.0. Furthermore, the shape changes are governed in a Gaussian manner based on the distance from the object centroid to avoid complete filling of the object space or the disappearance of an object due to random chance. This approach creates amoeboid-like objects with randomized boundaries allowing for highly concave contours and even the short-term splitting of an object in extreme cases. Additionally, while the individual shape changes are gradual from one frame to the next, the objects can be nearly unrecognizable over larger temporal distances. Lastly, unlike the "Synthetic Arrows", the objects perform perfectly rigid collisions with no velocity loss, making occlusion impossible. The main reason for this choice is that the following other datasets featuring synthetic amoeboids would require a realistic simulation of occluded object light transference, which would be mandatory but exceedingly challenging if object occlusions were allowed. Furthermore, object occlusions are generally less important in microscopy compared to macroscopic cameras, due to the focal plane specificity of microscopes. The average density of objects in this scenario is ~ 9.18 per frame.

Synthetic Amoeboids-PC

The aim of this dataset is to simulate object tracking on phase contrast microscopy recordings with objects displaying substantially different morphological and motion characteristics compared to yeast cells. The behavior of the objects in this dataset is identical to that of "Synthetic Amoeboids." However, instead of displaying objects with uniform grayscale color, the Canny edge detector [47] is applied to

the binary object mask, followed by a Gaussian blur with a large kernel size of 51 pixels, mimicking phase contrast microscopy visual characteristics in a simplistic manner. The average density of objects in this scenario is ~ 8.87 per frame.

Synthetic Amoeboids-PCC

The objects in this dataset behave similarly to those in "Synthetic Amoeboids-PC," with the only notable difference being that the objects undergo non-rigid collisions, resulting in a 10% velocity loss for each collision involving both objects. This seemingly minor change leads to the clumping of objects, making the tracking task simultaneously easier and more challenging. Object identification and segmentation become substantially more difficult due to the clumping, while the reduced momentum simplifies assignment. Additionally, a small force is applied to each object towards the center of the field of view. Although this force does not visibly alter object motility, it ensures that the object clump is likely to form within the field of view. The average density of objects in this scenario is ~ 20.27 per frame, much higher than in the previously discussed scenarios, as expected.

Synthetic Amoeboids-PCCA

This dataset does not exhibit any differences in terms of object behavior compared to "Synthetic Amoeboids-PCC." Instead, each recording is augmented with a highly disruptive artifact, which poses challenges for both object detection and tracking, even for the human eye. These artifacts consist of 100 white lines placed in a uniform random manner between the edges of the field of view, obscuring both the objects and the background. The artifact patterns generated in this way are static but unique to each recording, rendering it impossible for models to learn their positions. The average density of objects in this scenario is ~ 21.18 per frame, similarly high as in case of "Synthetic Amoeboids-PCC".

2.4.3 Comparative Tool Evaluation

For comparative evaluation, we selected Phylocell [44] [48] and YeaZ [49] as the two other cell tracking tools. The reason behind this choice was that the ground truth training and testing data were generated using Phylocell and were later manually

corrected by experts. On the other hand YeaZ provides a fair comparison to our solution as it employs a similar segmentation pipeline and a somewhat comparable tracking pipeline, utilizing the Hungarian method. As the ground truth data excludes cells that eventually leave the microscope's field of view throughout the recording (from now on referred to as "border tracks"), we conducted two evaluation scenarios for the tools. In one scenario, border tracks were removed in post-processing from all predictions to ensure an unbiased comparison, while in the other scenario, the prediction results were left unaltered.

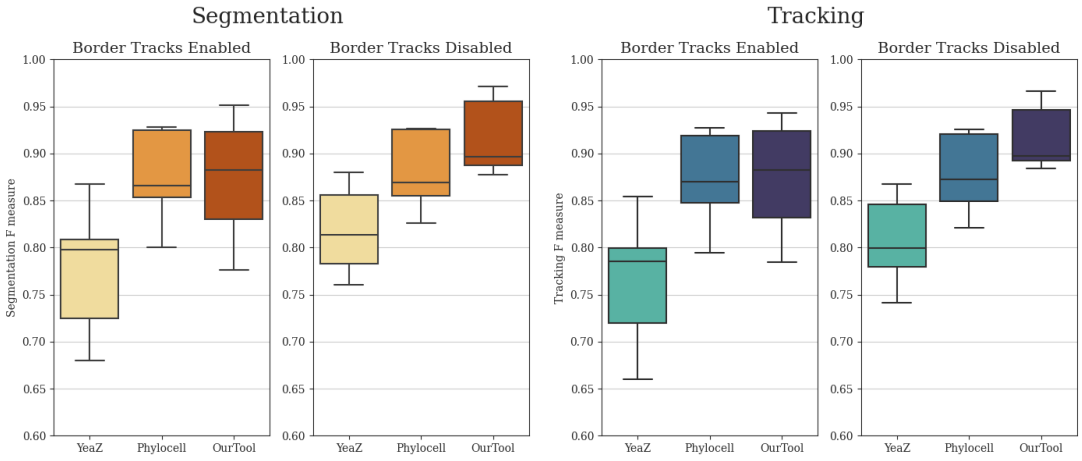


Figure 2.6: **Comparative tool evaluation**
Comparative evaluation of segmentation and tracking performances of Phylocell, YeaZ and our pipeline based on F-measures of segmentation IoU and tracking link matches with enabled and disabled border track predictions. Disabled border tracks present an unbiased comparison between the tools, while enabled border tracks show results with the unaltered outputs.

The segmentation and tracking F-measure results are depicted in Fig. 2.6. It is evident that our solution substantially surpasses both other tools in terms of segmentation and tracking quality, particularly in the more equitable scenario where the border tracks were eliminated. In the unaltered prediction case, the performance of Phylocell remained comparable to its prior results since the tool already automatically removes most border tracks. Conversely, YeaZ and our solution exhibited notably lower performance due to domain discrepancy between the predictions and ground truth data. Nevertheless, even in this case, our

solution outperformed Phylocell in terms of expected F-score values, although with a substantially higher degree of variance on the lower side.

These results indicate the effectiveness of our method, but also suggest at a strong connection between segmentation and tracking quality due to the apparent correlation between metric results. To investigate further, we used Phylocell instance segmentation inputs for our tracker. In this case, the mean tracking F-score of our method was 0.868 ± 0.015, showing a negligible difference compared to the tracking F-score of Phylocell at 0.878 ± 0.02. While our tracking method did not surpass the performance of Phylocell on the measured samples, this may be attributed to a highly non-uniform segmentation error distribution of Phylocell, which leads to certain track predictions being mostly incorrect while others are mostly correct. Although our tracker is theoretically capable of correcting sparse errors in mostly correct tracks, it is unable to rectify tracks composed predominantly of faulty or missing instance predictions. To verify this, we conducted an experiment detailed in Sec. 2.4.5, introducing segmentation errors in a more uniform manner.

2.4.4 Hyperparameter Dependencies

The parameters we considered to have a major impact on tracking performance for both local and global tracking are the local tracking range, the complexity of the local tracker model backbone, and the metric used for global consensus. For a more detailed description of these parameters and their functions in the architecture, please refer to Sec. 2.3.2.

The differences in tracking F-scores resulting from these parameters are presented in Fig. 2.7. Somewhat surprisingly, we observed minimal differences in performance based on the local tracking range and model complexity. Furthermore, smaller local tracking ranges exhibited marginally better performance, suggesting that the benefits of larger tracking kernels were outweighed by the increased model complexity, which slightly hindered training. On the other hand, these results also indicate that lightweight models with smaller local tracking ranges and simpler backbones are suitable for tracking predictions, resulting in substantially shorter inference times. As an additional benefit, while performing inference without

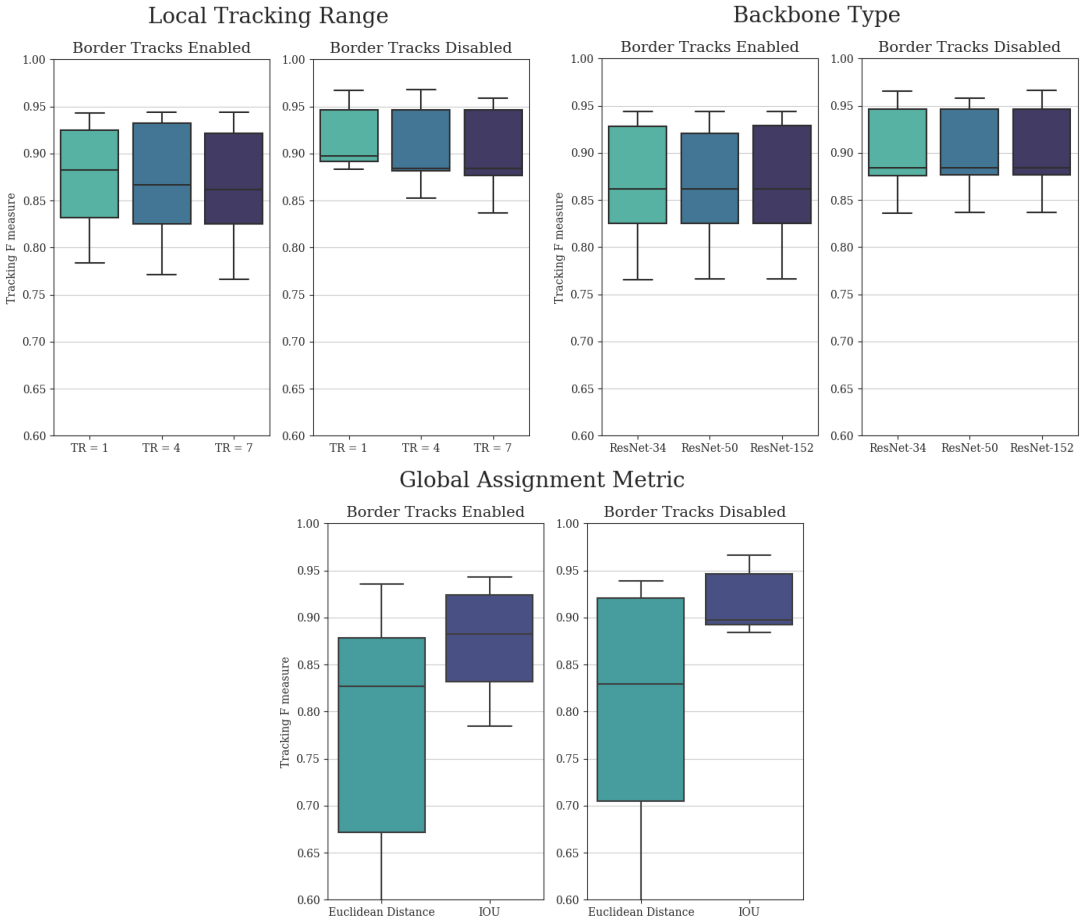


Figure 2.7: **Hyperparameter dependencies**
Comparative evaluation of tracking F-scores in function of local tracking ranges, backbone model complexities and global consensus metrics with enabled and disabled border track predictions. Disabled border tracks present an unbiased comparison between the parameters, while enabled border tracks show results with the unaltered outputs.

GPU acceleration remains slower by orders of magnitude, it is more reasonable with such lightweight models, making initial pipeline testing more accessible. As anticipated, the use of IoU as the metric for global consensus substantially outperformed Euclidean distance, underscoring the importance of incorporating cell morphology in short-term tracking.

2.4.5 Tracking Robustness

To comprehensively demonstrate the tracking stabilization and segmentation interpolation capabilities of our architecture, and to show that tracking quality is not solely reliant on initial instance segmentation quality, we conducted an ablation study. In this study, we removed every 1:15 (Noise 1:15) and 1:5 (Noise 1:5) segmentation instances in a uniform random manner before tracking. Furthermore, to showcase the potential advantage of longer local tracking ranges, we created a scenario where segmentation instances were removed in 7-frame-long blocks, with these elimination blocks also positioned in a uniform random manner with a 1:5 average chance of segmentation instance elimination (Box Noise 1:5). This box noise scenario presents a different but equally realistic challenge compared to the fully uniform noises, as in various applications, objects can disappear for several frames due to occlusion and limited field of view.

The applied noises substantially impacted both segmentation and tracking outcomes, as segmentation instances were removed and the previously continuous tracks were broken up. Therefore, it was the task of the tracker module to create continuous tracks despite the missing segmentation instances and to interpolate the removed instances as effectively as possible. The resulting F scores before and after re-tracking are presented in Fig. 2.8 for Noise 1:15 and Noise 1:5 with the thus far best performing local tracking range of 1. For Box Noise 1:5, we measured the performance of the architecture for local tracking ranges of 1, 4, and 7. These F scores are presented in Fig. 2.9.

The results reveal that for all examined noises, both the segmentation and tracking F scores showed substantial improvement compared to the disrupted tracks due to the interpolated segmentation instances and the reconnected tracks. Furthermore, in the case of Box Noise 1:5, the longer local tracking ranges were preferred, as they provided better coverage for the continuously missing segmented instances. These findings clearly demonstrate that while there is a substantial correspondence between instance segmentation and tracking results, both modules of our method strongly support each other, leading to simultaneous improvements in both, thus contributing to the observable correspondence.

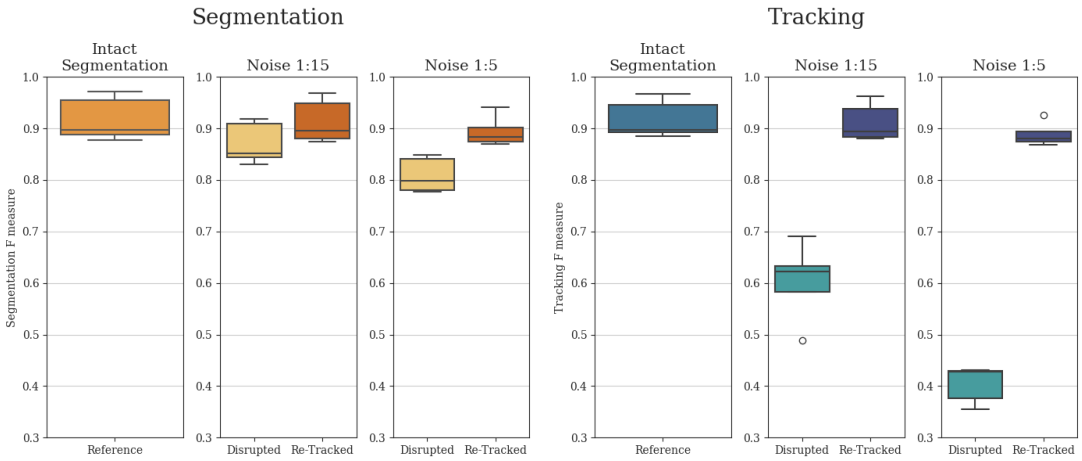


Figure 2.8: **Uniform random noise reconstruction**
Comparative evaluation of segmentation and tracking F-scores for Intact, Noise 1:15, and Noise 1:5 cases. Both segmentation and tracking results include baseline values disrupted by the given noise, as well as re-tracked values initiated with the disrupted segmentation.

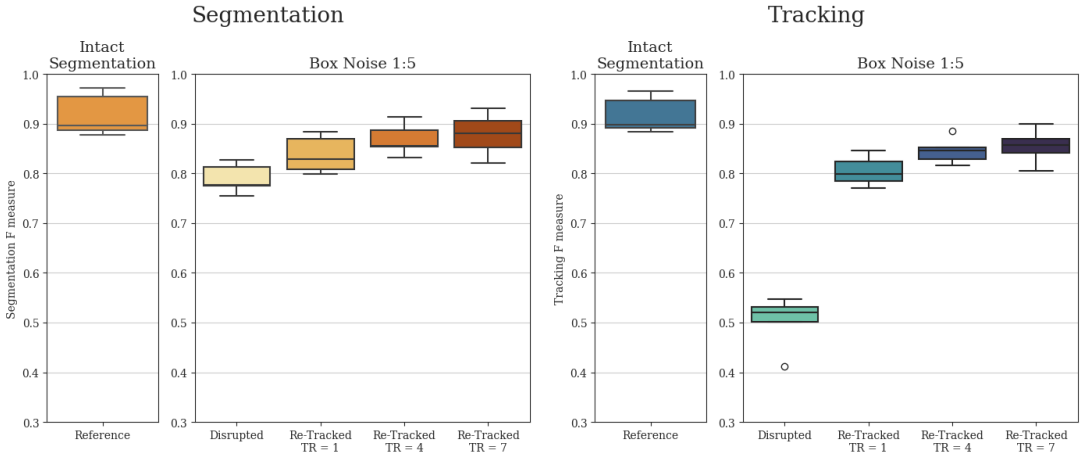


Figure 2.9: **Box noise reconstruction**
Comparative evaluation of segmentation and tracking F-scores for Intact and Box Noise 1:5 cases. Both segmentation and tracking results include baseline values disrupted by the noise, as well as re-tracked values initiated with the disrupted segmentation using local tracking ranges (TR) of 1, 4 and 7.

Moreover, during this experiment, we observed a substantially greater influence of tracking parameters compared to previous trials. Specifically, the accuracy of the segmentation confidence threshold of the tracking model and the minimal

similarity threshold of the global assignment step became notably more crucial, particularly in scenarios of reduced instance segmentation quality. This was especially evident with more complex models featuring larger local tracking ranges. An illustration of this relationship is depicted in Fig. 2.10 for Noise 1:15. While this observation might imply that extensive and costly hyperparameter tuning is necessary for the tracker module in case of lower quality instance segmentation results, in practice, these parameters can be readily and efficiently adjusted based on visual assessment of segmentation quality of the local tracker model and a global tracking sanity check on only a few consecutive frames. Conversely, the heightened significance of the accuracy of these parameters further underscores that while tracking may be straightforward in cases of good quality instance segmentation, achievable by nearly any model, in scenarios of poorer quality instance segmentation, a well-designed tracking architecture can play a pivotal role, substantially impacting both segmentation and tracking quality.

2.4.6 Comparative Datasets Evaluation

Using the five synthetic datasets described in Sec. 2.4.2, our yeast tracking architecture was trained on 80 synthesized videos for each type, each comprising 100 frames, and subsequently evaluated on 20 videos. The only modification applied to the architecture was the utilization of bounding box marking instead of centroid marking for the tracked objects. This adjustment was necessary due to the potential intersection of object paths and thus the possibility of occlusion, rendering centroid marking ambiguous in certain scenarios. The chosen TR value for each training was 4, as it provides a good balance between missing instance interpolation and prediction quality. The resulting segmentation and tracking F scores are displayed in Tab. 2.3, while sample prediction results are displayed in Fig. 2.12 with raw inputs for the given samples displayed in Fig. 2.11. These sample displays were chosen in an unbiased manner by always selecting the first training sample, regardless of prediction quality. However, the frames chosen for display were selected to best showcase the differences between the datasets.

