

Visual Detection and Tracking of Pigs

On the Use of Computer Vision in Livestock
Farming

Dipl.-Inf. Johannes Brünger

Dissertation
zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
(Dr.-Ing.)
der Technischen Fakultät
der Christian-Albrechts-Universität zu Kiel
eingereicht im Jahr 2022

Kiel Computer Science Series (KCSS) 2022/4 dated 2022-05-24

ISSN 2193-6781 (print version)

ISSN 2194-6639 (electronic version)

Electronic version, updates, errata available via <https://www.informatik.uni-kiel.de/kcss>

The author can be contacted via brngr@posteo.de

Published by the Department of Computer Science, Kiel University

Multimedia Information Processing Group

Please cite as:

- ▷ Johannes Brünger. *Visual Detection and Tracking of Pigs - On the Use of Computer Vision in Livestock Farming* Number 2022/4 in Kiel Computer Science Series. Department of Computer Science, 2022. Dissertation, Faculty of Engineering, Kiel University.

```
@book{bruenger2022diss,  
  author = {Johannes Br{"u}nger},  
  title   = {Visual Detection and Tracking of Pigs  
            - On the Use of Computer Vision in Livestock Farming},  
  publisher = {Department of Computer Science, Kiel University},  
  year     = {2022},  
  number   = {2022/4},  
  doi      = {10.21941/kcss/2022/4},  
  series   = {Kiel Computer Science Series},  
  note     = {Dissertation, Faculty of Engineering,  
            Kiel University.}  
}
```

© 2022 by Johannes Brünger

About this Series

The Kiel Computer Science Series (KCSS) covers dissertations, habilitation theses, lecture notes, textbooks, surveys, collections, handbooks, etc. written at the Department of Computer Science at Kiel University. It was initiated in 2011 to support authors in the dissemination of their work in electronic and printed form, without restricting their rights to their work. The series provides a unified appearance and aims at high-quality typography. The KCSS is an open access series; all series titles are electronically available free of charge at the department's website. In addition, authors are encouraged to make printed copies available at a reasonable price, typically with a print-on-demand service.

Please visit <http://www.informatik.uni-kiel.de/kcss> for more information, for instructions how to publish in the KCSS, and for access to all existing publications.

1. Gutachter: Prof. Dr.-Ing. Reinhard Koch
Christian-Albrechts-Universität
Kiel
2. Gutachter: Prof. Dr. Hauke Schramm
Christian-Albrechts-Universität
Kiel

Datum der mündlichen Prüfung: 06.05.2022

Zusammenfassung

Mit dem steigenden Bedarf an Fleisch und dem entsprechenden Wachstum der Nutztierbestände, sind automatisierte Überwachungssysteme eine sinnvolle Erweiterung der Werkzeuge um das Wohlergehen der Tiere sicherzustellen. Ebenso ermöglichen solche Systeme die Verarbeitung größerer Datenmengen für die Erforschung und Verhaltens-Studien für zukünftige Empfehlungen zur artgerechten Haltung. Diese Arbeit zeigt am Beispiel von Schweinen den Einsatz verschiedener Methoden zur kameragestützten Detektion und Verfolgung der Tiere. Die so entstehenden Aufenthalts- und Bewegungsinformationen können für die gezielte Untersuchung des Verhalten der Tiere in unterschiedlichen experimentellen Stallumgebungen oder Korrelation von Bewegung zu anderen Parametern der Tiergesundheit genutzt werden. Die Ergebnisse zeigen eine hohe Genauigkeit der vorgestellten Verfahren und bieten so praxisnahe Einsatzmöglichkeit an.

Abstract

With the increasing demand for meat and the corresponding growth in livestock populations, automated monitoring systems are a useful extension of tools to ensure animal welfare. Likewise, such systems enable the processing of larger amounts of data for research and behavioral studies for future welfare recommendations. This work uses pigs as an example to demonstrate the use of various methods for camera-based detection and tracking of animals. The resulting residence and movement information can be used for targeted study of animal behavior in different experimental housing environments or correlation of movement to other animal health parameters. The results show a high accuracy of the presented methods and thus indicate various practical applications.

Acknowledgements

First and foremost, I would like to thank my wife Anne, who has supported me in everything over the last few years and thus made it possible for me to write this dissertation.

Furthermore, I would like to thank all members of the research group for the wonderful time as a colleague and doctoral student. In particular, I would like to thank Renate and Torge for their great assistance.

I would like to thank Reinhard for the opportunity to work with him and for the support and advice during my time at the university. I would also like to thank Imke Traulsen and her team for the interdisciplinary collaboration which motivated me a lot.

And of course I would like to thank Reinhard, Hauke Schramm, Peer Kröger and Olaf Landsiedel for the review and involvement in the examination board.

Contents

Acknowledgements	ix
1 Introduction	1
1.1 Motivation	2
1.2 Contributions	4
1.3 Outline	5
I Principles of Detection and Tracking	9
2 Interrelationship of Detection and Tracking	11
2.1 Problem Formulation	12
2.2 Detection-Free Tracking	13
2.3 Detection-Based Tracking	13
2.4 Object Modeling	14
2.4.1 Appearance Model	15
2.4.2 Motion Model	19
2.4.3 Interaction Model	20
3 Detection	23
3.1 Traditional Detectors	25
3.2 Detection with Neural Networks	29
3.2.1 Deep Neural Networks	29
3.2.2 Different Approaches with Neural Networks	31
3.2.3 Detection Networks	32
3.2.4 Segmentation Networks	34
4 Tracking	39
4.1 Probabilistic Approach	39
4.2 Optimization based Approach	44