These results clearly display the difficulty difference between objects with more predictable behaviors, such as yeast cells or synthetic arrows, and comparatively

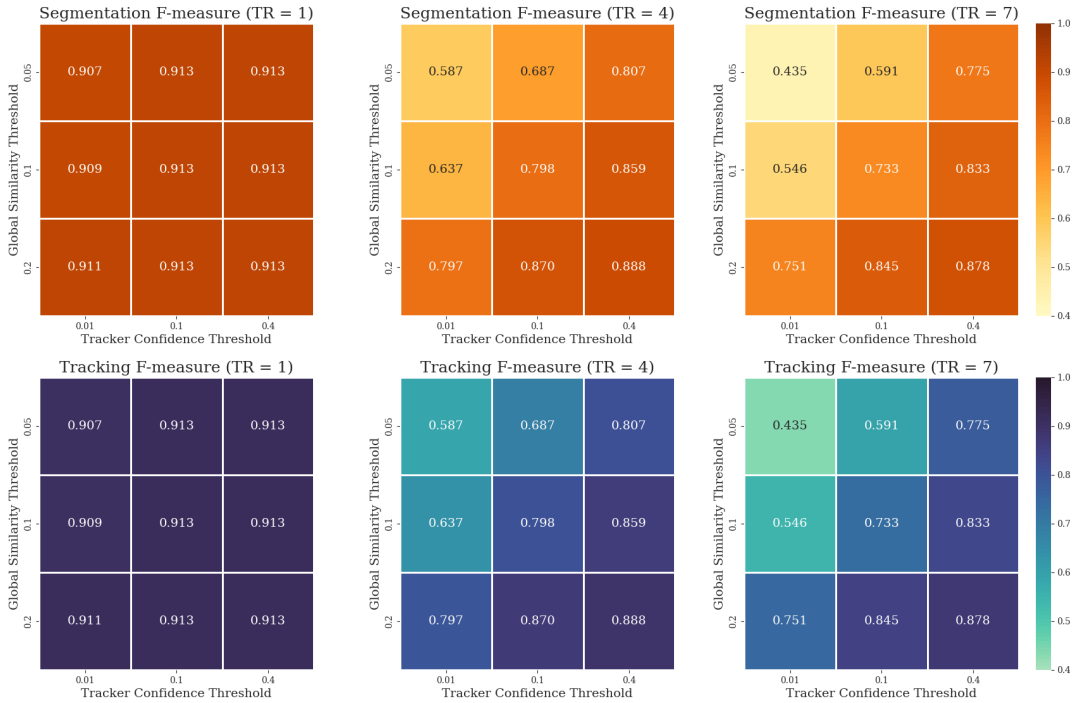


Figure 2.10: **Hyperparameter interdependence**
Illustration of the interdependence of hyperparameters: segmentation confidence threshold of the tracker model and minimal similarity threshold of the global assignment as a function of local tracking range (TR).

more challenging and unstable objects, such as the synthetic amoeboid versions. While the numerical results for amoeboids are lower compared to those for yeast cells or synthetic arrows, they are still acceptable given the difficulty of the tasks. The errors mostly arise from instance segmentation, as tracking F scores are never substantially lower than the segmentation F scores. Empirical assessment also shows that the predictions are mostly correct and would serve as a valuable baseline for later manual corrections if necessary. Furthermore, these results could be substantially improved through specialized augmentation techniques, hyperparameter tuning, using more complex backbone architectures for feature extraction, and increased training data. Therefore, based on our assessment, these results demonstrate the adaptability of our architecture in various object tracking scenarios with vastly different object morphologies and behaviors. However, they also highlight the different training requirements and expectations for different

	Segmentation F score	Tracking F score
S. Arrows	0.9185 ± 0.0057	0.8990 ± 0.0076
S. Amoeboids	0.7137 ± 0.0082	0.6605 ± 0.0093
S. Amoeboids-PC	0.6861 ± 0.0089	0.6662 ± 0.0098
S. Amoeboids-PCC	0.7726 ± 0.0101	0.7693 ± 0.0111
S. Amoeboids-PCCA	0.5078 ± 0.0129	0.5022 ± 0.0140
Yeast Reference	0.9234 ± 0.0136	0.9202 ± 0.0138

Table 2.3: **Synthetic sets numeric results**
Segmentation and Tracking F scores of the full pipeline for various synthetic datasets with vastly different object behaviors and challenges compared to natural yeast recordings.

datasets. A more detailed analysis of this aspect is described in Sec. 2.4.7.

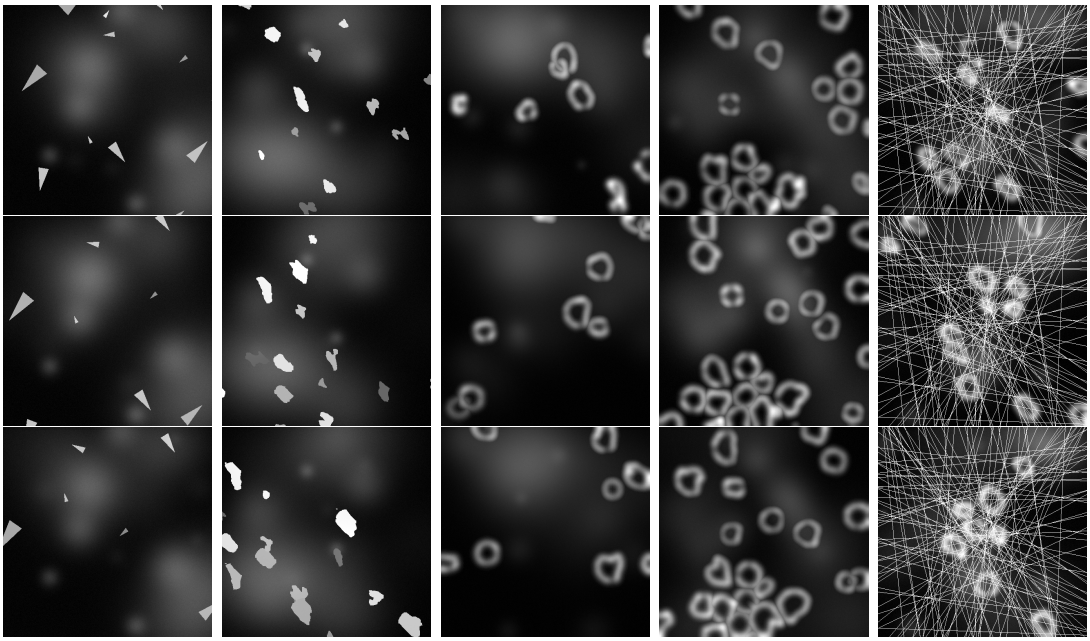


Figure 2.11: **Synthetic sets input samples**
A display of raw input samples on various synthetic datasets with highly different object behaviors, ordered from left to right as "S. Arrows," "S. Amoeboids," "S. Amoeboids-PC," "S. Amoeboids-PCC," and "S. Amoeboids-PCCA." The displayed consecutive images are only 4 frames apart to show understandable results even in cases of extremely fast-moving objects.

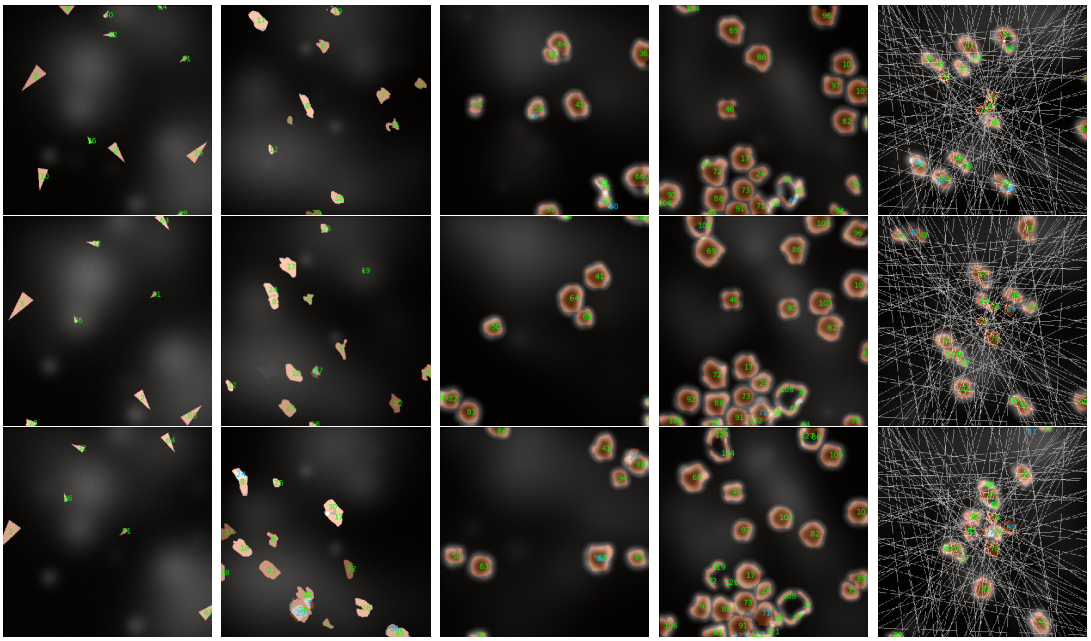


Figure 2.12: **Synthetic sets tracking sample results**
A display of segmentation and tracking results on various synthetic datasets with highly different object behaviors, ordered from left to right as "S. Arrows," "S. Amoeboids," "S. Amoeboids-PC," "S. Amoeboids-PCC," and "S. Amoeboids-PCCA." The displayed consecutive images are only 4 frames apart to show understandable results even in cases of extremely fast-moving objects.

2.4.7 Data Requirement Analysis

The training of neural network-based prediction models always requires a varied set of training samples, ideally covering all possible scenarios that may arise during inference. Furthermore, the number of sample points should be high enough to minimize dataset-specific learning, commonly referred to as overfitting to the training dataset. While both sample variety and the avoidance of overfitting can be improved through artificial data augmentation and other generalization techniques, there is a highly task-dependent limit for the minimal required amount of training data. Assessing this limit numerically before initial training and evaluation can be very challenging or nearly impossible. Thus, we provide example evaluation results using training sets with varying sample numbers as potential guidelines on the yeast tracking dataset, as well as the synthetic datasets described in Sec. 2.4.2.

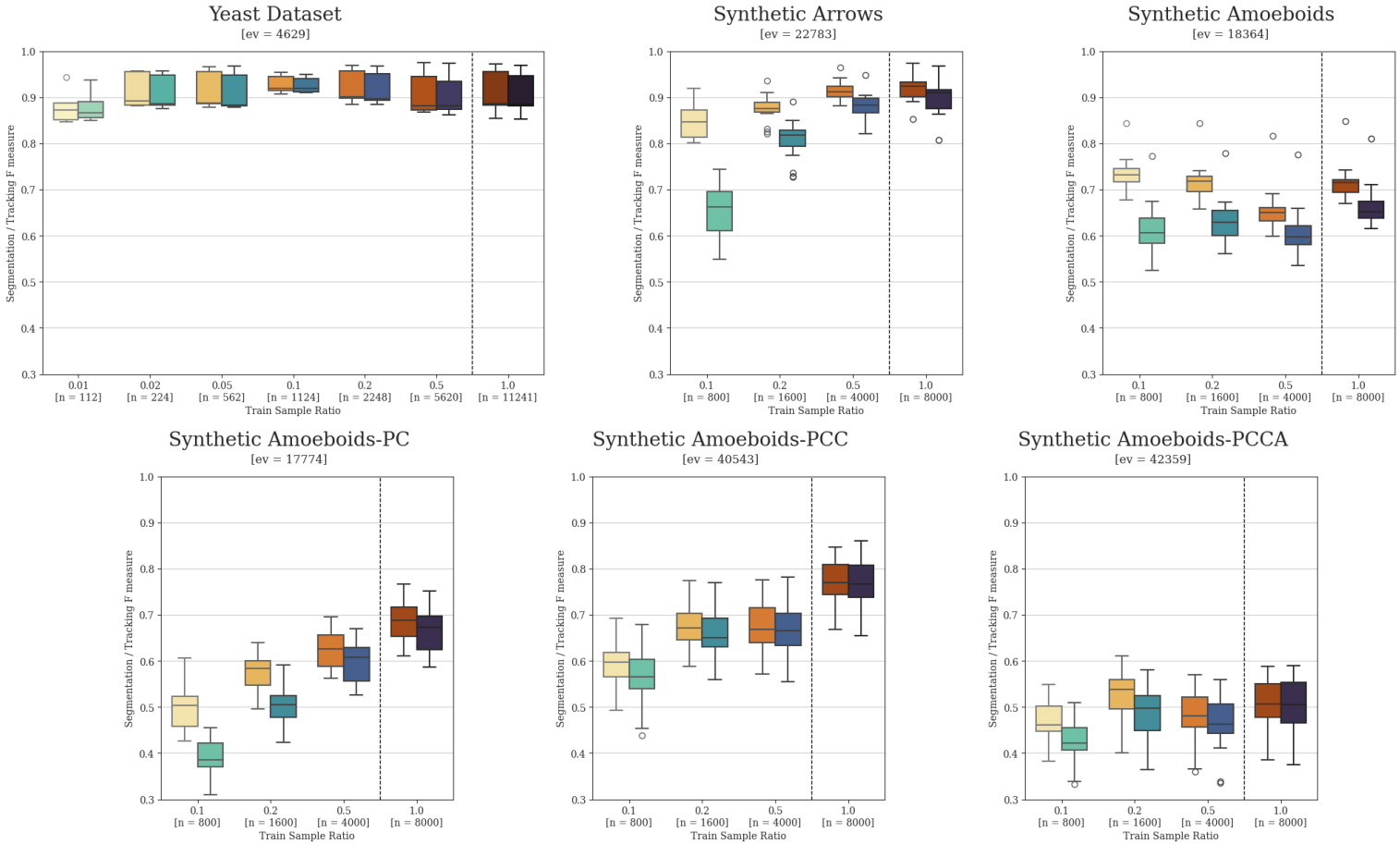


Figure 2.13: **Training data requirements**
A comparison of training data requirements and the impact of reduced sample sizes on yeast cell tracking and various synthetic datasets. The left boxes show segmentation F scores, while the right boxes show tracking F scores for each individually trained model pair. The dashed lines separate results obtained using the full training set reference models from those trained on randomly selected subsets. The x-axis upper number indicates the relative training set size ratio, while "n" represents the number of images in each set. The "ev" value denotes the number of object instances in each evaluation dataset contributing to the results.

The results using randomly selected subsets of the original dataset for the training of both instance segmentation and local tracking models are presented in Fig. 2.13. For the proper interpretation of these results, it must be noted that individual model trainings can vary to a certain degree due to the stochastic nature of the training process, which is especially true for datasets with lower sample sizes. Nevertheless, some clear patterns are still noticeable. Based on these, it appears that yeast cells are by far the easiest of these objects to segment and track, as only 2% of the 314, approximately 35.8-frame-long videos can lead to nearly identical results as training on the full dataset. Conversely, the various examined synthetic datasets showed varied results, clearly displaying how challenging the unique features of the given dataset are.

2.5 Discussion

To display the capabilities and characteristics of our architecture, we performed a comparative evaluation against other tools specifically designed for budding yeast cell segmentation and tracking. Additionally, we analyzed the hyperparameter sensitivity of our architecture. We also measured the reconstructive capabilities of the tracker module in cases of lower quality segmentation to demonstrate the robustness of the architecture as a whole. Furthermore, we evaluated the architecture on various synthetic datasets to display its retrainability and to provide guidelines for the difficulty of particular visual and motility patterns. Finally, we analyzed the training data requirements of the architecture on the available natural and synthetic datasets to aid potential future applications that require the retraining of the models.

Our architecture demonstrated outstanding state-of-the-art performance compared to other evaluated yeast segmentation and tracking tools in terms of both instance segmentation and tracking results, substantially outperforming Phylocell and YeaZ. Remarkably, our architecture even outperformed Phylocell in the scenario where border tracks were not removed, a condition that is biased towards Phylocell.

The results of the hyperparameter dependency analysis of our architecture indicate that shorter local tracking ranges are slightly preferred if segmentation

results are stable, as training less complex models is easier in terms of data intensity. However, as the tracking robustness analysis shows, longer local tracking ranges are preferred if the detection or segmentation of objects is imperfect, especially if objects are lost for multiple consecutive frames. In more difficult datasets, such scenarios can easily occur naturally due to object occlusion and various visual artifacts. It should be noted that, according to our results, the acceptance threshold values of the tracking architecture must be tuned more precisely for larger local tracking ranges.

Furthermore, the hyperparameter dependency analysis reveals that low-complexity encoding backbones of the local tracker model are sufficient for tracking yeast cells, making the model lightweight in terms of computational requirements, although GPU-accelerated inference is still highly recommended. Additionally, the analysis of the global assignment step shows that Intersection over Union (IoU) is the preferred metric for prediction similarity based local track matching.

Lastly, the analysis of performance on synthetic datasets, as well as the analysis of training data requirements on both synthetic and yeast datasets, showcases the general versatility of the architecture if a sufficient amount of annotated training data is available. As our architecture contains re-trainable models as predictors, these results show promise for applying the same architecture for other cell types after retraining on appropriate data. The results also indicate that all the analyzed synthetic sets are more challenging than yeast cells, but mostly valid predictions are still achievable on most of them with comparatively larger amounts of training data. Furthermore, we believe that while the analyzed synthetic sets differ from reality in many aspects, their characteristic features and the associated training data requirements and metric results can serve as a guide for the retraining of the models in terms of data requirements and performance expectations.

2.6 Data and Code Availability

Various data resources, as well as the segmentation and tracking pipeline, have been made openly accessible through a dedicated GitHub repository: `https://github.com/SzaboGergely0419/PPCU_IFOM_YeastTracker`. This repository includes supplementary resources such as different versions of the tracker model,

sample test data, and the complete set of training and test sets used for yeast tracking. Sample prediction results are uploaded directly to the repository. Additionally, a comprehensive Google Colab environment is provided within the repository, allowing for the execution of the entire pipeline.

Chapter 3

General time-symmetric tracking

3.1 Author Contributions

The concepts and results presented in this chapter were the outcome of a partial collaboration between Zsófia Molnár, András Horváth, and me. While I was responsible for the conceptual and performance-related redesign of the architecture, training of the models, creation of prediction results, and writing of the manuscript, the evaluation framework was a joint effort. Zsófia Molnár and I worked closely on designing this framework, with her playing a key role, particularly in the conceptualization of the metrics. The design and creation of the figures in this chapter were also a collaborative effort between us. András Horváth provided essential theoretical insights, critiques and guidance that further shaped the work.

3.2 Introduction

Detection, segmentation, and tracking of multiple objects in image sequences remain challenging tasks, requiring the optimization of numerous factors during the design of new methods. While most state-of-the-art tracking systems prioritize low latency for real-time applications such as self-driving cars and surveillance cameras, this emphasis on speed often comes at the expense of incorporating potentially valuable information and generalization capabilities, especially when inference time is not a limiting factor. As discussed in Chapter 2, a novel tracking architecture, termed the *"TS"* architecture due to its time-symmetric tracking approach, was introduced specifically for the task of tracking living cells in videomicroscopic

recordings. This architecture showed excellent performance in handling object assignment challenges during cell division and maintaining continuous tracking due to its reconstructive capabilities.

Although Chapter 2 presented an in-depth analysis of the *TS* architecture's performance in tracking budding yeast cells and other synthetic cell-like objects, its potential in other substantially different environments was not fully explored. Therefore, in this chapter, the focus is extended to a broader evaluation of the *TS* architecture using standardized metrics and comparative analyses against traditional approaches such as the Kalman filter [11]. Additionally, the architecture's capabilities are further examined in highly specific synthetic scenarios and in the challenging zero-shot knowledge transfer scenario between the synthetic MOTSynth-MOTS-CVPR22 training dataset [7] and the real-world MOTS dataset [50] for person tracking. By systematically extending the investigation of this architecture's performance to diverse tracking contexts, this chapter aims to highlight its flexibility and potential for broader applications beyond cell tracking.

All related codes, data, and models are available at `https://drive.google.com/drive/u/2/folders/1JbCJT4DMnzMIchqqC1IcKRjmYA0s5-IZ`, frozen at the time of publication. The updated and further maintained codebase can be accessed through the following GitHub repository: `https://github.com/SzaboGergely0419/Symmetry-Tracker`.

3.3 Architecture overview

Although the *TS* architecture, which serves as the foundation for the analyses presented in this chapter, is nearly identical to the version detailed in Chapter 2, several minor theoretical modifications were made to enhance its capabilities. Additionally, the practical implementation was completely restructured to accommodate the substantially increased computational requirements of the subsequent evaluation scenarios. For a macroscopic overview of the architecture, a high-level data flow illustration is presented in Fig. 3.1.

While the original implementation provides a pipeline implementation, it contains some inefficiencies, resulting in longer-than-necessary inference times and memory overflow errors in longer sequences. Since some of the data we intended

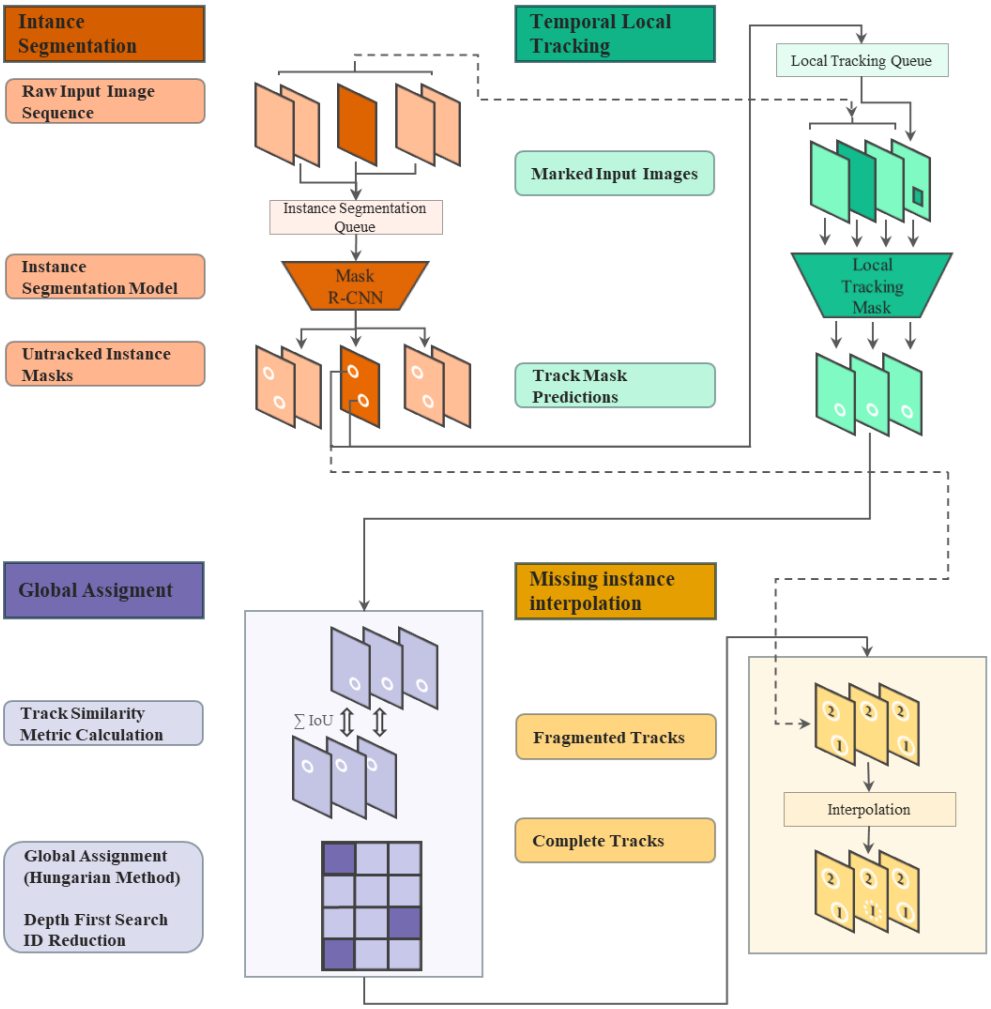


Figure 3.1: **Macro-architecture overview**
Data flow diagram of the *TS* architecture, illustrating the process from raw input image sequence to finalized track predictions.

to analyze — specifically the MOTS sequences — are particularly lengthy, we had to completely refactor and optimize the tracking architecture, while keeping the Detectron 2 [32] and SMP [51] based models intact. The updated pipeline implementation further modularizes the tracking segment by breaking it down into fully independent sub-tasks of data preparation, local tracking, global tracking and graph based ID reduction, employs preparatory calculations to minimize runtime in longer sequences, and mitigates memory usage inflation over time. A comparison of memory usage and runtime between the original and updated *TS* architecture is

presented in Fig. 3.2. These results were measured using the five yeast recordings provided through the demonstration environment of the original implementation [Ar1], which are relatively short and do not cause that implementation to crash. While the runtime benefits of the preparatory calculations are not evident in these short sequences, the stark reduction in memory usage inflation in the tracker segment is apparent. The improved version of the full pipeline, frozen at the time of publication, is accessible via the URL provided in Sec. 3.2.

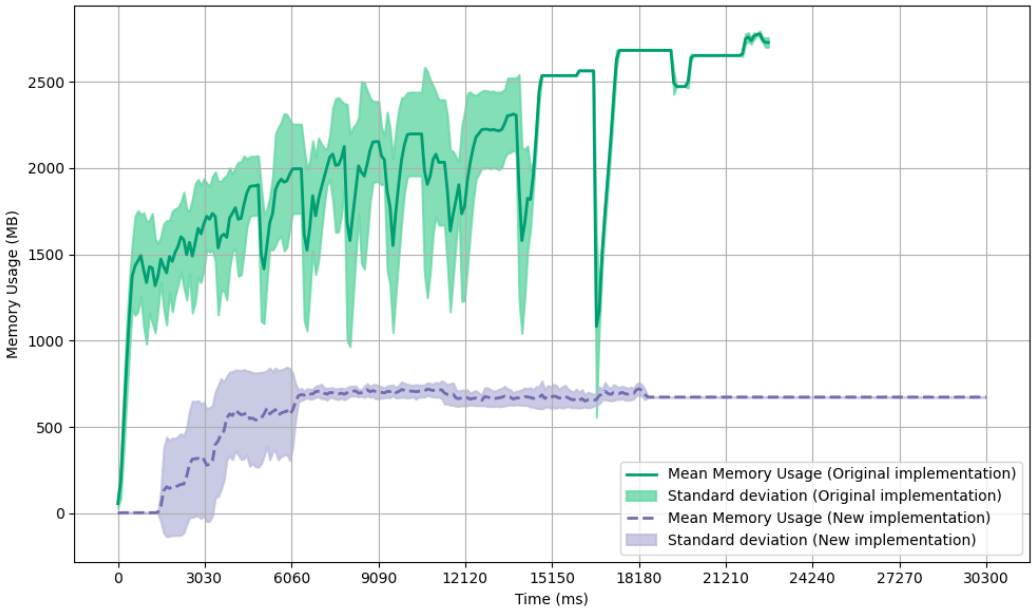


Figure 3.2: **Memory usage comparison**
A display of memory usage and runtime differences between the original implementation of the *TS* architecture tracking segment and the improved implementation.

3.4 Metric definitions

To evaluate the overall performance of the architecture and specifically highlight details of tracking performance, we employed two families of metrics. The first family is based on comparisons of Intersection over Union (IoU) scores, using a 50% matching threshold for binary acceptance or rejection, making the interpretation of results straightforward. The second family includes the widely used HOTA scores, along with associated DetA and AssA scores.[52]

3.4.1 IoU 50% scores

The scores in this family are heavily inspired by the F-scores used in Chapter 2. However, the tracking F-score used there combines detection and association errors, making it heavily dependent on detection performance. While this approach is valid, we believe that more easily interpretable results can be obtained by focusing solely on association, where detection was successful, akin to the calculation of HOTA. Here, we will define only the association scores, as our primary interest lies in a detailed analysis of tracking performance. For a more general analysis, please refer to the HOTA metric family.

Let $GT(t,n)$ represent the n-th binary ground truth mask at time t and $PD(t,m)$ the m-th binary predicted mask at time t. We define the IoU_{50} binary metric as:

$$IoU(GT(t,n),PD(t,m)) = \frac{|GT(t,n) \cap PD(t,m)|}{|GT(t,n) \cup PD(t,m)|} \tag{3.1}$$

$$IoU_{50}(GT(t,n),PD(t,m)) \quad = \quad \begin{cases} 1 & \text{if } IoU(GT(t,n),PD(t,m)) > 0.5 \\ 0 & \text{if } IoU(GT(t,n),PD(t,m)) \leq 0.5 \end{cases} \tag{3.2}$$

Thus, we can define a successful object detection in a simplified way without needing optimal assignment: if any GT object has a match based on the IoU_{50} metric to any PD object on the same frame, it will be a unique match due to the matching threshold of 50%, as only a single object can occupy a single image point. Furthermore, if $|GTD_{50}(t,t+1)|$ is the number of ground truth object IDs mutually present at time t and $t+1$ where detection was successful at time t, and $|PDD_{50}(t,t+1)|$ is the number of predicted object IDs mutually present at time t and $t+1$ where detection was successful at time t, we can define the true positive association count TPA_{50}, the false positive association count FPA_{50}, and the false negative association count FNA_{50} as follows:

$$\begin{aligned} TPA_{50}(t,t+1) = &\sum_{n,m} IoU_{50}(GT(t,n),PD(t,m)) \\ &\cap IoU_{50}(GT(t+1,n),PD(t+1,m)) \end{aligned} \tag{3.3}$$

$$FPA_{50}(t,t+1) = |PDD_{50}(t,t+1)| - TPA_{50}(t,t+1) \tag{3.4}$$

$$FNA_{50}(t, t+1) = |GTD_{50}(t, t+1)| - TPA_{50}(t, t+1) \tag{3.5}$$

Based on these values, association precision AP_{50}, association recall AR_{50}, and association F-scores AF_{50} can be calculated for the temporal position $(t, t+1)$ as:

$$AP_{50} = TPA_{50}/(TPA_{50} + FPA_{50}) \tag{3.6}$$

$$AR_{50} = TPA_{50}/(TPA_{50} + FNA_{50}) \tag{3.7}$$

$$AF_{50} = \frac{2AP_{50}AR_{50}}{AP_{50} + AR_{50}} \tag{3.8}$$

While precision, recall, and F-scores are typically presented in the range of $[0, 1]$, we will present them multiplied by 100 to align with the value range of the HOTA metric family.

3.4.2 HOTA metric family

For a detailed description of the HOTA metric family, please refer to the 2020 paper by Luiten *et al.* [52]. In summary, unlike the previously widely used MOTA and IDF1 scores, HOTA effectively balances the importance of both association and detection, making it an excellent metric for evaluating the overall performance of a tracking system. Additionally, HOTA can be decomposed into detection accuracy (DetA) and assignment accuracy (AssA) scores. While our primary focus is on the performance of the tracking model, the overall performance of the entire system is also crucial for interpreting the tracking results. Therefore, DetA, AssA and HOTA scores will be all presented in the analysis. Unlike the segmentation IoU-based metrics defined in Sec. 3.4.1, these scores are computed for the bounding boxes of each object instance to align better with comparisons to other tracking models that typically predict only bounding boxes, not segmentations. Hence, we advise interpreting the IoU 50% scores as the primary indicators of model performance, with the HOTA metric family serving as a comparative measure that may obscure some of the true potential of the segmentation-based tracker.

3.5 Tracker variants

To evaluate the tracking capabilities of the *TS* architecture, we compared it with the widely used Kalman filter [11], a default choice in many object tracking environments. Although advancements have made the mathematical limitations regarding linearity and parameterization of the Kalman filter more flexible [20, 21, 53], the core concept remains the same: tracking objects based on a limited set of parameters such as position and its derivatives, while the Kalman filter optimally balances the estimated temporal forward prediction between past positions, updated according to a state transition model, and new object detections, transformed by an observation model. Although this temporal forward prediction approach is computationally lightweight, making it ideal for live object tracking scenarios, it disregards morphological information and other input data details not represented by the observation model. Moreover, temporal forward predictions must be assigned to observations at different temporal positions to perform tracking, which involves assigning objects in different state spaces. While this might seem sound at first glance, the similarity measurement is highly constrained by the state transition and observation models, limiting the information that can be compared between observed and forward-predicted objects. This often defaults to an oversimplified metric, such as L2 distance between centroid positions. Additionally, any temporal forward prediction model ignores future data, which is reasonable for live tracking scenarios but overlooks potential information when the entire temporal sequence is available at the time of prediction.

In contrast, the *TS* architecture's local tracking segment inherently learns temporally local behaviors and predicts segmented masks using both past and future data. These predictions are compared in the same state space, as they are made by the same model from different perspectives. We believe that the difference between temporally forward-predicting models and the symmetric tracking capabilities of the *TS* architecture is somewhat analogous to a comparison between the Forward algorithm [54] and the Viterbi algorithm [55], although the temporally symmetric parallel predictions and the updating of past predictions based on new information in the Viterbi algorithm are conceptually distinct. On the other hand, while it is natural for the *TS* architecture to skip missed object

instances and re-interpolate them after tracking, temporal forward predictors like the Kalman filter can also skip missed instances if the state transition model is applied multiple times after a matching detection is not found based on the matching criterion. In our implementation of the Kalman filter-based tracker, we allowed temporal forward predictions and re-interpolations with a maximum of 8 frames distance, matching the local tracker models' TR value of 4, resulting in a maximum assignment distance of $2TR = 8$.

To further assess the impact of positional and morphological information in assignments, we evaluated two restricted variants of the *TS* architecture: *TS-L2* and *TS-Shape*. The *TS-L2* variant uses the same local tracking model but retains only the centroids of the predicted masks for L2 distance-based similarity comparison, ignoring all morphological information. The *TS-Shape* variant aligns the centroids of predicted masks before calculating IoU-based similarity, focusing solely on morphology and disregarding positional data. Evaluating the *TS-L2* variant is particularly interesting, as it serves as a middle ground between the Kalman filter and the *TS* architecture by using visual cues from inputs while ignoring morphology during assignment.

Despite the substantial differences in prediction methodologies among these four models, their predictions can be handled similarly. We applied the unaltered Hungarian algorithm-based global assignment step of the *TS* architecture, followed by depth-first search of connected IDs and interpolation of missed instances to all predictions. Furthermore it must be noted that following track prediction, we omitted any tracks shorter than 10 frames in length. While this introduces a reverse dependence from tracking quality to detection quality, it also adds a sense of realism to the tests, as in autonomous systems such low confidence detections are often omitted too.

3.6 Datasets

While evaluating novel methods on natural datasets is crucial, such datasets for MOTS (Multi-Object Tracking and Segmentation) tasks are relatively rare, especially those with ample training data where all objects are accurately labeled and tracked. For instance, the recently released SA-V dataset, allegedly used

to train the SAM 2 model, provides a large number of high-quality tracks but only for a few objects per recording [56]. This limitation makes it unsuitable for training the instance segmentation stage of the evaluated architectures and almost completely prevents the assessment of the recall capabilities. Moreover, the *TS* architecture has already been proven to outperform other methods that were specifically designed for budding yeast cell detection and tracking on a natural dataset of such videomicroscopic recordings.

Therefore, we opted to create various synthetic scenarios, building on the synthetic datasets presented in Chapter 2, to evaluate specific performance differences among the four tracking models described in Sec. 3.5. The code used to generate these scenarios, along with the resulting datasets, is available at the URL provided in the abstract. Although these scenarios are artificial, their aim is to simulate key features and events commonly observed in natural settings. Additionally, to ensure evaluation on natural datasets, we trained models on the synthetic MOTSynth-MOTS-CVPR22 dataset [7] and then evaluated their performance on real-world samples from the MOTS dataset [50].

3.6.1 Visual signaling scenario

For the foundation of this synthetic dataset group, we used the *Synthetic Arrows* scenario defined in Sec. 2.4.2. This baseline scenario was selected for its simplicity: the objects move quickly with near-linear motion characteristics, and there is minimal visual information to be gained from the objects' morphological features, except for the indication of their forward direction. Consequently, we anticipated the Kalman filter to perform well in this scenario, providing a benchmark for evaluating the *TS* architecture variants.