Contents

5	Evaluation Metrics	49
5.1	Metrics for Detection	49
5.2	Metrics for Tracking	53
II	Detection and Tracking of Pigs	59
6	Applications	61
6.1	Problem Description	62
6.2	Related Work	63
6.2.1	Pigs	63
6.2.2	Other Livestock or Laboratory Animals	65
6.3	Data Sets	67
6.3.1	Synthetic Data	69
6.3.2	Data Preparation	70
6.4	Animal Model	72
6.4.1	Motion Analysis	74
6.5	Ground Truth Data	76
6.5.1	Ground Truth for Tracking Evaluation	78
7	Detection-Free Tracking	79
7.1	Detection and Tracking with Analysis by Synthesis	81
7.1.1	Fitness-Function	83
7.1.2	Update Iteration	88
7.1.3	Recovery	89
7.2	Evaluation	90
7.2.1	Evaluation of Detection	91
7.2.2	Evaluation of Tracking	94
7.2.3	Runtime	95
8	Detection-Based Tracking	97
8.1	Detection of Pigs with Panoptic Segmentation	97
8.1.1	Framework	98
8.1.2	Categorical Segmentation	102
8.1.3	Instance Segmentation	105
8.2	Tracking by Detection	111
8.2.1	Transition Costs	111

8.2.2	Bipartite Matching between Frames	113
8.2.3	Global Bipartite Matching	114
8.2.4	Minimum-Cost Flow	116
8.3	Evaluation	118
8.3.1	Evaluation of Detection Methods	118
8.3.2	Evaluation of Tracking Methods	130
8.3.3	Runtime	136
9	Conclusion	139
A	Deep Neural Networks	143
B	Covariance Matrix Adaptation - Evolution Strategy	147
	Bibliography	151

List of Figures

2.1	Schematic representation of <i>Detection-Free Tracking</i> . (a) The real objects are shown opaque, their tracked position semi-transparent. (b) In the next time step, the objects have moved and the tracked positions are obsolete. (c) Many observations are generated, based on the object model. (d) The best matching observation is chosen as the new tracked position of the object.	13
2.2	Schematic representation of <i>Detection-Based Tracking</i> . (a) The real objects are shown opaque, their tracked position semi-transparent. (b) In the next time step, observations are available in the form of external detections. (c) The detections are assigned to the already known targets. (d) The best association is chosen as the new position. If no assignment can be found, a new trajectory is generated (in this case a false positive).	14
3.1	Visualization of examples of traditional detection methods. (a) Rectangular feature to evaluate the difference in brightness between the eye area and the upper cheek [VJ01]. (b) Stored representation and its distance to potential detections (with NCC) [KMM12]. (c) Points with high redetection probability for tracking [Shi+94].	27
3.2	More visualizations of traditional detection methods applied to an example image (a). (b) Histogram of gradients [DT05] for feature description. (c) Graph based segmentation for part models [FH04], and (d) region proposals based on the segmentation from (c).	29

List of Figures

3.3	Examples of different approaches used for object detection with neural networks. (a) Classical bounding boxes based on region proposals. (b) Semantic segmentation by assigning each pixel to a class. (c) Instance segmentation by assigning each pixel to an object instance. (d) Key point detection to find features and subsequently assign them to instances.	31
3.4	Schematic representation of different network architectures for object detection. (a) A two stage approach with RPN and Classification-Head [RHG+15]. (b, c) One stage detection as in [RDG+16] and [LAE+16].	33
3.5	Schematic representation of different architectures of segmentation networks with encoder and decoder parts (depiction adopted from [Yak19]). (a) Fully convolutional network with simple upsampling [LSD15]. (b - d) U-Net, PSPN and FPN with different strategies for processing the latent representation [LSD15; ZSQ+17; LDG+17].	35
4.1	The Bayesian Network consisting of two random variables (configuration and observation) and their dependency is extended as a Markov chain (figure adopted from [Smi07]).	40
4.2	Schematic depiction of a particle-filter. (a) The initial state at time step n with the posterior distribution of the red target and its approximation as weighted particles. (b) Precision phase with the motion model applied to the new particles. (c) Update phase with the evaluation based on the observation likelihood.	42
4.3	Tracking of three objects as min-cost/ max-flow problem. The association is solved by the assignment of transition probabilities as costs to the edges of the flow-graph. (figure adopted from [BFT+11])	45
5.1	Example of <i>Intersection over Union</i> (IoU) for a case with bounding boxes and one with pixel-wise calculation.	51

5.2 Example of a tracking result for evaluation. The assignment of the detections to the ground truth positions is based on the IoU. If the assignment changes along a ground truth trajectory, an identity switch is counted. If a ground truth trajectory is interrupted (e.g. by a false negative detection) a fragmentation is counted. 54

6.1 Example images, depicting possible applications of the proposed methods. (a) Detection of all the pigs in the pen. (b) Heatmap of time-averaged location in the pen. (c) Tracking of individual pigs and their movements. 62

6.2 Some sample images from the first data set [BTK18]. Some piglets were marked with color patterns for individual identification for the original study. 68

6.3 Some sample images from the second data set [BGT+20]. The example shows the dense grouping of animals and the distortions during active night vision caused by dirt on the lens. 69

6.4 Examples of the synthetically generated training images. The 3D models of the pigs are randomly placed and the environmental conditions are randomly set. 70

6.5 Visualization of different types of detection approaches on an example image part (a). Ellipses (b) give a much better approximation than the classic bounding boxes (large overlap) (c) or keypoints (d) where the affiliation to the individual animals has to be resolved afterwards. [BGT+20] 73

6.6 Visualization of the five ellipse parameters (position of the centroid (x, y) , the two main-axis lengths (a, b) and the rotation of the ellipse θ). The motion vector \vec{h} is measured relative to the heading of the ellipse (a). The distribution of the motion vector length in data set #1 (b) shows the low average movement of the animals between two consecutive images. 74