In contrast, the modified versions of the *Synthetic Arrows* dataset that we created were expected to be more challenging. In these scenarios, the arrows undergo a specific color change before a turning event, similar to how cars use turn signals, indicating the direction in which the arrow will turn after a number of T frames. The arrow then executes a 90°turn in the signaled direction. The first variant, *Synthetic Arrows TR-1*, initiates a turn with a 20% chance per frame, with a signaling period of $T = 4$ frames. The second variant, *Synthetic Arrows*

TR-2, initiates a turn with an 80% chance per frame, with a signaling period of $T = 2$ frames. These scenarios were designed to test how well the *TS* variants interpret visual cues and to assess the extent to which visually signaled kinematic events disrupt the Kalman filter.

3.6.2 Semi-random positioning scenario

Similarly to the "Visual Signaling" scenario, we also used the *Synthetic Amoeboids* scenario defined in Sec. 2.4.2 as a baseline. However, unlike the original *Synthetic Amoeboids* scenario, we applied no morphological changes to the objects from one frame to the next, making them even more recognizable based on their morphological features. Unlike the *Synthetic Arrows* scenario, the amoeboids possess unique morphological characteristics due to the Perlin noise [46] applied in their generation process. While these objects are still relatively easy to track, we anticipated that disregarding morphological cues would be disadvantageous for both the Kalman filter and the *TS-L2* model, even for this baseline scenario, especially when compared to *Synthetic Arrows*.

The modified versions of the *Synthetic Amoeboids* baseline dataset that we created were also expected to be increasingly challenging. While in these variants, the objects still exhibit semi-random but almost linear movement patterns, excluding collisions, their final position in each frame is adjusted by a random uniform positional noise at a maximum distance of $1/D$ relative to the field of view in both the x and y directions. As $1/D$ increases, this makes object tracking solely based on position progressively more difficult, thereby raising the importance of object morphology. The two variants of this scenario are *Synthetic Amoeboids RP-1/20* and *Synthetic Amoeboids RP-1/5*, with random positioning relative distances of 1/20 and 1/5, respectively. We expect these variants to be particularly challenging for tracking models that ignore morphological information.

3.6.3 MOTS challenge

The MOTSynth-MOTS-CVPR22 training dataset includes 767 full HD videos, each 1,800 frames long, generated within the computer game GTA V [7], with pedestrians annotated as objects. The official test set consists of seven naturally

captured and manually annotated samples. However, because the challenge's evaluation kit has been outdated for some time, we opted instead to use three publicly released full HD resolution sequences from the MOTS challenge training set as our test set, available at `https://motchallenge.net/data/MOTS/` (2024). Our models were trained on the first 600 recordings of the training set, with the remaining training samples used for evaluation. Notably, we were unable to perform deep fine-tuning of model hyperparameters on the training dataset either, due to the immense computational requirements and training times. Thus, we opted to use the original augmentation scheme and hyperparameters of the *TS* model, with a local tracking range of 2. Therefore, it is likely that, with a more specialized model, even better results could be achieved.

To the best of our knowledge, the only official submission to the MOTSynth-MOTS-CVPR22 challenge is based on the widely recognized "Tracking without Bells and Whistles" (*Tracktor*) model by Bergmann *et al.* [57]. Although only this single submission exists, it employs a popular state-of-the-art tracking method, making its performance the most reliable benchmark for comparison in our evaluation.

3.7 Evaluation results

We conducted a comprehensive evaluation of the tracker models *Kalman* filter, *TS*, *TS-L2*, and *TS-Shape* using the metrics defined in Sec. 3.4 across both synthetic scenarios and the MOTS challenge variant outlined in Sec. 3.6. The evaluation results are presented as Kernel Density Estimation (KDE) distribution estimates [58] for the AssA and HOTA metrics, and as mean values for all other metrics.

For proper interpretation of the results, it is important to preliminarily note that an observable increase in DetA and HOTA scores often coincides with particularly low AssA and especially AR_{50} scores. While these elevated DetA and HOTA values are technically correct, they primarily result from the exclusion of unreasonably short tracks, which removes lower-confidence detection instances and artificially boosts detection precision, thereby inflating DetA and HOTA scores. Therefore, as our focus is primarily on tracking performance, we caution against drawing far-reaching conclusions from these inflated DetA and HOTA values.

3.7.1 Synthetic scenarios

To establish a baseline for the other scenarios, we first present the comparative evaluation results between the *Synthetic Arrows* and *Synthetic Amoeboids* datasets in Fig. 3.3. While the models *Kalman*, *TS*, and *TS-L2* demonstrated similar performance in the simplistic *Synthetic Arrows* scenario, the *TS* architecture clearly outperformed both the Kalman filter and the *TS-L2* model in the morphologically more complex *Synthetic Amoeboids* scenario. This result highlights the potential benefit of incorporating morphological information. However, the *TS-Shape* model substantially underperformed on both datasets, particularly in the *Synthetic Amoeboids* scenario, indicating that positional information alone is more valuable in these scenarios than morphological information, and that the combined benefit of both types of information is not merely additive. Notably, the comparatively better performance of the *TS-Shape* model on the *Synthetic Arrows* is likely due to the arrows being often accurately identifiable by their specific area, whereas different amoeboids might share similar surface features by chance.

Next, we present the results for the "Visual signaling" scenario defined in Sec. 3.6.1 in Fig. 3.4. As anticipated, the Kalman filter's performance deteriorates as the movement patterns of the objects become more dependent on visual signals. In contrast, the *TS* and *TS-L2* architectures show a lesser decline in performance, with the difference being clearly noticeable but relatively modest. Notably, the *TS-L2* architecture performs similarly to the *TS* architecture, as the visual signals are encoded within the architecture and positional estimates, even though the assignment metric in the architecture disregards morphological information. The *TS-Shape* architecture continues to perform the worst, as there is minimal morphological information available to differentiate the objects.

Lastly, we present the results for the "Semi-Random Positioning" scenario defined in Sec. 3.6.2 in Fig. 3.5. Here, the *TS* architecture shows a clear advantage over both the Kalman filter and the *TS-L2* architecture, emphasizing the benefit of predicting full object morphologies instead of relying solely on positional assignments. While the *TS-Shape* architecture still performs far worse than the others, its performance remains relatively stable across the scenario, as it is unaffected by the position of the objects.

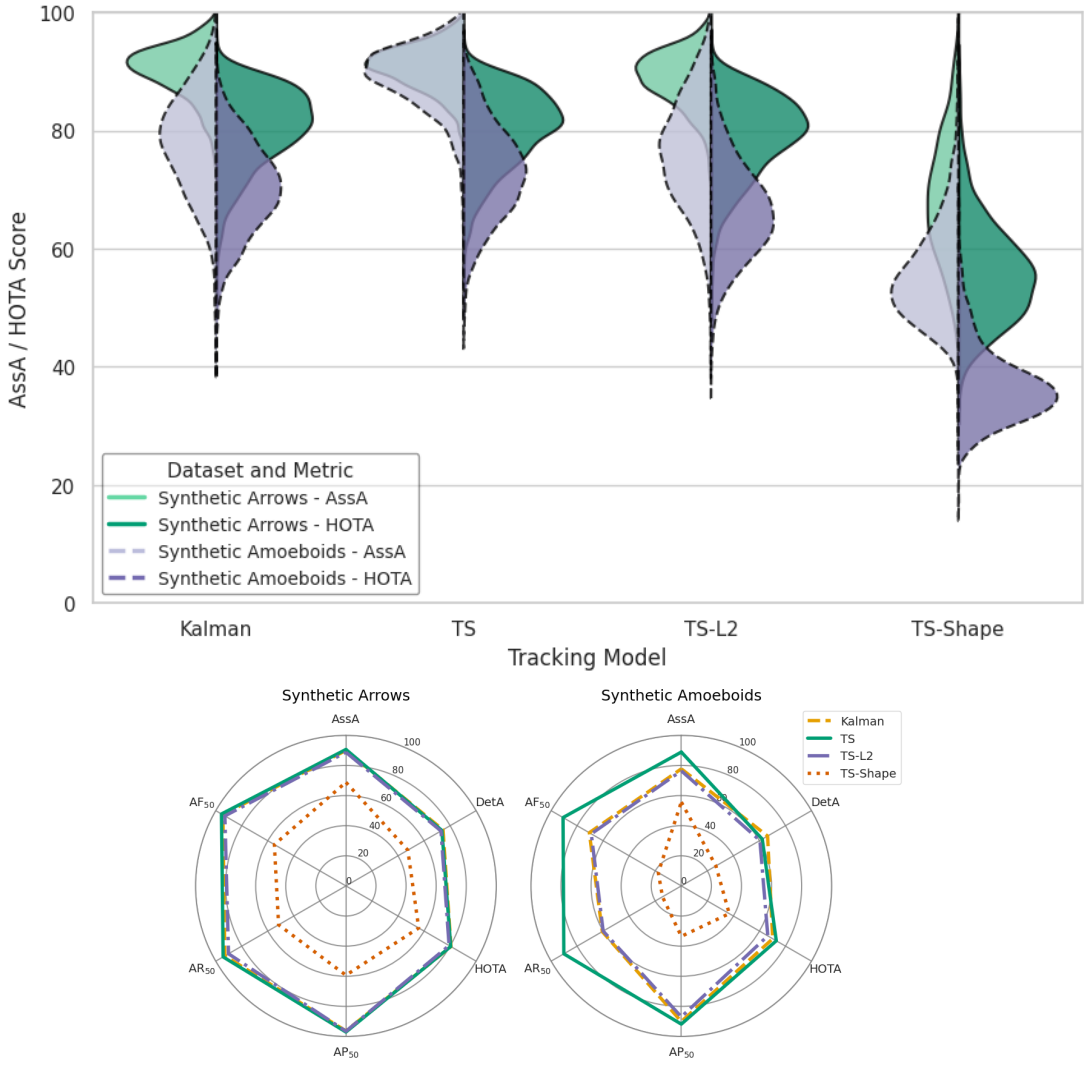


Figure 3.3: **Baseline metric results**
KDE (top) and mean (bottom) metric results of tracker models *Kalman*, *TS*, *TS-L2* and *TS-Shape* for datasets *Synthetic Arrows* and *Synthetic Amoeboids*.

3.7.2 MOTS challenge

Similarly to the synthetic scenarios, we present the results for the MOTS challenge samples described in Sec. 3.6.3 in Fig. 3.6. Due to the particularly challenging nature of this task, the object detection score is low. Still, the *TS* model performs notably better than the other models. Interestingly, while the *TS-Shape* architecture still has the lowest performance overall, it performs much better than in the

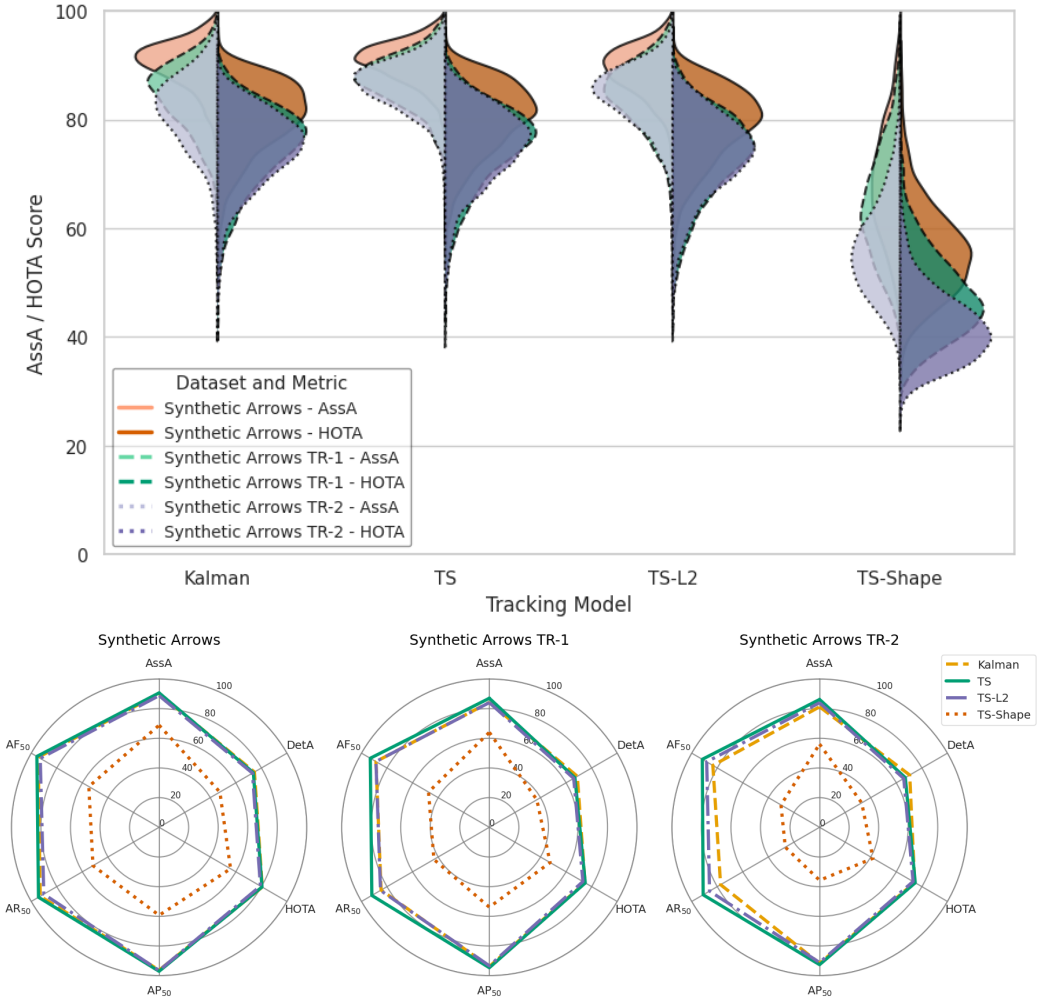


Figure 3.4: **"Visual signaling" metric results**
KDE (top) and mean (bottom) metric results of tracker models *Kalman*, *TS* and *TS-L2*, *TS-Shape* for scenario "Visual signaling" datasets *Synthetic Arrows*, *Synthetic Arrows TR-1* and *Synthetic Arrows TR-2*.

synthetic scenarios. We believe this can be attributed partially to the particularly clear visual differences between the pedestrian objects to be tracked.

The overall performance of the *TS* architecture, with a mean HOTA score of 48.56, closely matches the HOTA score of 48.8 achieved by the benchmark *Tracktor* model. Moreover, the *TS* architecture achieves a substantially higher mean AssA score of 82.39, compared to the *Tracktor* model's AssA score of 44.6. This suggests that the comparable HOTA scores are primarily due to imperfect detection and

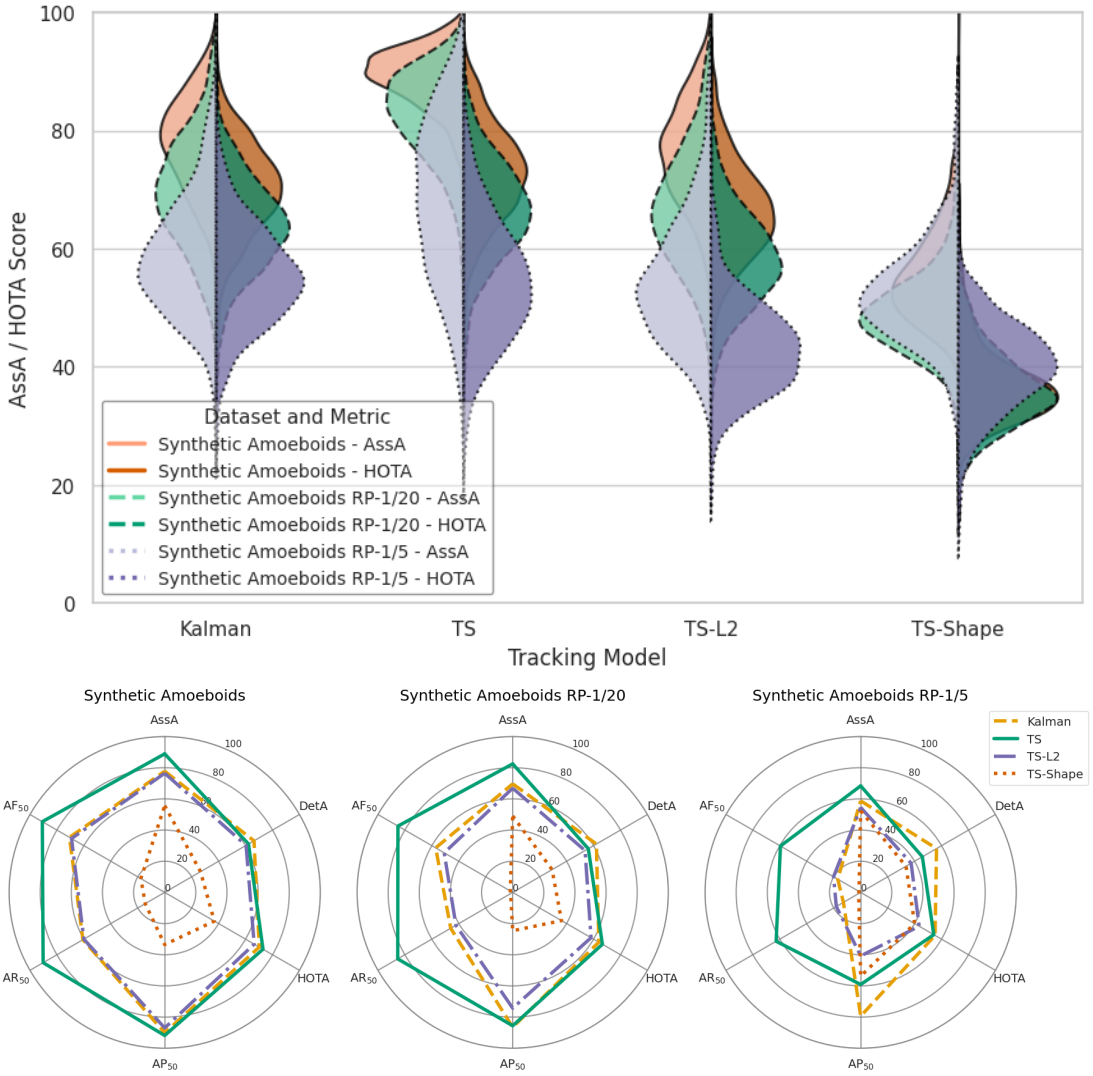


Figure 3.5: **"Semi-random positioning" metric results**
KDE (top) and mean (bottom) metric results of tracker models *Kalman*, *TS*, *TS-L2* and *TS-Shape* for scenario "Semi-random positioning" datasets *Synthetic Amoeboids*, *Synthetic Amoeboids RP-1/20* and *Synthetic Amoeboids RP-1/5*.

segmentation by the Detectron 2 based Mask R-CNN instance segmentation step within the *TS* architecture, while its novel tracking approach far outperforms the *Tracktor* model. Furthermore, since the Mask R-CNN with a ResNet-X feature pyramid backbone used by the *TS* architecture is widely recognized as one of the top-performing instance segmentation models [59, 60, 61], it is highly likely that

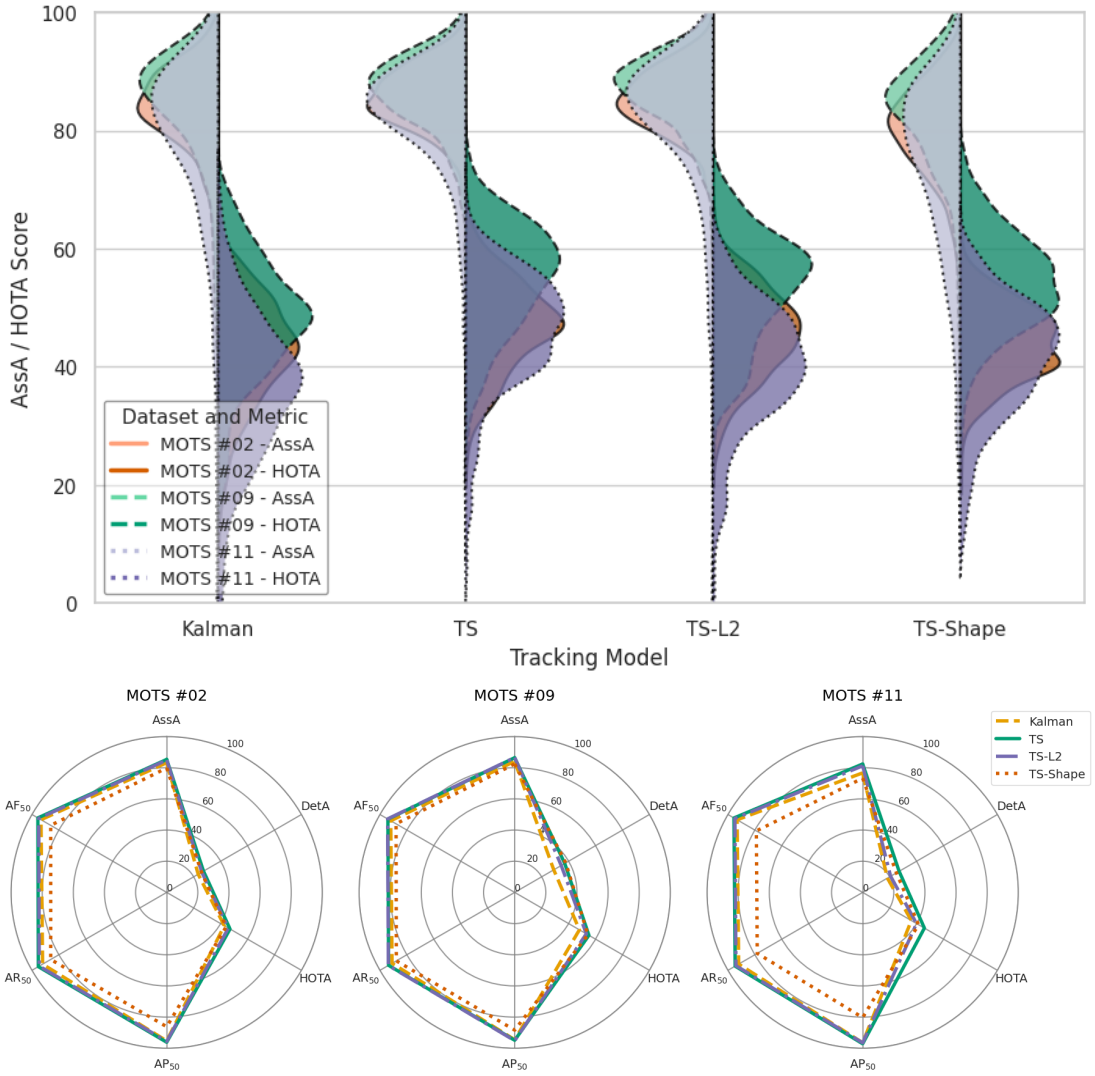


Figure 3.6: **MOTS challenge metric results**
KDE (top) and mean (bottom) metric results of tracker models *Kalman*, *TS*, *TS-L2* and *TS-Shape* for the MOTS dataset samples described in Sec. 3.6.3

with a more specialized training scheme of the instance segmentation model — including dataset-specific data augmentation and hyperparameter tuning — a substantially higher HOTA score could be achieved.

3.8 Discussion

Even from the baseline comparison of the *Synthetic Arrows* and *Synthetic Amoeboids* datasets, the clear advantage of the *TS* tracking architecture over the Kalman filter is evident. The *TS* architecture's ability to process multimodal information — including positional, morphological, and other image characteristics — rather than just kinematic data, proves beneficial when such information is relevant. Furthermore, the results suggest that the optimal integration of positional and morphological information for track assignment far outperforms the use of either information alone.

In the "Visual Signaling" scenario, the results reveal that incorporating movement pattern forecasting based on visual signals can enhance movement prediction, even when morphological information is not used in the track assignment step. This is demonstrated by the increasingly superior performance of the *TS-L2* model compared to the Kalman filter as the frequency of color-signaled motility events increases. This highlights the importance of visual cues in multi-object tracking, such as signal lights, turn signals, head movements, and even subtle gait or movement changes often observed in real-world environments. Notably, while human perception and real-time tracking models can only utilize past and present data, it is possible — and likely — that visual cues appearing in the future can aid in predicting motion that occurred before those cues. Processing these cues at inference time can be advantageous, and the *TS* architecture is uniquely capable of capturing and utilizing this information.

In the "Semi-random Positioning" scenario, where object movement is less predictable based on past positions and morphological information becomes crucial, the *TS* architecture shows clear superiority over the Kalman filter. Furthermore, in the *Synthetic Amoeboids RP-1/5* dataset, the performances of the *TS-L2* and *TS-Shape* models nearly balance out, demonstrating the increased importance of morphological information when positional estimates are unstable or when lower temporal sampling rates lead to less consistent movement patterns.

On the MOTS dataset, the *TS* architecture achieved a HOTA score comparable to that of the benchmark *Tracktor* model, reflecting similar overall performance.

However, the associative tracking performance of the *TS* architecture far surpassed the *Tracktor* benchmark. This strongly suggests that the *TS* architecture offers superior tracking capabilities, and with more dataset-specific training and optimization of the instance segmentation component within the *TS* architecture, substantially higher overall tracking results could likely be achieved on the MOTSynth-MOTS-CVPR22 dataset. Moreover, this performance highlights the *TS* architecture's potential applicability beyond videomicroscopic cell tracking to offline personnel tracking tasks such as surveillance, crowd movement analysis, and other related applications. Additionally, the *TS* tracking architecture's ability to deliver state-of-the-art performance across vastly different tasks strongly indicates its suitability for a wide range of tracking applications, provided its lower inference speed is acceptable in exchange for predictive performance.

Chapter 4

Object centering bias

4.1 Introduction

Evaluation and validation of machine learning tools in the image processing field are frequently conducted on large, publicly available datasets such as MNIST [13], CIFAR-10 [62], ImageNet [63], MS-COCO [14], and many others. The data distribution in these datasets has been thoroughly examined from many perspectives. However, the interesting, non-background pixels are usually concentrated in the middle of the images, and objects rarely appear at the edges. In contrast, for certain applications — such as collision prevention in self-driving cars — objects appearing at the edges of the images might be especially critical.

While the architecture of convolutional networks suggests that they are shift-invariant, this common assumption is not fully correct. It has been demonstrated in the 2019 paper by R. Zhang [64] that these structures are dependent on spatial shifting.

In this chapter of the thesis, extending on the 2020 results of O. S. Keyhan and J. C. Gemert [65] and the 2021 results of Md. Am. Islam *et al.* [66] I present evidence, that convolutional neural networks (CNNs) are capable of learning non translation equivariant behaviors — which appear at the edges of the image —, thus training and validating on the previously mentioned data sets might provide biased results and the trained CNNs might not reach the expected performance in realistic circumstances where objects may appear near the edges of the images. Furthermore we propose multiple validation environments capable of showing

these biases in an emphasised manner and we propose multiple simple solutions to this issue which could prevent the biased behavior without any substantial drawbacks.

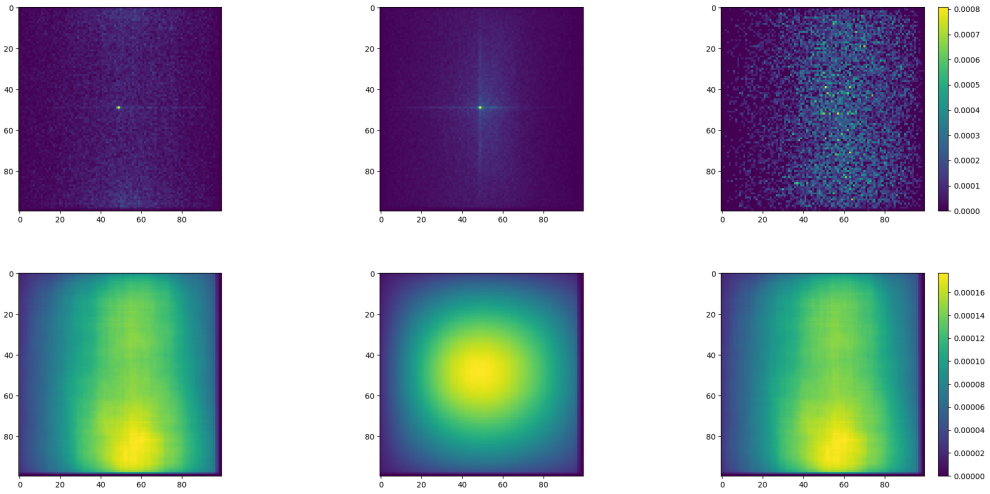
4.2 Object distribution on popular data sets

In the human visual system, there is only one location in the eye, the *fovea centralis*, where the density of light-processing cones is substantially higher than in the surrounding regions. This area is responsible for detailed central vision as well as primary color vision. [67] Since the nervous system can typically focus on only one region at a time, it is easier to center important elements in the field of view. Furthermore, it is also more aesthetically pleasing to view images where the important details and objects are positioned at the center [68]. This evolutionary consequence of human behavior can introduce a substantial bias in all human-acquired datasets.

This bias can be easily observed in simpler datasets such as MNIST, where all objects are centered, and a completely black region surrounds the edges of every image. However, in other, more complex datasets such as CIFAR, ImageNet, and MS-COCO, this bias is still present, although it might be less obvious. Measuring this bias is challenging in classification datasets where the positions of objects are not precisely annotated, but it can be easily computed in datasets where object locations are marked with bounding boxes or masks.

To illustrate this phenomenon, heatmaps representing the frequency of certain objects according to their positions in the MS-COCO dataset are showcased in Fig. 4.1.

It could be easily assumed that this bias is irrelevant in convolutional network training, given their supposed shift invariance. However, as mentioned earlier and demonstrated in the following sections of the chapter, this phenomenon can introduce a substantial bias in prediction performance, particularly when objects are positioned close to the boundary of the image.



Car Person Handbag

Figure 4.1: **MS-COCO object distributions**
Appearance probability of certain objects in the MS-COCO dataset. The top row displays a heatmap representing the frequency of center points calculated for all objects, while the bottom row shows a similar heatmap for the bounding boxes. As observed, there is a high probability that the centroid or even the entire bounding box of a randomly selected object will be located near the center, whereas objects rarely appear around the edges of the image.

4.3 Regional training results on CNNs

After recognizing the possible non-equivariant translational behavior of CNNs during initial measurements, a specific object segmentation task was designed and evaluated using the popular U-Net architecture [31]. In this section and in Sec. 4.4, all measurement results were generated using the same U-Net architecture, employing cross-entropy as the loss function, with 20 epochs of training on 60,000 randomly generated sample images with a batch size of 32, and evaluated on 16,000 samples. The task itself involved semantic segmentation and classification of handwritten digits using the MNIST handwritten digits dataset. However, the images were placed onto randomly selected and rescaled samples from the ImageNet dataset, which served as the background. The task was purposefully designed to be easy and solvable for the U-Net architecture, as the non-background regions always had the maximum possible value in the input images, and classifying

the MNIST handwritten digits dataset is a near-trivial task, even for non-state-of-the-art architectures. The handwritten digits were always positioned in the middle region of the much larger ImageNet samples, and a 168x128 region was cropped as the training and test samples. The positioning limits for all measurements mentioned in Sec. 4.3 and Sec. 4.4 were established to ensure that no parts of the objects left the image. Even in the most extreme cases, the objects just touched the edge of the image. Example input images with the corresponding labels are shown in Fig. 4.2.

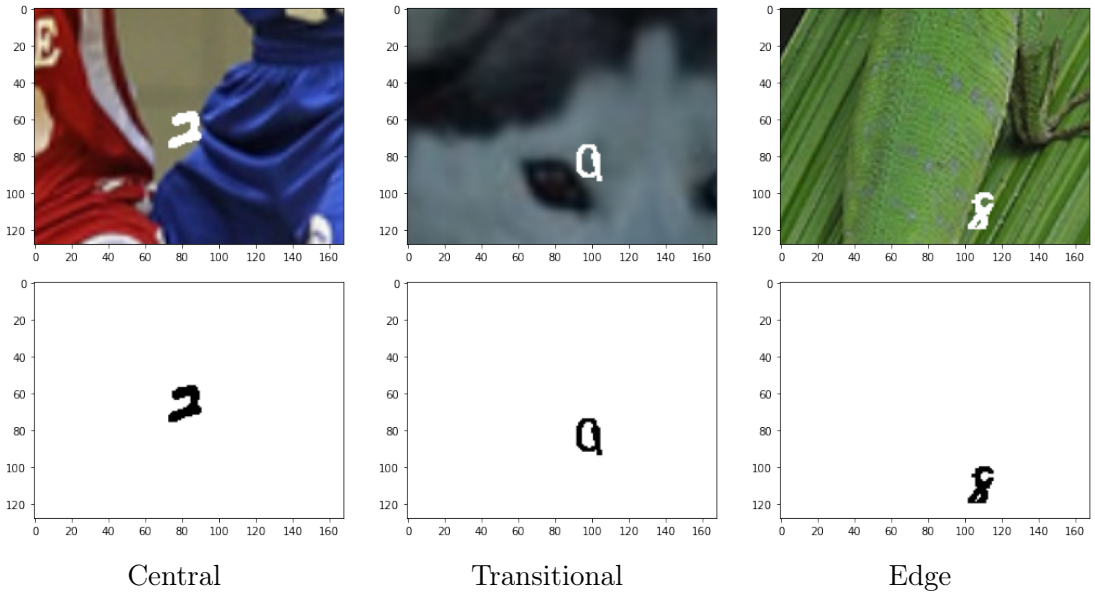


Figure 4.2: **MNIST-ImageNet samples**
Example input images are shown in the upper row, while the corresponding background labels are displayed in the lower row for the training sessions utilizing ImageNet samples as backgrounds and MNIST handwritten digit samples as the objects to be segmented and classified.

Using these images, five initial trainings were performed with the same U-Net architecture to test the hypothesis of non-equivariant translation. In half of the cases, the handwritten digits were positioned in the central 30 percent of the images, which corresponds to the central 9 percent of the image in terms of area. In the other half, the handwritten digits were positioned outside the central 70 percent of the images.

	Loss central 30%	Loss excluded central 70%	Different per Same loss ratio
Central training	0.00041	6.974354	16,086.638
Edge training	0.000690	0.000417	1.898

Table 4.1: **Initial positional bias results**
Mean values of the losses corresponding to the training and evaluation at the center and at the edges, along with the ratios between testing on the same type of dataset versus testing on the opposite type. The ratios indicate that the results deteriorate in both cases when the training and test sets have different object placement patterns. However, the increase in loss is far more drastic in the case of central training.

For the test data, two sets were initially created in a similar manner, and the mean test losses for each neural network were measured using both similarly positioned test data and oppositely positioned data. The loss values were almost the same on average for the similar datasets used for training and testing. However, switching the datasets revealed a remarkably extreme difference. Training only at the edges of the images and testing at the center increased the test loss values by approximately 1.9 times on average. In contrast, training only at the center and testing at the edges increased the loss values by more than 16,000 times. The averaged values for the four possible pairings, as well as the exact ratios between the losses for the same neural networks, are presented in Tab. 4.1.

Following this initial measurement, the problematic region was further localized by decreasing the allowed central region for the objects in the training images from 1 to 0.1. The objects were placed during testing on bands ranging from 0.0-0.1 to 0.9-1.0, where 0.0-0.1 marks the most central region of the image, and 0.9-1.0 marks the edge of the image. For each training case, five independent neural networks with the same U-Net architecture were trained, and the results for each instance were averaged. The results, shown in Fig. 4.3 and Tab. 4.2, indicate that when the objects are placed only in the central 60 percent of the images, the mean losses begin to increase rapidly even when positioned more than 30 percent away from the edges of the images. This phenomenon is present even though the objects are entirely present in the images even when placed at

the 90 percent band from the central region, so this drop in performance cannot be attributed to occlusion. It can also be observed that the outermost region, 0.9-1.0, is sensitive to objects placed in the central 80 percent of the images. Both the slightly more restrictive central 60 percent and the less restrictive central 80 percent object placements appear to be quite realistic based on the measurement results described in Sec. 4.2. Therefore, it can be concluded that unless special care is taken in the placement of the objects, the training and evaluation results on many popular datasets are likely to be biased, and the performance of CNNs in realistic applications, where objects may appear in a non-centered manner, will decline substantially.

In all measurement descriptions, only the main and most determining parameters of the experiments will be listed. For a detailed set of parameters and the PyTorch [69] based implementation of all experiments, please refer to the original publication [Ar3] and the codes in the related supplementary material.