List of Figures

6.7	Evaluation of motion statistics on data set #1. The scatter plot of all the movements with their distance and their direction (relative to the heading of the pig) reveals a high noise rate for the position and a nearly uniform distribution of the direction of the movement.	75
6.8	Two examples showing the problem of overlapping ground truth ellipses (a). The filled part of the ellipses shows the identified orientation of the animals. The pixel masks with correct depth sorting (b) and the ellipses obtained from them (c) allow a better comparison with potential detections (d). [BGT+20]	77
6.9	Example of automatic ground truth generation on synthetic images. The 3D objects are rendered for the input image (a) and simultaneously projected into a label image (c) (where identity is represented by different greenish color).	78
7.1	Example of the fragility of the categorical segmentation in case of strong overlaps. If the center of the animals is not visible, the segmentation cannot provide meaningful information about the hidden animal (b). The instance segmentation, on the other hand, does not have this problem (c). [BTK15]	80
7.2	Plots of the five degrees of freedom in the fitness-function, showing the multi-modality in the first two dimensions (position of the proposed ellipse).	82
7.3	Illustration of the optimization. The optimizer starts with the last valid position, which is slightly off (a). In each iteration samples are evaluated and the internal state is updated (b). After convergence, the final result is found (c). [BTK15]	83
7.4	Overview flow-chart of the proposed fitness-function. The resulting fitness value is a weighted sum of four components.	84

7.5 Results of different approaches for binary foreground segmentation. Classical threshold-based approaches have difficulty with barn equipment (b, c), while movement-based approaches fail because of the animals' lack of structure (d). Segmentation networks can overcome these difficulties (e). 85

7.6 Cost-map generation as in [BTK18]. Due to tight masking of the input image (a) the binary segmentation based on histogram equalization (b, c) is a reasonable foundation for the generation of the cost-map (d). 86

7.7 Visualization of the recovery process as described in [BTK18]. By expanding the search area (b), local maxima can be found quickly (c), indicating untracked animals. 90

7.8 Visualization of the detection results based on the corresponding cost maps. (a) The simple binary segmentation based on the color information results in errors in the cost map and misleading information, which ultimately cause erroneous detections. (b) The much better segmentation done with the neural network does not show these problems and is close to the segmentation with ground truth data (c). 92

7.9 Plot of relationship between detection accuracy and animal overlap for the 500 frames of tracking sequence #1. Overlap is calculated as a normalized value across the sequence. 94

7.10 Plot of average runtime and number of fitness-function evaluations over the population size of the CMA-ES optimizers. 96

8.1 Visualization of the different steps of pixel segmentation for an example image (a). The binary segmentation (b) distinguishes only between foreground and background. The center segmentation can be used to separate the individual animals (c) or the classes are defined to identify body parts (d). Or the network is trained to directly tell the affiliation of the pixels to the individual animals (e). Images from [BGT+20] 98

List of Figures

- 8.2 Schematic representation of the used modular framework [BGT+20]. The auto-encoder produces an intermediate representation (U-Net architecture depiction adopted from [Yak19]), which is passed through different heads. The final network output is processed afterwards to yield the desired results. 99
- 8.3 Example of fragility of center segmentation in the presence of heavy overlap. When the center of the animals is not visible, the segmentation cannot provide meaningful information about the hidden animal (b). While instance segmentation (c) can theoretically provide unambiguous assignment in such cases. 106
- 8.4 Illustration of the forces acting on the pixels to form the clusters (image adopted from [DNV17]): The variance term (yellow arrows) is used to pull the pixels towards the cluster center (crosses) while the distance term (red arrows) ensures a minimum distance between the different clusters. Both forces are only active as long as the threshold values are not reached (inner circle for cluster variance and outer circle for distance) [BGT+20] 107
- 8.5 Visualization of the clustering of the combined segmentation. The original input image is shown on the left with the ground truth label below it. The top row depicts the two-dimensional embedding space. The bottom row shows the corresponding binary segmentation and cluster assignment. Snapshots are taken after 1, 2, 3, 10, and 80 gradient updates. The network was trained exclusively on this one input image to produce the results shown here for illustration. 110
- 8.6 Example of bipartite matching between successive frames. In time step $n + 2$ a false assignment occurs, which is caused by a false-negative detection (faded blue ellipse) and a false-positive detection (faded gray ellipse) occurring simultaneously. 113

8.7	Graphical illustration of the hierarchical application of bipartite matching. In the first step, local linking is performed between pairs of images. In the second step, tracklets are also linked across multiple time steps to close the gaps. . . .	115
8.8	Example of a flow network for the min-cost flow method. Each observation is modeled internally with two nodes and is connected to all observations in previous and following time steps (within a time window).	116
8.9	Demonstration of the results of the different detection methods on an example image (cropped) (a). Shown are the results of binary segmentation (b), center segmentation with the classes <i>outer edge of an animal</i> and <i>inner core of an animal</i> (c), body part segmentation for orientation recognition with the classes <i>head</i> and <i>rest of body</i> (d) and the combined segmentation (e). [BGT+20]	119
8.10	(a) Evaluation of different thresholds δ_v and δ_d which control the distances in the discriminative loss. Measured in panoptic quality (PQ). (b) Evaluation of the minimum cluster size (on the reduced 320×256 px images) showing a stable interval in the range 80 to 200. [BGT+20]	124
8.11	Visual results from the test set of data set #1. Each row shows the original image (a), the prediction of the center segmentation (b), the ellipses extracted from the center segmentation (c), the prediction of the combined instance segmentation (d), and the ellipses extracted from the combined instance segmentation (e) (including orientation recognition where the filled part of the ellipses shows the identified orientation of the animals.)	127
8.12	Visual results from the test set of data set #2. Each row shows the original image (a), the prediction of the center segmentation (b), the ellipses extracted from the center segmentation (c), the prediction of the combined instance segmentation (d), and the ellipses extracted from the combined instance segmentation (e) (including orientation recognition where the filled part of the ellipses shows the identified orientation of the animals.) [BGT+20]	128