4.4 Saliency-shift maps of regionally trained U-nets

Following the initial measurements, it was hypothesized that the class saliency maps, also referred to as attention maps [15, 16] could change depending on the position of the object, even if the positional shift of the object was minuscule. To ascertain whether this information would contribute to a deeper understanding of the phenomena, the differences in the saliency maps for the setups described in Sec. 4.3 were measured, and the absolute differences between the saliency map generated for the non-shifted (centrally placed) object and the saliency maps generated for the same images with the shifted object and background were calculated. It is important to emphasize that the objects never left the image, in the most shifted cases, the objects remained fully inside the image, positioned near the edges. For better visualization, the saliency map differences were represented as a matrix of the same size as the input images, with the coordinates representing the amount of shift in the given direction — the origin positioned at the center — and the values representing the dispersion-normalized [70] difference values. Some samples of the resulting images can be seen in Fig. 4.4.

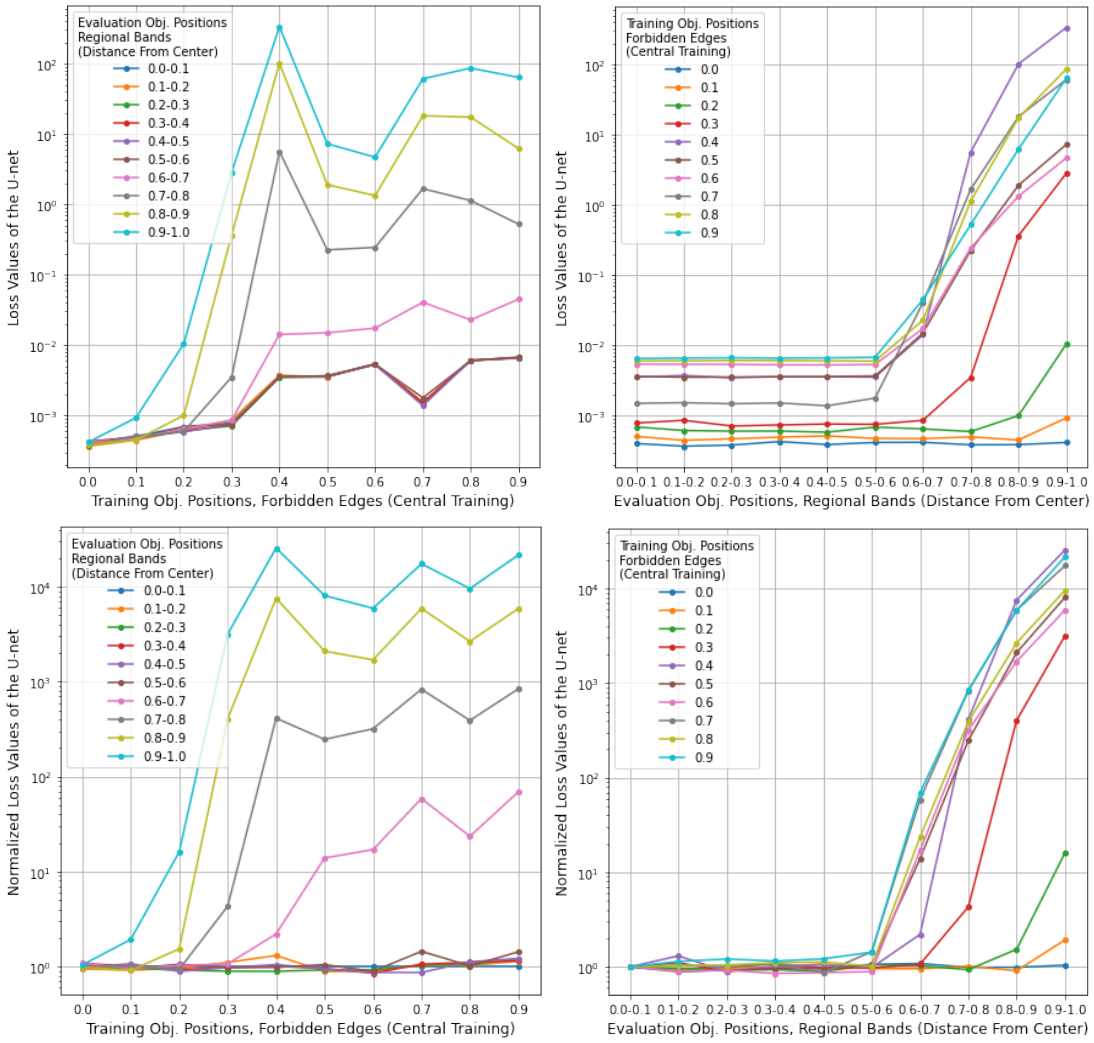


Figure 4.3: **Positional bias distributions**
Mean loss values of the U-Nets trained on the "band regions" described in Sec. 4.3. The loss values begin to increase drastically starting at the 0.5-0.6 evaluation band unless objects are specifically positioned near the edges during training. This indicates that more than 64 percent of an image will be segmented and classified with significantly worse performance in terms of area. For the numeric values, please refer to Tab. 4.2.

Based on these results, it can be strongly suspected that zero-padding has a categorically distinct effect, which cannot be achieved even by images with large completely black regions. This is because the padding has zero values in each layer, while the zero values originating from the input image can change from

Forbidden Outer Region	LossVal 0.0-0.1	LossVal 0.1-0.2	LossVal 0.2-0.3	LossVal 0.3-0.4	LossVal 0.4-0.5
0.0	0.945	0.980	1.082	0.980	1.062
0.1	0.917	0.963	1.008	1.070	0.957
0.2	0.959	0.918	0.937	0.889	1.055
0.3	1.106	0.894	0.972	0.979	0.976
0.4	1.308	0.891	1.012	1.049	0.981
0.5	0.887	0.923	0.955	0.947	1.048
0.6	0.898	0.921	0.853	0.868	0.891
0.7	1.036	1.032	1.061	0.866	1.447
0.8	1.025	1.048	1.104	1.125	1.018
0.9	1.141	1.212	1.150	1.217	1.429

	LossVal 0.5-0.6	LossVal 0.6-0.7	LossVal 0.7-0.8	LossVal 0.8-0.9	LossVal 0.9-1.0
0.0	1.085	1.085	0.987	0.989	1.041
0.1	0.956	0.956	1.019	0.909	1.927
0.2	1.028	1.028	0.934	1.535	16.101
0.3	1.066	1.066	4.315	404.340	3,164.413
0.4	2.202	2.202	411.924	7,549.861	25,575.147
0.5	13.921	13.921	246.706	2,098.795	8,063.583
0.6	17.030	17.030	319.163	1,697.250	5,924.594
0.7	58.702	58.702	831.526	5,900.890	17,408.368
0.8	23.522	23.522	390.934	2,655.341	9,557.831
0.9	69.234	69.234	839.521	5,890.076	21,579.727

Table 4.2: **Positional bias distribution values**
Mean loss values for training-evaluation pairs averaged over five individually trained U-nets. The rows display the loss values for models trained with specific "outer" restrictions on object positions, while the columns show the normalized loss values for evaluations conducted with varying regional band restrictions on object positions.

one layer to another. It is believed that the effects of zero-padding, or any other padding exhibiting behavior that is characteristically different from the input image itself, along with the potentially large-scale non-equivariant translational behavior of the maximum pooling layers, cause the drastic differences in loss values under different training circumstances described in Sec. 4.3. Furthermore,

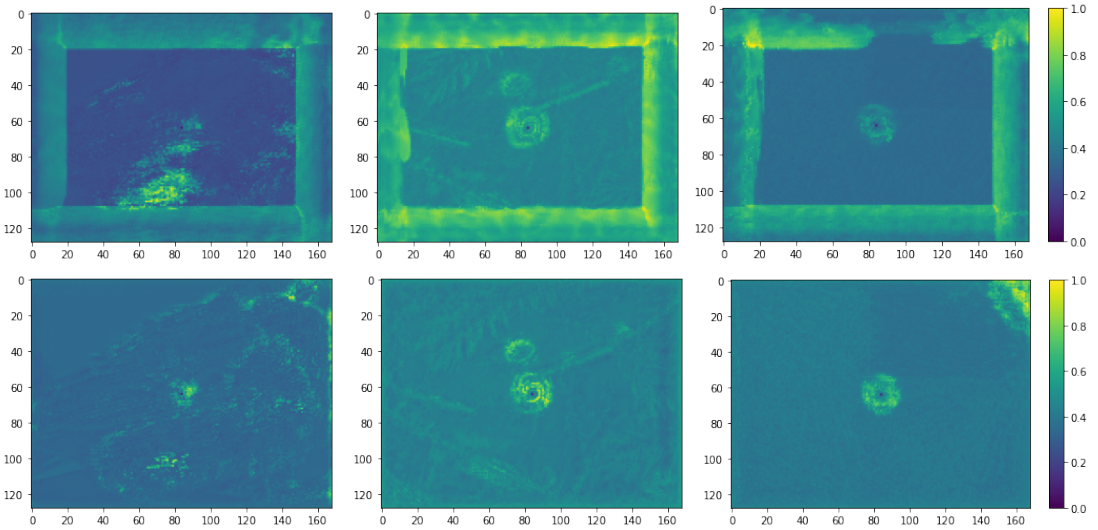


Figure 4.4: **Saliency map differences**
Normalized saliency map differences based on the shift vectors, with zero shift positioned at the center. The top row depicts the results using U-Nets trained with objects positioned in the middle, allowing for a central 30 percent, while the bottom row depicts the results using U-Nets trained with objects at the edges, prohibiting a central 70 percent. As observed, the CNNs trained on images containing objects solely at the center exhibit a distinct and large difference in their saliency maps when the objects are present at the edges. In contrast, the CNNs trained with objects positioned at the edges do not exhibit such differences.

based on the measurement results detailed in Sec. 4.3, there is a high likelihood that a convolutional neural network learns unwanted, meaningless, and destructive non-equivariant translational behavior near the edges. This issue can be easily mitigated by deliberately positioning objects near the edges of the images during training, as this object positioning compels the convolutional neural network to disregard the otherwise unusual behavior of zero-padding.

From a different perspective, a noticeable resemblance exists between the effect of boundary conditions and the mechanisms behind Dropout and, in particular, Dropblock. Thus, based on the observed behaviours, it is hypothesized that zero-padding can be considered a pathological case of Dropblock, where all activations are systematically zeroed out in every layer within a specific region. However, in the cases of Dropout and Dropblock, the remaining activations are scaled to maintain the average activation, while this mechanism is absent in the case of

boundary conditions. Additionally, the regions for Dropout and Dropblock are randomly chosen for each layer, whereas the boundary condition consistently affects the outermost pixels.

4.5 Performance of commonly applied models at the boundary

To investigate the previously introduced caveat of neural networks under prediction tasks different from semantic segmentation and classification, an instance segmentation Mask R-CNN variant architecture was examined [12], implemented in the Detectron2 environment [32]. A pretrained version of the Mask R-CNN network with a ResNet-50 feature pyramid backbone [71] was utilized with ROI align. For the sake of reproducibility, pretrained weights provided by Detectron2 for the MS-COCO dataset were employed to investigate how object shift affects the detection accuracy of the model on the dataset.

A similar decline in the objectness score of detected objects was observed as they approached the boundary of the image. Notably, this drop was not observed for large objects that occupied more than 25 percent of the image area. Such objects were detected with high confidence, even when partially out of frame. For example, individuals were detected even when only one of their arms was visible in the image. In contrast, the detection accuracy of smaller objects decreased substantially as they neared the edge of the image. An illustration of this phenomenon is presented in Fig. 4.5.

To further investigate the phenomenon, the performance of the commonly applied classification network VGG-16, on the ImageNet dataset was observed. Ten classes were selected, with ten images from each class, creating a mini dataset of 100 images. These images were manually shifted to ensure that the interesting objects were always positioned at the boundary. Pretrained models were downloaded from the TorchVision [72] model library and their performance was investigated. These models demonstrated high classification accuracy and performed well on the original samples. However, as anticipated based on the previous results, performance dropped substantially and often resulted in misclassification when objects approached the boundary. An image illustrating this effect can be seen in

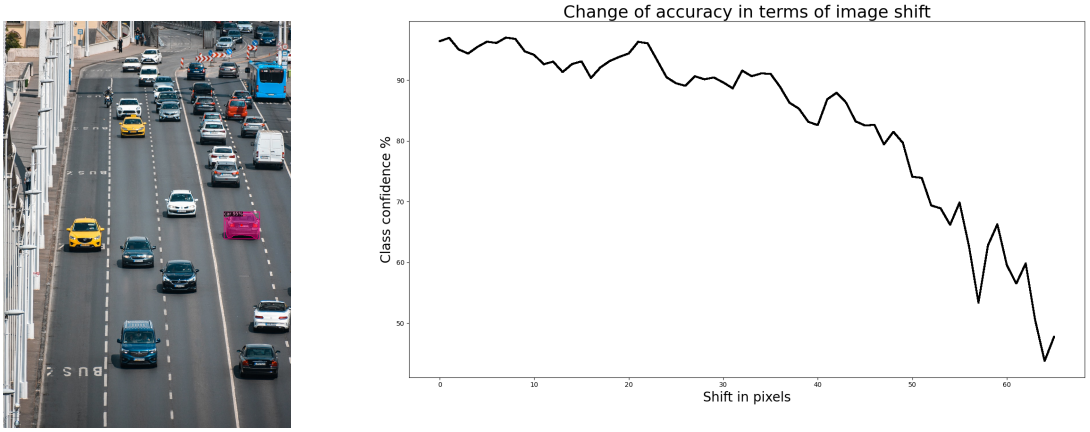


Figure 4.5: **Detection bias sample**
Depiction of the boundary effect using a Mask R-CNN, with a ResNet-50 feature pyramid backbone network, pretrained on the MS-COCO dataset. The original input image can be seen on the left, where the object marked by purple instance mask is detected with high confidence. The right plot displays how confidence changes with each periodic right shifts of the image. Even in case of maximal shift none of the pixels of the object are out of frame, the drop in detection confidence happened only due to an indirect effect of the boundary.

Fig. 4.6. The results indicate that shift invariance causes only a minor change in classification accuracy, but an abrupt decrease is observed when the object nears the boundary. This decrease is not related to shift invariance. It is important to note that the periodic shifts applied introduced a strong edge in the middle of the image, but this was not the reason for the accuracy drop, as classification results changed only slightly with smaller shifts that also created this effect.

These results demonstrate that the phenomenon described and investigated in the previous section with simple datasets also exists in complex networks and problems, including classification, detection, and segmentation.

4.6 Solutions for the mitigation of object centering bias

As highlighted by the results in Sec. 4.3, training with objects solely positioned in the central regions can introduce a notable bias in CNNs. Conversely, while

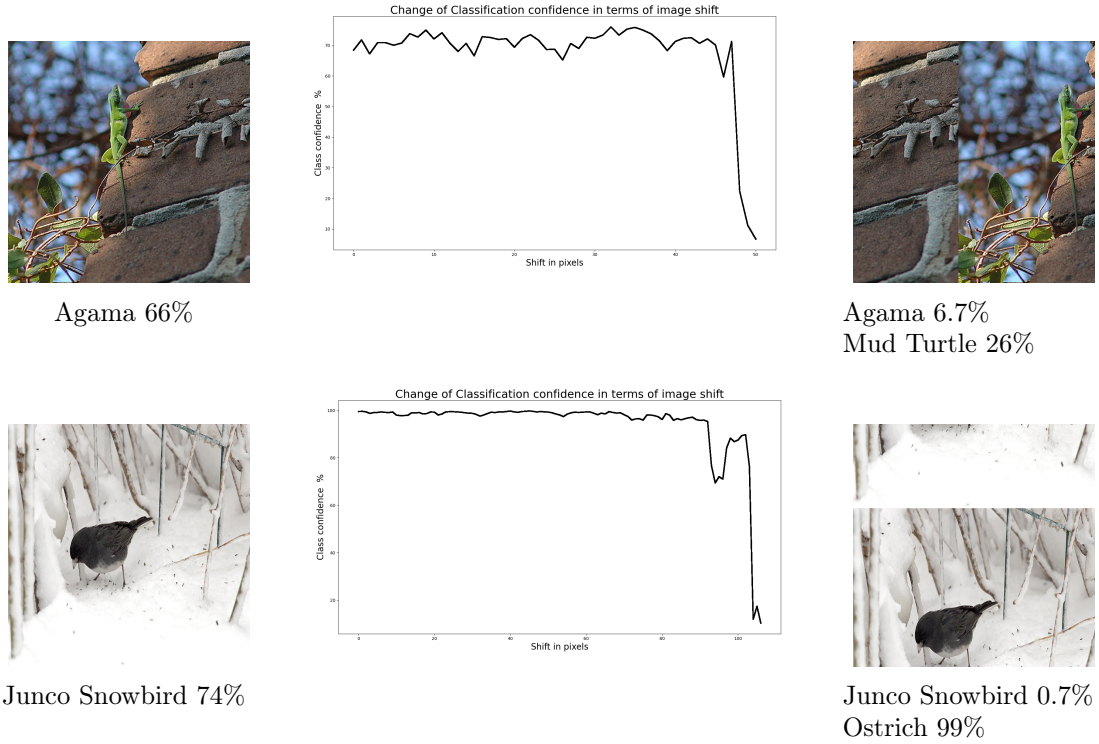


Figure 4.6: **Classification bias samples**
Sample cases showing the drastic effect of shifting objects to the boundary of the image on classification networks, using the pretrained version of VGG-16 with ImageNet samples. The first column displays the original images which are classified correctly. The second column depicts how the classification confidence of the original class changes by shifting the image. The last column depicts the largest investigated shift. The classification confidence for the original class and the newly predicted class are displayed below the images along with their confidence scores.

training with objects placed exclusively at the edges of the images still introduces some degree of bias, its effect is substantially smaller, by several orders of magnitude, compared to the central training case. This section aims to explore data manipulation-based and architecture-specific solutions to mitigate the impact of object centering bias.

4.6.1 Image cropping

One straightforward solution to address this issue is to disregard the outer regions of images. This can be achieved by employing a network with smaller input

dimensions, eliminating the use of boundary conditions altogether, and avoiding the inclusion of these pixels in computations. This approach can directly mitigate object centering bias. However, it reduces the visible and analyzable region available to the network, which may not be acceptable in many scenarios. Moreover, to apply such cropping-based transformations, it is necessary to know the shape and position of the objects in advance. Otherwise, cropping can result in the loss of critical features or even entire objects. Given these constraints, this method is impractical for a considerable number of real-world applications, unlike the other solutions discussed in this section that do not suffer from these limitations.

4.6.2 Image shifting

If the application and available data permit — based on the measurement results in Sec. 4.3 — adjusting or augmenting the training and evaluation datasets to achieve a uniform spatial distribution for each object class can possibly be an effective solution. To validate this hypothesis, a Mask R-CNN was retrained on the MS-COCO dataset by shifting objects toward the boundaries of the images. For images containing multiple objects, which is typical for this dataset, one object was selected at random, and the image was shifted periodically until the edge of the given object bounding box aligned with the boundary of the image.

Similarly, several classification architectures were retrained on the ImageNet dataset using large random translations. Although translation is a standard data augmentation technique, it typically involves only a few pixels. In this experiment, periodic translations were applied randomly, ranging from zero pixels to up to one-quarter of the image width and height. This process introduced potential issues and artifacts, such as objects being cut in half, since the exact locations of objects are unknown in this dataset. However, this manipulation increased the likelihood of objects being positioned closer to the boundaries.

After training these models, performance was evaluated on both the original datasets and two modified datasets, where objects were deliberately moved to the image boundaries. For ImageNet, the earlier-described small subset of 100 images, with objects positioned at the boundaries, was used. For MS-COCO, the

previously described method was employed to generate test images with random shifts.

The results for ImageNet are presented in Tab. 4.3, while the results for object detection on MS-COCO are shown in Tab. 4.4. As evident from these results, when objects were positioned at the boundaries during training, model performance on the original test sets dropped slightly. This decline can likely be attributed to the artifacts introduced by the shifting method. However, for test sets where objects were at the boundaries, models trained with a central bias performed substantially worse. In classification tasks, top-1 accuracy dropped by over 65%, while detection and segmentation tasks showed a decline of around 15%. In contrast, models trained with objects placed closer to the boundaries — or randomly shifted closer to the boundaries in the classification task — experienced only a slight decrease in accuracy on the original datasets at an average drop of 4%, and maintained comparable performance on the shifted test sets.

Compared to the image cropping solution described in Sec. 4.6.1, this image-shifting method is more versatile and can be applied to a wider range of applications. However, architectural solutions — as discussed in following Sec. 4.6.3 — could provide an objectively better and more practical approach. Such solutions would not require manipulation of the training dataset, would avoid the introduction of new artifacts, and could be applied universally, even when the shapes and positions of objects are unknown.

4.6.3 Toroidal boundary condition

Toroidal or otherwise periodic boundary conditions are widely utilized in classical image processing, physics simulations, cellular automata, and other fields. However, their usage is not a standard practice in convolutional neural networks (CNNs). Based on the results of the image-shifting solution described in Sec. 4.6.2, it was hypothesized that applying toroidal boundary conditions in convolutional layers could mitigate the object centering bias in a similar manner. Unlike zero-padding, this approach would avoid artificially created edges, serving as a purely architectural solution.

Architecture	ImageNet	ImageNet Boundary
VGG-16 Orig.	70.8%	7%
VGG-16 Shifted	66.4%	62%
VGG-16 Toroidal	70.6%	66%
ResNet-50 Orig.	72.1%	5%
ResNet-50 Shifted	66.75%	61%
ResNet-50 Toroidal	71.8%	65%
DenseNet121 Orig.	75.1%	3%
DenseNet121 Shifted	69.3%	66%
DenseNet121 Toroidal	74.2%	68%

Table 4.3: **Classification bias mitigation**
Top-1 test accuracies of various architectures on the original ImageNet test set (first column) and a manually created small subsample, where images have objects positioned at the boundary (second column). The rows list three architectures: VGG-16 [73], ResNet-50 [71], and DenseNet-121 [74]. Each model is evaluated in three variants: trained on the original ImageNet training set using zero-padding (Orig.), trained on a version of the dataset with objects shifted towards the boundaries (Shifted), and trained on the original dataset with toroidal boundary conditions (Toroidal).

	MS-COCO Orig	MS-COCO Shifted
Box mAP Orig.	33.6%	15.6%
Box mAP Shifted	31.7%	28.4%
Seg mAP Orig	. 31.4%	17.4%
Seg mAP Shifted	29.4%	25.3%

Table 4.4: **Detection bias mitigation**
Mean average precision (mAP) results for a Mask R-CNN network with a ResNet-50 backbone and a feature pyramid network with ROI align on the MS-COCO dataset. Two versions of the test set were evaluated: the original MS-COCO test set (MS-COCO Orig) and a modified version where a randomly selected object was always shifted to the boundary (MS-COCO Shifted). mAP results are reported for bounding box detection (Box) and instance segmentation tasks (Seg) under two different training conditions. Rows marked with (Orig.) show results on the unaltered MS-COCO dataset, while rows marked with (Shifted) show mAPs for networks trained on a dataset where a randomly selected object was always shifted to the boundary.

Furthermore, when using shifted images, all hidden layers of the network are still affected by zero-padding. However, applying toroidal boundary conditions to every convolutional layer eliminates the need to handle the influence of zero-valued boundaries, which would otherwise still affect hidden layers, even when using images with black edges.

To evaluate the impact of toroidal boundary conditions, the same regional training and evaluation procedures described in Sec. 4.3 were repeated, with the convolutional layers configured to use toroidal boundary conditions. The resulting loss values for each training region versus each evaluation band are shown in Fig. 4.7 and Tab. 4.5. As these results indicate, employing periodic boundary conditions for every convolutional layer effectively eliminates the object centering bias. When comparing the loss values of zero-padding to toroidal boundary conditions, the latter performs more than 37,000 times better in the outermost regions without causing any significant drop in prediction accuracy in the central regions.

The effect of toroidal boundary conditions was also tested on the original ImageNet dataset and its shifted variant, where objects are positioned at the boundary. The results, presented in Tab. 4.3, demonstrate that changing the boundary conditions can substantially improve the model performance for boundary-positioned objects. Moreover, saliency map differences were measured based on object shifts, following the methodology outlined in Sec. 4.4. The results are shown in Fig. 4.4. When comparing these to Fig. 4.8, it becomes apparent that training only with objects at the center yields results comparable to training with objects at the edges — an outcome that was not observed with zero-padding. Additionally, larger differences in saliency maps were eliminated using toroidal boundary conditions, and the images in Fig. 4.8 only depict minor perturbations caused by the individual test image backgrounds, which are exaggerated due to the normalization.

4.7 Discussion

The presented results highlight a significant limitation of conventionally designed and evaluated convolutional neural networks caused by boundary conditions. When objects are positioned close to the edge of an image, the zero activations introduced

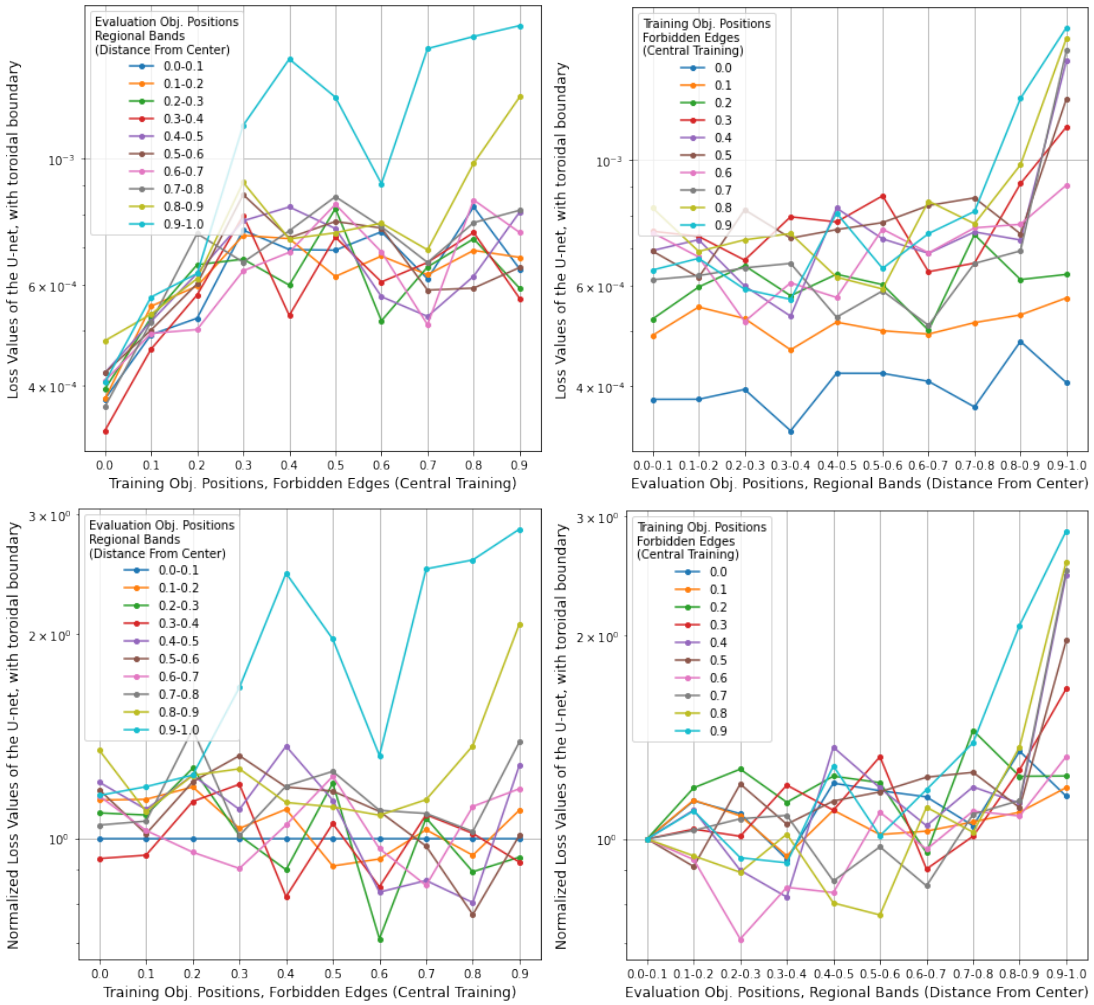


Figure 4.7: **Positional bias distributions - Toroidal boundary**
Mean loss values of the U-Nets trained on the "band regions" described in Sec. 4.3, with toroidal boundary conditions in each convolutional layer. Compared to the values in Fig. 4.3, the relative increase of loss values based on training and evaluation positions is minimal. For the numeric values, please refer to Tab. 4.5.

at the boundary result in markedly different neural network activations compared to when the objects are at the center. This phenomenon was systematically analyzed through classification, semantic segmentation and instance segmentation tasks on the MNIST, ImageNet, and MS-COCO datasets. To address this issue, two approaches were explored: shifting images so that objects appear at the boundaries during training, and employing toroidal boundary conditions instead

Forbidden Outer Region	LossVal 0.0-0.1	LossVal 0.1-0.2	LossVal 0.2-0.3	LossVal 0.3-0.4	LossVal 0.4-0.5
0.0	0.00038	0.00038	0.00039	0.00033	0.00042
0.1	0.00049	0.00055	0.00053	0.00046	0.00052
0.2	0.00053	0.00060	0.00065	0.00058	0.00063
0.3	0.00075	0.00073	0.00067	0.00079	0.00078
0.4	0.00069	0.00072	0.00060	0.00053	0.00082
0.5	0.00069	0.00062	0.00082	0.00073	0.00075
0.6	0.00074	0.00068	0.00052	0.00061	0.00057
0.7	0.00062	0.00063	0.00065	0.00066	0.00053
0.8	0.00082	0.00069	0.00072	0.00074	0.00062
0.9	0.00064	0.00067	0.00059	0.00057	0.00081

	LossVal 0.5-0.6	LossVal 0.6-0.7	LossVal 0.7-0.8	LossVal 0.8-0.9	LossVal 0.9-1.0
0.0	0.00042	0.00041	0.00037	0.00048	0.00041
0.1	0.00050	0.00049	0.00052	0.00053	0.00057
0.2	0.00060	0.00050	0.00074	0.00062	0.00063
0.3	0.00087	0.00064	0.00066	0.00091	0.00114
0.4	0.00073	0.00069	0.00075	0.00072	0.00150
0.5	0.00078	0.00083	0.00086	0.00074	0.00128
0.6	0.00076	0.00069	0.00076	0.00077	0.00090
0.7	0.00059	0.00051	0.00066	0.00069	0.00156
0.8	0.00059	0.00085	0.00077	0.00098	0.00164
0.9	0.00064	0.00074	0.00081	0.00128	0.00171

Table 4.5: **Positional bias distribution values - Toroidal boundary**
Mean loss values for training-evaluation pairs averaged over five individually trained U-nets with toroidal boundary conditions applied to convolutional layers. The rows display the loss values for models trained with specific "outer" restrictions on object positions, while the columns show the normalized loss values for evaluations conducted with varying regional band restrictions on object positions.

of zero-padding. These methods led to a substantial improvements in prediction performance at the image boundaries, which has potential implications for a variety of practical applications.

Although the influence of boundary conditions on translation invariance and

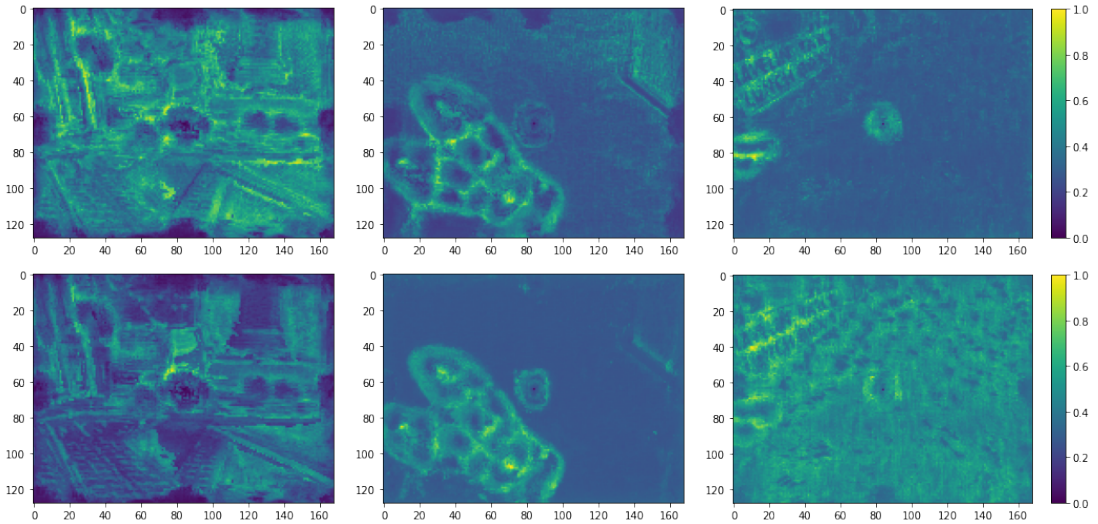


Figure 4.8: **Saliency map differences - Toroidal boundary**
Normalized saliency map differences calculated and displayed similarly to Fig. 4.4. The top row shows results for models trained with the central 30% region allowed, while the bottom row depicts results for models trained with the central 70% region restricted. Utilizing toroidal boundary conditions completely eliminated the large saliency map differences at the image edges.

object positioning has been addressed in previous studies, this work introduces several novel contributions: (1) identifying object centering bias in widely used benchmark datasets, (2) conducting a comprehensive analysis of this bias across different tasks, datasets, and models, including saliency map difference evaluations, (3) recognizing the presence of this bias even when objects are fully present within the images, (4) proposing boundary conditions as a primary factor contributing to the bias, and (5) presenting multiple fundamentally different simple practical solutions to almost completely mitigate these effects.

Chapter 5

Summary

For the specific task of videomicroscopic multi-object detection, instance segmentation, and tracking of budding yeast cells, I developed a novel architecture that substantially reduces theoretical limitations compared to the commonly used forward-tracking approach. Although this architecture is unsuitable for live tracking due to its time-symmetric data access and slower inference speed, it offers potential for superior performance from a theoretical standpoint. Following architecture design, my primary aim was to demonstrate its practical utility for the original task of budding yeast cell tracking while also showcasing its advantages across a wide range of multi-object tracking scenarios.

Initially, my focus was on evaluating the architecture for videomicroscopic cell tracking. The results showed that the architecture performed exceptionally well in yeast cell segmentation and tracking, outperforming established tools like Phylocell and YeaZ in various tests. Notably, it even surpassed Phylocell in scenarios biased in Phylocell's favor, highlighting the robustness and accuracy of this approach. Through hyperparameter tuning, it was found that shorter local tracking ranges were more beneficial when segmentation was reliable, as they reduced both model complexity and data requirements. However, in instances where segmentation was inconsistent — such as when objects disappeared over several frames — longer local tracking ranges helped preserve tracking accuracy. This finding underscores the need to adjust tracking thresholds, particularly in datasets where object detection is challenging. It was also determined that a low-complexity encoding backbone was sufficient for yeast cell tracking, making

the model computationally efficient, although using a GPU-accelerated inference is still recommended for optimal performance. For track matching, Intersection over Union (IoU) proved to be the most reliable metric for evaluating prediction similarity. Further tests using biologically inspired synthetic datasets demonstrated the model's versatility, suggesting that with sufficient annotated training data, the architecture can be retrained for other cell types. Although the synthetic datasets presented greater challenges compared to the yeast data, the architecture still produced reliable and empirically valuable predictions, even if they might require occasional manual correction. These findings provide valuable insights into the model's data requirements and retraining potential, supporting its broader application in diverse cell-tracking tasks.

Following this, I shifted focus to a more generalized evaluation of the architecture. Using multiple synthetic scenarios and a subset of the MOTS challenge dataset, a detailed analysis of the instance segmentation and tracking model was conducted, particularly focusing on the novel tracking mechanism. Although the architecture was originally designed for yeast cell tracking, where it excelled, the evaluation was expanded to assess its performance in environments with different modalities. Comparative analyses were conducted against two restricted variants of the local tracking model as well as the widely used Kalman filter. The results demonstrated the clear architectural advantages of the proposed model, while also highlighting the limitations of the alternative approaches. These comparisons emphasized the critical role of selecting the appropriate architecture when addressing different morphological and visual cues. Furthermore, the proposed architecture was able to achieve state-of-the-art overall segmentation and tracking performance on the MOTS personnel tracking dataset, while delivering far superior associative tracking performance compared to the benchmark Tracktor model. While there is still room for further refinement, these findings illustrate the architecture's unique strengths and its utility in multi-object tracking tasks on pre-recorded data.