List of Figures

- 8.13 Failure cases in the assignment of the image points to the animals present in the image during instance segmentation. The membership of the image pixels to the clusters is indicated by the color (d). Since the subsequent ellipse extraction tries to combine all pixels belonging to a cluster with the ellipse, false detections occur (e). 129
- 8.14 Failure cases where center segmentation fails. If the animals cross or are too close to each other, the pixels can merge with the "inner core" class of different animals or the pixel of a single animal can be separated into two parts (b). This disturbs the extraction of the ellipses (c). [BGT+20] 129
- 8.15 Visual representation of tracking results on network detections (ISN) for tracking sequence #1. Assigned target IDs are represented by colors to show continuity or interruptions of each trajectory. 133
- 8.16 Visual representation of tracking results on network detections (ISN) for tracking sequence #2. Assigned target IDs are represented by colors to show continuity or interruptions of each trajectory. 134
- 8.17 Visual representation of tracking results on network detections (ISN) for tracking sequence #3. Assigned target IDs are represented by colors to show continuity or interruptions of each trajectory. 135
- A.1 Visualization of the classical and the dilated convolution with highlighted input pixels (blue) which were taken into account for the calculation of the one output pixel (cyan). [DV16] 144
- A.2 Visualization of high-level network modules, to easily train very deep architectures or use different receptive fields. Depictions adopted from [HZR+16; SLJ+15] 145

B.1	Schematic visualization of the CMA-ES optimization process evolving over generation g with population-size $\lambda = 19$ and $l = 2$ parameters. The background shows the fitness function value (brighter = higher). Also shown are the covariance matrix C (green ellipse), the mean m (blue dot) and the sample points (population) according to C and m (red points). [BTK15]	148
-----	--	-----

List of Tables

6.1	Information about the two data sets. Since the images are from different studies, they differ in some respects.	68
6.2	Split of the available data into training, validation and test data. In addition, the number of day and night vision images are listed.	72
6.3	Evaluation of the overlap of individual animals based on the manual annotations.	77
7.1	Detection results of the proposed detection-free tracking method. Only the overlap of detections and ground truth positions and no target identity is evaluated. For all metrics the threshold for the <i>IoU</i> -Score was set to 0.5.	93
7.2	Tracking results of the proposed detection-free tracking method according to the metrics presented in Section 5.2. The evaluation is performed separately for three different segmentation methods. All experiments are repeated 10 times each, so the reported values are the mean with the corresponding std.	95
7.3	Continuation of the evaluation from Table 7.2 with the metrics <i>mostly tracked</i> (MT), <i>partially tracked</i> (PT) and <i>mostly lost</i> (ML). For details see Section 5.2.	95
8.1	Accuracy results of the binary segmentation experiment. The experiment was additionally carried out separately on the daylight (D) and night vision (N) images of the second data set. Best results in bold.	120
8.2	Categorical segmentation accuracy and detection metric results of the center segmentation experiment. Best results in bold.	120

List of Tables

8.3	Categorical segmentation accuracy and detection metric results of the center segmentation experiment carried out separately on the daylight (D) and night vision (N) images of the second data set.	121
8.4	Detection metric results of the instance segmentation with the combined approach. The accuracy is given for the binary segmentation, produced by the segmentation head.	121
8.5	Detection metric results of the instance segmentation with the combined approach carried out separately on the daylight (D) and night vision (N) images of the second data set. The accuracy is given for the binary segmentation, produced by the segmentation head.	122
8.6	Results of orientation recognition: The network can correctly recognize the orientation in 88% resp. 94% of the correctly found animals (true positive).	122
8.7	Results of the ablation study on the test data of dataset #2. Only the combined segmentation with a reduced image resolution of 320×256 pixels was evaluated. The results illustrate the marginal impact of the different architecture choices (U-Net vs. Feature Pyramid Network (FPN)) as well as the different backbones. Results from [BGT+20]	123
8.8	Results for a five-fold cross-validation on data set #2. In each experiment, one of the cameras was declared as the test set while the images from the four remaining cameras were used for training and validation [BGT+20]	125
8.9	Evaluation of instance segmentation on the test set of data set #2. During training, synthetic data were mixed in to artificially increase the variance of the input samples.	126
8.10	Tracking results of the bipartite matching tracking method. The evaluation is performed separately on detections from an instance segmentation network (ISN) and on the ground truth positions (GT) as a baseline.	131
8.11	Tracking results of the global bipartite matching tracking method. The evaluation is performed separately on detections from an instance segmentation network (ISN) and on the ground truth positions (GT) as a baseline.	132

8.12	Tracking results of the minimum-cost flow tracking method. The evaluation is performed separately on detections from an instance segmentation network (ISN) and on the ground truth positions (GT) as a baseline.	132
8.13	Evaluation of correct target assignments with the used transition cost generation. The accuracy is calculated on the ground-truth data for different time-gaps on all three tracking sequences.	136
8.14	Runtime evaluation of the proposed experiments for categorical segmentation and combined instance segmentation: All values are given as mean runtime and standard deviation over the 226 images of the test set of data set #2.	137
8.15	Comparison of the runtime of the three methods shown on the three tracking sequences. For sequence #3 with twice as many animals in the image, the runtime of all methods increases accordingly.	138

Introduction

Year by year the global demand for meat increases. Between 1990 and 2014 the amount of meat-products from pigs increased by 65% from 69,7 to 115,3 million tons [Sta18]. Germany is one of the largest exporters of pork and over 25 million pigs are housed in Germany alone. Even though the herd has decreased slightly in the last 10 years (approx. 4%), the number of farms keeping pigs has decreased by almost 40% in the same period [Sta20]. This centralization of rearing to fewer and ever larger farms means that the individual farmer has to look after more and more animals. At the same time, consumers are becoming more concerned about animal welfare, so that animal welfare is increasingly becoming a focus of public attention.

In order to enable farmers to achieve high animal welfare without economic losses, the influencing factors in industrial agriculture are increasingly also being studied in research. For example, there are studies on the influence of housing, space requirements, substrates or even enrichment to keep the animals active [WD09; VTH+16; NEM+17]¹, as non-optimal conditions lead to unnatural behavior, injuries and stress. [SG94].

To evaluate the influence of each studied factor on animal welfare, a metric must be used whose evaluation cannot falsify or influence the results of the study. One possibility is the subsequent evaluation (e.g. based on injuries) during the slaughter process [BDK+19; BVK20]. The other way to evaluate welfare (on the running farm) is to analyze the behavior of the animals [UVK+14].

Assessment of conditions in the running farm has the advantage that such methods can be used not only in research but also in normal op-

¹Although the work described above mainly relates to the study of pigs, similar work has been done for other livestock.