In parallel to the primary focus on multi-object tracking, I discovered and conducted a thorough analysis of an intriguing failure of convolutional networks and related benchmark datasets. The results indicate that boundary conditions can introduce zero activations in each layer. Consequently, objects located near the edges of an image generate highly different activations compared to those

positioned in the center. This phenomenon was extensively investigated across classification and instance segmentation tasks on the MNIST, ImageNet, and MS-COCO datasets, utilizing localization with region-based performance evaluations and comparing differences in saliency maps. Furthermore, I demonstrated that this issue can be mitigated by either shifting the image towards the boundary during training or by applying toroidal boundary conditions instead of zero-padding. This training methodology substantially enhanced the network's accuracy at the boundaries, almost entirely eliminating the bias. The implications of this finding hold considerable potential importance for various practical applications.

New Scientific Results

Thesis 1a

I developed a novel deep-learning-based multi-object instance segmentation and tracking architecture for videomicroscopic recordings of budding yeast cells. On a yeast tracking dataset collected by IFOM, the architecture achieved IoU-based segmentation and tracking F-scores of $[0.918 \pm 0.019,\ 0.917 \pm 0.016]$*, respectively. This performance surpassed that of competing state-of-the-art tools designed for this particular task, specifically Phylocell* $[0.881 \pm 0.020,\ 0.878 \pm 0.020]$ *and YeaZ* $[0.818 \pm 0.022,\ 0.807 \pm 0.023]$.
Corresponding publication: [Ar1]

Thesis 1b

The proposed architecture is inherently capable of reconstructing fragmented tracks due to its novel time-symmetric tracking approach, greatly improving tracking consistency, a critical requirement for accurate cell inheritance assignment. For instance, in scenarios with uniform random removal of every fifth object instance, the tracking-based reconstruction improved the tracking F-score from 0.404 ± 0.016 *to* 0.888 ± 0.013.
Corresponding publications: [Ar1, Ar2]

Thesis 1c

I evaluated the proposed architecture on various synthetic and semi-synthetic datasets, showcasing its robustness in addressing potential challenges in cell tracking environments and other natural tracking contexts, such as pedestrian tracking. The results demonstrate the versatility of the architecture, the reliability of its novel tracking approach, and can serve as a guide of expected performance on other datasets.
Corresponding publications: [Ar1, Ar2]

Thesis 1d

I conducted an ablation study on the proposed architecture, separating the contributions of motility-based and morphology-based tracking, which the original architecture integrates seamlessly. The results revealed the individual impact of motility and morphology on tracking performance across the evaluated scenarios and highlighted the advantages of a tracking method that combines both. Furthermore, the proposed architecture consistently matched or outperformed the widely used Kalman filter in all scenarios. Notably, in the scenario with semi-randomized object motility, the architecture achieved a 3.49-fold improvement in association F-score, due to its ability to utilize all temporally local imaging information for tracking.
Corresponding publication: [Ar2]

Thesis 2a

I demonstrated that widely used benchmark datasets, such as MS-COCO, exhibit an object positioning bias, strongly favoring objects located near the center of the image. This bias can result in prediction performance that is more than five orders of magnitude lower near the edges of the image, even when the objects are fully visible. The effects of this bias were analyzed for segmentation, detection and classification tasks, with a detailed localization of its impact based on prediction performance results and saliency maps.
Corresponding publication: [Ar3]

Thesis 2b

I proposed architectural and data manipulation-based solutions to mitigate this bias. The most effective approach involved replacing zero-padding in all convolutional layers of the model with toroidal boundary conditions. This modification led to a performance improvement of more than 37,000 times in the most extreme cases. Corresponding publication: [Ar3]

Author publications

Publications related to the thesis

[Ar1] G. Szabó, P. Bonaiuti, A. Ciliberto, and A. Horváth, "Enhancing cell tracking with a time-symmetric deep learning approach," *arXiv preprint arXiv:2308.03887*, 2023. 1.1, 3.3, 5, 5, 5

[Ar2] G. Szabó, Z. Molnár, and A. Horváth (2024). "Post-Hoc MOTS: Exploring the Capabilities of Time-Symmetric Multi-Object Tracking", *arXiv preprint arXiv:2412.08313*, 2024 1.1, 5, 5, 5

[Ar3] G. Szabó and A. Horváth, "Mitigating the bias of centered objects in common datasets," in *2022 26th International Conference on Pattern Recognition (ICPR)*, pp. 4786–4792, IEEE, 2022. 1.2, 4.3, 5, 5

Other publications of the author

[Au1] R. Bagdy-Bálint, G. Szabó, Ö. H. Zováthi, B. H. Zováthi, Á. Somorjai, C. Köpenczei, and N. K. Rózsa, "Accuracy of automated analysis in cephalometry," *Journal of Dental Sciences*, Elsevier, 2024.

[Au2] D. Babicz, S. Kontár, M. Peto, A. Fulop, G. Szabó, and A. Horváth, "Receptive field size optimization with continuous time pooling," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1449–1458, 2021.

[Au3] W. A. Bosbach, B. Németh, R. Zelei, J. F. Senge, B. Pasztor, L. Ebner, M. Szabo, S. Anderson, G. Szabo, P. Dlotko, A. Horvath, K. Daneshvar, and J.

Heverhagen, "SCR 2023 poster presentation: An open access AI-based pattern recognition tool for application in MSK imaging – suitability and limitations of resources today [Poster]," *Swiss Congress of Radiology*, Zenodo, 2023. DOI: 10.5281/zenodo.7958527.

[Au4] G. Szabó, "Time-symmetric generalization of multi-object tracking," *PhD Proceedings Annual Issues of the Doctoral School Faculty of Information Technology and Bionics*, vol. 19, pp. 189–192, 2024. [Online]. Available: `https://m2.mtmt.hu/api/publication/35446463`.

[Au5] G. Szabó, "Time symmetric tracking of yeast cells using convolutional neural networks," *PhD Proceedings Annual Issues of the Doctoral School Faculty of Information Technology and Bionics*, vol. 18, pp. 145–148, 2023. [Online]. Available: `https://m2.mtmt.hu/api/publication/34176533`.

[Au6] G. Szabó, "The effects and mitigation of object centering bias in common datasets," *PhD Proceedings Annual Issues of the Doctoral School Faculty of Information Technology and Bionics*, vol. 17, pp. 191–194, 2022. [Online]. Available: `https://m2.mtmt.hu/api/publication/35446504`.

[Au7] G. Szabó, "Investigation of cell motility," *PhD Proceedings Annual Issues of the Doctoral School Faculty of Information Technology and Bionics*, vol. 16, pp. 169–172, 2021. [Online]. Available: `https://m2.mtmt.hu/api/publication/35446512`.

References

[1] E. Meijering, O. Dzyubachyk, I. Smal, and W. A. van Cappellen, "Tracking in cell and developmental biology," in *Seminars in cell & developmental biology*, vol. 20, pp. 894–902, Elsevier, 2009. 1.1

[2] B. Isherwood, P. Timpson, E. J. McGhee, K. I. Anderson, M. Canel, A. Serrels, V. G. Brunton, and N. O. Carragher, "Live cell in vitro and in vivo imaging applications: accelerating drug discovery," *Pharmaceutics*, vol. 3, no. 2, pp. 141–170, 2011. 1.1

[3] L. H. de Wit, C. E. Lefevre, R. W. Kentridge, G. Rees, and A. P. Saygin, "Investigating the status of biological stimuli as objects of attention in multiple object tracking," *PloS one*, vol. 6, no. 3, p. e16232, 2011. 1.1

[4] S. Kamkar, F. Ghezloo, H. A. Moghaddam, A. Borji, and R. Lashgari, "Multiple-target tracking in human and machine vision," *PLoS computational biology*, vol. 16, no. 4, p. e1007698, 2020. 1.1

[5] A. Ess, B. Leibe, and L. Van Gool, "Depth and appearance for mobile scene analysis," in *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, 2007. 1.1

[6] P. Dendorfer, "Mot20: A benchmark for multi object tracking in crowded scenes," *arXiv preprint arXiv:2003.09003*, 2020. 1.1

[7] M. Fabbri, G. Brasó, G. Maugeri, O. Cetintas, R. Gasparini, A. Ošep, S. Calderara, L. Leal-Taixé, and R. Cucchiara, "Motsynth: How can synthetic data help pedestrian detection and tracking?," in *Proceedings of the*

IEEE/CVF International Conference on Computer Vision, pp. 10849–10859, 2021. 1.1, 3.2, 3.6, 3.6.3

[8] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE transactions on neural networks*, vol. 22, no. 2, pp. 199–210, 2010. 1.1

[9] Y. Xian, B. Schiele, and Z. Akata, "Zero-shot learning-the good, the bad and the ugly," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4582–4591, 2017. 1.1

[10] Z. Murez, S. Kolouri, D. Kriegman, R. Ramamoorthi, and K. Kim, "Image to image translation for domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4500–4509, 2018. 1.1

[11] R. E. Kálmán, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960. 1.1, 2.2, 3.2, 3.5

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017. 1.2, 2.3.1, 4.5

[13] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012. 1.2, 4.1

[14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014. 1.2, 2.3.1, 4.1

[15] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013. 1.2, 4.4

[16] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," *Advances in neural information processing systems*, vol. 31, 2018. 1.2, 4.4

[17] H. Gholamalinezhad and H. Khosravi, "Pooling methods in deep neural networks, a review," *arXiv preprint arXiv:2009.07485*, 2020. 1.2

[18] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, pp. 1150–1157, Ieee, 1999. 2.2

[19] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981. 2.2

[20] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *Signal processing, sensor fusion, and target recognition VI*, vol. 3068, pp. 182–193, Spie, 1997. 2.2, 2.3.2, 3.5

[21] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 adaptive systems for signal processing, communications, and control symposium (Cat. No. 00EX373)*, pp. 153–158, Ieee, 2000. 2.2, 2.3.2, 3.5

[22] T. Hu, S. Xu, L. Wei, X. Zhang, and X. Wang, "Celltracker: an automated toolbox for single-cell segmentation and tracking of time-lapse microscopy images," *Bioinformatics*, vol. 37, no. 2, pp. 285–287, 2021. 2.2

[23] H.-F. Tsai, J. Gajda, T. F. Sloan, A. Rares, and A. Q. Shen, "Usiigaci: Instance-aware cell tracking in stain-free phase contrast microscopy enabled by machine learning," *SoftwareX*, vol. 9, pp. 230–237, 2019. 2.2

[24] N. Dietler, M. Minder, V. Gligorovski, A. M. Economou, D. A. H. L. Joly, A. Sadeghi, C. H. M. Chan, M. Koziński, M. Weigert, A.-F. Bitbol, *et al.*, "A convolutional neural network segments yeast microscopy images with high accuracy," *Nature communications*, vol. 11, no. 1, p. 5723, 2020. 2.2

[25] Y. Chen, Y. Song, C. Zhang, F. Zhang, L. O'Donnell, W. Chrzanowski, and W. Cai, "Celltrack r-cnn: A novel end-to-end deep neural network for cell segmentation and tracking in microscopy images," in *2021 IEEE 18th*

International Symposium on Biomedical Imaging (ISBI), pp. 779–782, IEEE, 2021. 2.2

[26] S. Berg, D. Kutra, T. Kroeger, C. N. Straehle, B. X. Kausler, C. Haubold, M. Schiegg, J. Ales, T. Beier, M. Rudy, *et al.*, "Ilastik: interactive machine learning for (bio) image analysis," *Nature methods*, vol. 16, no. 12, pp. 1226–1232, 2019. 2.2

[27] D. Vlah, A. Kastrin, J. Povh, and N. Vukašinović, "Data-driven engineering design: A systematic review using scientometric approach," *Advanced Engineering Informatics*, vol. 54, p. 101774, 2022. 2.2

[28] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, pp. 843–852, 2017. 2.2

[29] V. Katariya, M. Baharani, N. Morris, O. Shoghli, and H. Tabkhi, "Deeptrack: Lightweight deep learning for vehicle trajectory prediction in highways," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 18927–18936, 2022. 2.2

[30] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017. 2.3.1

[31] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015. 2.3.1, 4.3

[32] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." `https://github.com/facebookresearch/detectron2`, 2019. 2.3.1, 3.3, 4.5

[33] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: fast and flexible image augmentations," *Information*, vol. 11, no. 2, p. 125, 2020. 2.3.1

[34] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955. 2.3.2

[35] X. Wang, A. Jabri, and A. A. Efros, "Learning correspondence from the cycle-consistency of time," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2566–2576, 2019. 2.3.2

[36] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, 2018. 2.3.2

[37] P. Iakubovskii, "Segmentation models pytorch." `https://github.com/qubvel/segmentation_models.pytorch`, 2019. 2.3.2

[38] PyTorch Contributors, "torchvision.transforms." `https://pytorch.org/docs/stable/torchvision/transforms.html`, 2023. 2.3.2

[39] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972. 2.3.2

[40] M. Maška, V. Ulman, P. Delgado-Rodriguez, E. Gómez-de Mariscal, T. Nečasová, F. A. Guerrero Peña, T. I. Ren, E. M. Meyerowitz, T. Scherr, K. Löffler, *et al.*, "The cell tracking challenge: 10 years of objective benchmarking," *Nature Methods*, pp. 1–11, 2023. 2.3.2, 2.4.2

[41] A. Arbelle, S. Cohen, T. R. Raviv, and T. Ben-Haim, "Bgu-il (5) description," 2.3.2

[42] I. E. Toubal, N. Al-Shakarji, D. Cornelison, and K. Palaniappan, "Ensemble deep learning object detection fusion for cell tracking, mitosis, and lineage," *IEEE Open Journal of Engineering in Medicine and Biology*, 2023. 2.3.2, 2.4.2

[43] M. Primet, *Probabilistic methods for point tracking and biological image analysis*. PhD thesis, Université René Descartes-Paris V, 2011. 2.4

[44] G. Charvin, "Phylocell," 2021. 2.4.1, 2.4.3

[45] S. Anjum and D. Gurari, "Ctmc: Cell tracking with mitosis detection dataset challenge," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 982–983, 2020. 2.4.2

[46] K. Perlin, "An image synthesizer," *ACM Siggraph Computer Graphics*, vol. 19, no. 3, pp. 287–296, 1985. 2.4.2, 3.6.2

[47] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986. 2.4.2

[48] S. Fehrmann, C. Paoletti, Y. Goulev, A. Ungureanu, H. Aguilaniu, and G. Charvin, "Aging yeast cells undergo a sharp entry into senescence unrelated to the loss of mitochondrial membrane potential," *Cell reports*, vol. 5, no. 6, pp. 1589–1599, 2013. 2.4.3

[49] F. Padovani, B. Mairhörmann, P. Falter-Braun, J. Lengefeld, and K. M. Schmoller, "Segmentation, tracking and cell cycle analysis of live-cell imaging data with cell-acdc," *BMC biology*, vol. 20, no. 1, p. 174, 2022. 2.4.3

[50] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe, "Mots: Multi-object tracking and segmentation," in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pp. 7942–7951, 2019. 3.2, 3.6

[51] P. Iakubovskii, "Segmentation models pytorch." `https://github.com/qubvel/segmentation_models.pytorch`, 2019. 3.3

[52] J. Luiten, A. Osep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixé, and B. Leibe, "Hota: A higher order metric for evaluating multi-object tracking," *International journal of computer vision*, vol. 129, pp. 548–578, 2021. 3.4, 3.4.2

[53] B. Millidge, A. Tschantz, A. Seth, and C. Buckley, "Neural kalman filtering," *arXiv preprint arXiv:2102.10021*, 2021. 3.5

[54] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The annals of mathematical statistics*, vol. 37, no. 6, pp. 1554–1563, 1966. 3.5

[55] G. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973. 3.5

[56] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, *et al.*, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024. 3.6

[57] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking without bells and whistles," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 941–951, 2019. 3.6.3

[58] G. R. Terrell and D. W. Scott, "Variable kernel density estimation," *The Annals of Statistics*, pp. 1236–1265, 1992. 3.7

[59] C. B. Murthy, M. F. Hashmi, N. D. Bokde, and Z. W. Geem, "Investigations of object detection in images/videos using various deep learning techniques and embedded platforms—a comprehensive review," *Applied sciences*, vol. 10, no. 9, p. 3280, 2020. 3.7.2

[60] A. B. Abdusalomov, B. M. S. Islam, R. Nasimov, M. Mukhiddinov, and T. K. Whangbo, "An improved forest fire detection method based on the detectron2 model and a deep learning approach," *Sensors*, vol. 23, no. 3, p. 1512, 2023. 3.7.2

[61] V. Pham, C. Pham, and T. Dang, "Road damage detection and classification with detectron2 and faster r-cnn," in *2020 IEEE International Conference on Big Data (Big Data)*, pp. 5592–5601, IEEE, 2020. 3.7.2

[62] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)," *URL http://www. cs. toronto. edu/kriz/cifar. html*, vol. 5, p. 4, 2010. 4.1

[63] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. 4.1

[64] R. Zhang, "Making convolutional networks shift-invariant again," in *International conference on machine learning*, pp. 7324–7334, PMLR, 2019. 4.1

[65] O. S. Kayhan and J. C. v. Gemert, "On translation invariance in cnns: Convolutional layers can exploit absolute spatial location," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14274–14285, 2020. 4.1

[66] M. A. Islam, M. Kowal, S. Jia, K. G. Derpanis, and N. D. Bruce, "Position, padding and predictions: A deeper look at position information in cnns," *arXiv preprint arXiv:2101.12322*, 2021. 4.1

[67] E. N. Willmer and W. D. Wright, "Colour sensitivity of the fovea centralis," *Nature*, vol. 156, no. 3952, pp. 119–121, 1945. 4.2

[68] Y.-C. Chen, C. Colombatto, and B. J. Scholl, "Looking into the future: An inward bias in aesthetic experience driven only by gaze cues," *Cognition*, vol. 176, pp. 209–214, 2018. 4.2

[69] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019. 4.3

[70] M. Morisita, "I σ-index, a measure of dispersion of individuals," *Researches on population ecology*, vol. 4, no. 1, pp. 1–7, 1962. 4.4

[71] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. 4.5, 4.3

[72] S. Marcel and Y. Rodriguez, "Torchvision the machine-vision package of torch," in *Proceedings of the 18th ACM international conference on Multimedia*, pp. 1485–1488, 2010. 4.5

[73] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 4.3

[74] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017. 4.3