1. Introduction

erations to monitor conditions continuously. Since it is important not to disturb the natural behavior of the animals, sensors are used for position and activity detection. These sensors can be applied either locally on the animal (e.g. as ear tags) or globally in the barn (e.g. in the form of cameras). A good overview of the possibilities for automatic monitoring, as well as indicators for behavioral analysis in pigs, is provided by [MMC+16]. Ear tags or collars can report the position of the animal, but since the transmitter is attached to the head, this does not allow any conclusion about the general orientation of the animal. In addition, the sensor must be purchased and maintained for each individual animal. For this reason, computer vision is increasingly being used, where a few cameras can monitor the entire barn with all the animals. An overview of different applications using computer vision in livestock can be found in [NES17].

1.1 Motivation

In this thesis, methods for visual detection and tracking of pigs in livestock production are presented and evaluated. Motivated by the increasing use of cameras in research facility and conventional barns, visual tracking offers a non-invasive and low maintenance option for behavioral research. If the position and orientation of the animals in the images are reliably detected, versatile evaluation options arise. For example, the position of the animals extracted from the images can be associated with specific areas in the barn and thus actions such as feed intake can be inferred [KBH+13]. If animals come close to each other or touch each other (mounting and chasing), this can indicate interaction or even aggression [VIO+14; LJP+16; NHE+16a]. In addition, the behavior of the entire group can also be evaluated. As an example, the temperature in the stable changes the pattern with which the animals distribute themselves in the pen for resting [NRH+15a]. Or by tracking the animals, the change in position over time can be incorporated into an activity index [OMK+14] or locomotion analysis [KBO+14].

Although the topic of visually tracking objects in images is a well researched area, tracking animals in confined spaces presents some difficulties. And while the relevance of tracking was recognized early on, the difficulties were also highlighted:

1.1. Motivation

A further current research topic in image analysis, which has relevance to monitoring animal behaviour, is the segmentation and tracking of moving objects. Tracking animals is difficult, because they are difficult to identify and their movements are often unpredictable.

Frost et al. [FSB+97], page 151

So the research question addressed in this thesis is whether it is possible to enable robust tracking of animals based on camera images alone, despite the difficulties mentioned above. Overall, there are three main issues specific to this problem, all of which present some challenges:

- ▷ Special camera perspective
- ▷ Limited space, which results in unintentional or influenced movements of the animals
- ▷ Visual indistinguishability of the individual tracking objects

First and foremost, in experiments with animals that involve observing behaviour, care must always be taken to ensure that the recording conditions are such that they do not affect the normal behaviour of the animals. Otherwise this would falsify the results of the behavioural study. As a result of this limitation, the camera position cannot always be selected optimally, and the field of view may be limited. With larger test setups, often certain areas such as transitions between individual zones cannot be seen or the camera cannot completely cover a larger observation area. This results in animals appearing or leaving the monitored area. Complete or partial covering of animals can also be caused by structural or supply structures such as heat lamps or the feeding system. Occlusions are also caused by behavioural patterns such as crawling over each other or crowded lying. The lighting of the scene cannot be chosen as desired either, as light can influence the behaviour of the animals. In cases where the animals need natural light the tracking method needs to cope with a wide variety of lighting situations. This for example is the case when the observation takes place in livestock environments. Besides the challenging lighting conditions also dust and dirt in the barn is often a problem. Dirt on the camera lens makes the detection of the animals more difficult or reduces the lifespan of the hardware.

1. Introduction

In economically managed breeding farms, the high occupancy of the individual stables can lead not only to the occlusion of individual animals, but also to evasive movements due to lack of space. This means that the movements of the animals are not only based on their natural behaviour, but are caused either by the stable size or by other animals. Instead of clear straight paths, the animals take detours or are suddenly startled or displaced in a resting situation. Such movements are difficult to predict as they are not logical but are caused by external factors. Such impulses from outside lead to movement patterns, which may be untypical for the species and are difficult to model with movement models. The narrowness of the habitats can also lead to conflicts between the animals, which end in bullying or even attacks. Such interactions or collisions are important in two ways. Firstly, they are to be seen as a characteristic of social group behaviour and secondly, the contacts of the target objects are a critical state with regard to the detection and tracking of individual animals. Such close touches always carry the risk of a target exchange, where the tracking procedure exchanges the tracks of two different targets.

The swap of tracking targets is especially possible if the animals are visually hardly different. In case of occlusions and without individual distinctions it is not possible to follow an animal over a complete sequence without any doubt. Leaving and appearing in the monitored area can also lead to interrupted trajectories, which cannot be assembled afterwards if the animals cannot be identified beyond doubt.

1.2 Contributions

Many parts of this work are based on interdisciplinary collaboration between computer science and agricultural sciences or veterinary medicine. Even if the primary research questions about animal welfare or the behavioral effects of various environmental variables tend not to be answered by computer science, its contribution is the basic building block that makes the methods presented here possible. As motivated in the previous section, it is only with robust detection and tracking of animals that automated processing of images, and thus analysis of significant amounts of data, becomes possible. My contribution to this research is therefore dedicated

to this very topic and in addition to Part II in this thesis, it has already been described in the following publications:

- ▷ Johannes Brünger, Imke Traulsen and Reinhard Koch: *Randomized global optimization for robust pose estimation of multiple targets in image sequences*, in *Math. Model. Comput. Methods*, Volume 2, 2015
- ▷ Johannes Brünger, Imke Traulsen and Reinhard Koch: *Model-based detection of pigs in images under sub-optimal conditions*, in *Computers and Electronics in Agriculture*, Volume 152, 2018
- ▷ Johannes Brünger, Maria Gentz, Imke Traulsen and Reinhard Koch: *Panoptic Segmentation of Individual Pigs for Posture Recognition*, in *Sensors*, Volume 20, Issue 13, 2020

Applications of the described methods were shown in the following publications:

- ▷ Steffen Küster, Matthias Kardel, Stefanie Ammer, Johannes Brünger, Reinhard Koch and Imke Traulsen: *Usage of computer vision analysis for automatic detection of activity changes in sows during final gestation*, in *Computers and Electronics in Agriculture*, Volume 169, 2020
- ▷ Maria Gentz, Cornela Meckbach, Sebastian Zeidler, Vivien Loges, Johannes Brünger, Reinhard Koch and Imke Traulsen: *Activity behaviour of pigs: comparison of manual and automated video analyses*, Submitted to *Computers and Electronics in Agriculture*, 2020

In addition, a publication on the classification of injuries in the slaughter process is also available:

- ▷ Johannes Brünger, Sabine Dippel, Christina Veit and Reinhard Koch: *'Tailception': using neural networks for assessing tail lesions on pictures of pig carcasses*, in *Animal*, Volume 13, Issue 5, 2018

1.3 Outline

This work is structured as follows: Part I describes the basics and principles of detecting and tracking objects in images. Since these two disciplines of

1. Introduction

computer science are closely related and have a long tradition, the classical approaches are introduced as well as the basics of the methods used in this thesis. Part II then describes specific applications in the area of behavioral research in farm animals. Data sets from studies in the investigation of behavioral patterns in pigs are used to describe different approaches to robust detection of pigs in the barn, as well as tracking of the animals. The methods presented follow several of the approaches presented in the first part and thus give a good overview of their performance. For this purpose, the methods are also evaluated with established metrics in each case. The paper concludes with an outlook and suggestions for further possibilities to apply the rapidly developing technology of artificial intelligence in a more targeted way in this field.

Part I

**Principles of Detection and
Tracking**

Interrelationship of Detection and Tracking

In general, (multi-)object tracking has been extensively researched. The problem has a wide range of applications, from tracking pedestrians or vehicles in surveillance videos, to evaluating the performance of individual athletes, to biomedical research. Whether passers-by, athletes, cells or animals, it is always a matter of identifying individual targets, preserving their identity and creating an individual trajectory over the entire sequence of the input video.

For the setting described in the introduction, the task can be defined as follows. Starting from a sequence of images of the animals in their usual environment, all target objects are to be found and recognized over the entire sequence, so that from the resulting trajectory, the location of the individual animals in each individual time step can be determined. Thereby the only source of information for the whole process are the pixels of the video images.

The actual tracking problem consists of two sub-problems. Firstly, in each frame of the sequence the objects must be found respectively distinguished from each other (*detection*) and secondly, the objects in the individual frames must be correctly assigned to each other (*data association*) over the complete sequence. The two sub-problems can be addressed either in combination or separately. In the combined approach *Detection-Free Tracking (DFT)* the objects are manually initialized in the first frame and then localized in the subsequent frames, whereby the tracking-algorithm maintains an object-model of the tracked targets. This approach is best suitable for tasks with a fixed number of objects, as clues in the initial appearance are memorized and evolved over time. In *Detection-Based Tracking*

2. Interrelationship of Detection and Tracking

(DBT) the two sub-problems of detection and tracking are dealt with by different procedures. A separate object detector is used to independently detect the objects in each frame. These detections are then linked by the actual tracking method to form the final trajectories. These methods are suitable for situation with a changing number of targets but are, of course, heavily dependent on the performance of the object detector.

2.1 Problem Formulation

Adopting the definition of [LXM+14] the previous informal description can be formalized as follows: Given the image sequence that shows the objects to be tracked, the joint state of the N depicted objects in the t -th frame is defined as $\mathbf{S}_t = (s_t^1, s_t^2, \dots, s_t^N)$. For the i -th object the vector s^i stores the describing parameters like position, size, rotation etc. The sequential states $s_{i_s:i_e}^i = (s_{i_s}^i, \dots, s_{i_e}^i)$ define the trajectory of the object, where i_s and i_e are respectively the first and last frame in which target i exists. $\mathbf{S}_{1:t} = (\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_t)$ denotes all the sequential states of all the objects from the first frame to the t -th frame. Not every object necessarily has to be visible throughout the entire sequence. Therefore, the final trajectory of the object may have to be composed of several shorter trajectories. In *Detection-Free Tracking* the joint states \mathbf{S}_t are modeled, memorized and iteratively updated over time, using observations \mathbf{O}_t from the current (t -th) frame. In contrast the approach of *Detection-Based Tracking* is usually handled in two steps, where a detector first processes all the frames of the sequence and proposes M potential detections (or observations) $\mathbf{O}_t = (o_t^1, o_t^2, \dots, o_t^M)$ of the tracked targets in the t -th frame. The collected observations from the first frame to the t -th frame $\mathbf{O}_{1:t} = (\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_t)$ are then processed and linked together to form the final trajectories of the N tracked targets. Of course, the number of observations M does not have to match the number of true objects N , so that holes in the trajectories or completely wrong additionally existing trajectories can occur, which have to be evaluated accordingly in the evaluation (as described in Chapter 5).

2.2 Detection-Free Tracking

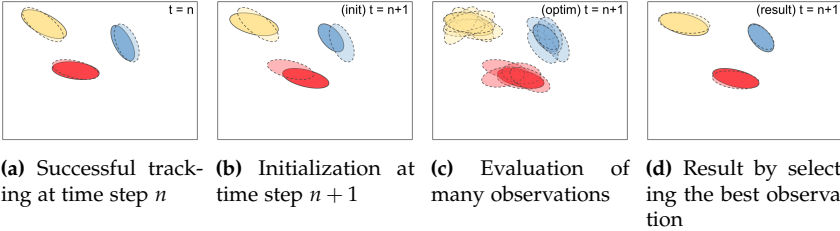


Figure 2.1. Schematic representation of *Detection-Free Tracking*. (a) The real objects are shown opaque, their tracked position semi-transparent. (b) In the next time step, the objects have moved and the tracked positions are obsolete. (c) Many observations are generated, based on the object model. (d) The best matching observation is chosen as the new tracked position of the object.

In *Detection-Free Tracking*, the problem of locating the targets and matching the observations to the already known targets are integrated in a combined approach. The detected targets are iteratively tracked at each time step, so that based on the current position and the new observations, the parameters of the targets are adjusted (see Figure 2.1). An *object model* (see Section 2.4) is maintained to enable the assignment, or to simplify the search for the position in the next frame of the sequence. This object model contains information about the appearance of the target, its movement or interaction with other targets, which allows to predict the possible next positions or to recognize the target. Using the data from the object model, the observations extracted from the current image can be evaluated to track the target. This evaluation is usually done with a likelihood function, which evaluates the probability that an observation has relevance for one of the tracked targets.

2.3 Detection-Based Tracking

In *Detection-Based Tracking*, the detections are generated separately from the actual tracking and subsequently combined by the tracking process

2. Interrelationship of Detection and Tracking

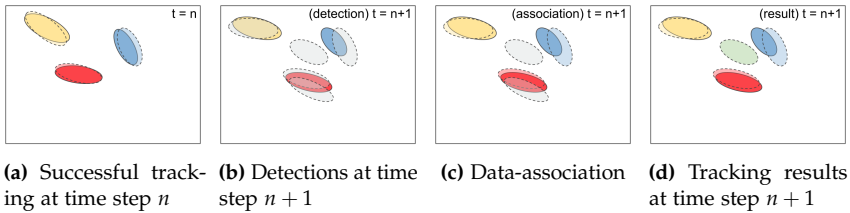


Figure 2.2. Schematic representation of *Detection-Based Tracking*. (a) The real objects are shown opaque, their tracked position semi-transparent. (b) In the next time step, observations are available in the form of external detections. (c) The detections are assigned to the already known targets. (d) The best association is chosen as the new position. If no assignment can be found, a new trajectory is generated (in this case a false positive).

to form the trajectories (see Figure 2.2). When creating the detections, the detector has no global overview and therefore cannot maintain an object model during detection. On the subsequent tracking, the *data-association* is used to link the provided detections to trajectories of single individual targets. In order to preserve the identity and to assign the detections to the correct targets, object models are also built up here and adapted over time. In the process, special cases must also be taken into account. For example, if there are more detections in the current time step than targets from the last time step, some detections must either be discarded during the association (e.g. by non-maxima-suppression), or new trajectories must be started (if a new target has entered the scene).

2.4 Object Modeling

Regardless of whether *Detection-Free Tracking* or *Detection-Based Tracking* is used, a key issue in multiple object tracking is the maintenance of the identities of the individual targets. If detections are available, they need to be associated correctly to the existing targets or in the case of detection-free tracking, the tracking algorithms must be able to find the target currently being tracked in the scene. If occlusions occur, the association is required not only on a frame-to-frame basis, but also over longer time intervals. So

2.4. Object Modeling

the algorithm must be able to rediscover a previously lost target. In the case of visual tracking, the input consists of visual observations, which in turn consist of the individual pixel values of the input images. However, the images are only a two-dimensional projection of the real three-dimensional scene and such dimension reduction often induces ambiguities and occlusions. In addition most cameras are subject to systematic errors in the process of image generation, which adds more difficulties such as overexposed pixels or noise in the image [HK94]. Tracking algorithms try to handle such sub-optimal input data by creating a model of the target to describe its features and discriminate it from the surrounding and the other tracking-targets. Such models can consist of features based on the appearance, motion or even interaction and makes the association of observations to individual targets possible. However, the detections can of course also contain errors. Detectors sometimes detect objects where there are none (false positive) or overlook objects (false negative). Tracking algorithms must therefore be able to handle such situations. Sometimes the detectors provide a confidence score, which the trackers can use to assess the detections.

2.4.1 Appearance Model

In visual tracking, the color and intensity values of the individual pixels together with their local distribution are the most important information for distinguishing the searched objects from the background and even identifying them individually. This should be as robust as possible, so that the distinction or identification works over a longer period of time in many successive images. The single pixel values are not very meaningful, because the object is represented by a set of pixels and the appearance is determined by their local distribution. Therefore, features are used that combine the pixels in a meaningful way. This grouping to features can be done manually or learned based on training data. However, the goal in creating the features is always to be able to use a comparison or evaluation function with which the image areas can be classified as possible observations of the target object.

2. Interrelationship of Detection and Tracking

Intensity and Color

The most straightforward way to visually describe an object is often by its dominant color. Accordingly, one can try to find an object in an image by identifying its dominant color and searching for a collection of pixels with that color. However, finding, for example, a "yellow object" is complicated by the fact that with classical color representations, each individual pixel can encode up to 16 million different colors, and yellows alone can have thousands of variations. Solutions to this problem can be color spaces that define the color value in a fixed range (e.g. the HSV color space), or learned color attributes ([VSV+09]) that mimic human color perception and map all representable colors to the eleven base colors ([BK69]). This makes it possible to search for the object with a simple comparison of similarity based on color. However, objects are not always monochromatic, so the use of a color histogram is a useful extension, as it allows the distribution of colors over the object to be modeled. Color histograms can be compared with metrics such as the *Bhattacharyya distance* [Bha43].

In order to group together the pixels, which are to be used for the description of the object, the spatial appearance of the object plays an important role. In the classical raster representation of an image, the pixels are arranged in a regular two-dimensional matrix. Therefore, the simplest definition of an image area or a subset of the pixels is also a rectangular submatrix. For the description of an object in the image, a bounding box is usually defined, i.e. the two-dimensional hull which, aligned with the coordinate system of the image, completely encloses the object. However, this results in the loss of information about the shape of the object (see the next subsection on shape and contours). In addition to the loss of shape information, the bounding box also contains a portion of pixels that do not belong to the object and thus dilute the appearance model. The big advantage of the rectangular boxes, however, is the performant evaluation of the contained color and intensity patterns. Therefore, the classical bounding boxes are used in many works for object detection. Depending on the application, the pixels in the bounding box are compared with the stored description of the object, e.g. with Normalized Cross Correlation or classification patterns like *Local binary pattern* (LBP) [OPH94] or *Random fern classification* [OFL07]. In [DSF+14] a kernelized least squares classifier

2.4. Object Modeling

is learned, based on the color features of the targets patch.

However, a major problem with bounding boxes that enclose the entire object is their translation and rotation invariance. If the bounding box is moved or rotated by only a few pixels, the relative positioning of the pixels within the box and thus their description of the object changes drastically. One way to counteract this is to use the color histograms already mentioned, which can be used to subsample or even completely ignore the spatial distribution of the pixels, thus reducing the invariance with respect to translation and rotation. Another way is to reduce the size of the bounding boxes and use several of them to describe individual parts of the objects. Examples are the part models or key points. With part models, the image regions are no longer assigned to the entire object, but to individual parts of the object, so that firstly a more specific description of the object parts can be found and secondly the spatial positioning of the parts among each other can be taken into account for the classification of the object. Deformable-Part-Models are described in Section 3.1 and use exactly this idea. Another method to reduce the influence of the background pixels within the bounding box, or to better react to deformations of the object, is the use of key-points. Key-points define important, easily recognizable points on an object, which are described by small patches. The patches are selected and described in such a way that they are as robust as possible to changes in illumination and rotation [Shi+94; Low+99; BTV06]. Similar to part models, these key points can then be grouped together to uniquely describe objects [BL02].

Shape and Contours

As described in the previous section, due to the performance advantage, a rectangular box is often used for generating the description of the object. However, unless the shape of the object being searched for also happens to be rectangular and exactly aligned with the coordinate system of the image, there will always be pixels within the bounding box that do not belong to the object. There are even examples where the background makes up a large part of the bounding box (long thin objects rotated 45 degrees to the image axes), so that a description of the object based on the contents of the bounding box would be falsified. In such cases, but also for many

2. Interrelationship of Detection and Tracking

other irregularly shaped objects, pixel-precise segmentation of the objects can solve the problem and provide additional information about the shape of the object. In the domain of image processing the task of segmentation is defined as the partitioning of the image into multiple segments, where the pixel in the individual segments share a common characteristic. These segments then can be used to identify object boundaries or highlight objects from the background. The simplest example of a segmentation is the binary segmentation of a gray-scale image based on a single threshold on the intensity value. If the threshold is applied globally to the whole image, the pixels above the threshold are classified as foreground and all remaining pixel are classified as background or vice versa. Of course, such a global threshold is very susceptible to changes due to shadowing, as the global threshold cannot adapt to local contrast conditions. An extension are therefore the adaptive (or local) thresholding methods, where the neighbourhood of the processed pixel is analyzed and the local threshold is adopted accordingly. There are also more sophisticated ways to calculate a global threshold like [Ots79] where the pixels of the two classes are clustered and the spread of the clusters is minimized. Using standard clustering algorithms like K-Means or Expectation Maximization (EM) on additional features such as color can also help to group the pixels into segments based on their appearance. Unfortunately, these clustering procedures are often not capable of correctly reproducing the spatial relationship between the pixels. Therefore segmentation was also solved with graph theory and Energy Minimization methods [SM00; HZC04]. With selective search [VUG+11; UVG+13] a hierarchical segmentation for object proposals was introduced which combined image areas based on multiple similarity metrics.

The shape of the searched object is also a useful feature if the object is completely textureless or monochrome. In such a case the significance of the color distribution for the classification of the objects is low and a classification via the contour can provide a lot of additional information. Likewise, there are object classes in which the color is present, but occurs in such high variance that it is also not suitable for classification. For example, in the detection of persons, the colors of the clothes are a feature that occurs in such high variance that it describes the individual person well, but is not suitable to describe the whole object class. The contour of

2.4. Object Modeling

a person in an upright position, on the other hand, is easily recognizable.

There are different ways to detect and describe the contour. The histograms for example can also contain gradient information (*Histogram of oriented gradients* or HOG [DT05]), which describes the shape contained in the patch. Occlusions would significantly affect the histogram, so another way to encode the contour is with splines [BB05; Zel14] where short line segments in the observed image are matched with the stored contours of the targets.

With depth sensors even the actual shape of the objects can be captured and used to detect and discriminate the individuals by region growing, based on their surface normals [MMP+17]. The same can of course be applied to classical color images. If the objects and the background are visually so different that the „objectness“ can be represented by features, the gradient or level set methods can be used to fit a contour model [PC14]. As with colors, it can also make sense with contours not to describe the shape exactly, but rather to choose an abstract representation. So in many cases, for example, only the position and orientation of the objects is stored, while the actual shape is abstracted with a model such as rotated rectangles or ellipses [KBD04; NHE+16b].

2.4.2 Motion Model

With the help of the appearance model and the associated similarity measure, the individual detections can be compared and assigned. In order to make the tracking even more robust, additional previous knowledge can be modelled. For example, if the tracked objects have a constant speed and direction of movement (linear motion), the next position can be predicted based on the current movement. With this prediction the unambiguousness of the assignment can be greatly increased. When people are tracked, for example, the assumption can be made that they will not suddenly change their destination. Therefore good results can be achieved with a linear model [BRL+09]. But also more complex non-linear models were successfully applied to people tracking, to make the assignment of detections more robust [YN12]. In addition to the possibility of using the motion model to simplify the assignment based on appearance, there are also descriptors which are based on the movement and which can then be

2. Interrelationship of Detection and Tracking

compared directly. The *Aggregated Local Flow Descriptor* (ALFD) encodes the relative motion pattern between two bounding boxes [Cho15] and enables the measurement of affinity between two detections directly. It is based on the optical flow of keypoints (local interest point), and evaluates the relative position of the points in the corresponding bounding boxes over time.

Animal movement in confined environments is often restricted (see Section 1.1). Therefore, complex motion models may not be applicable in a meaningful way in such cases. However, even the assumption that animals move only a small distance within a time step can be used as a simple motion model.

2.4.3 Interaction Model

Motion models can express the probability of a future position. However, there are also situations where the future position was planned but can never be reached by the tracked object. This can happen due to a decision made by the object itself or due to external influences. On the one hand, the object may encounter fixed obstacles and on the other hand, other objects may be on the way which block the desired position or cross the path to it. To avoid collisions, people would adjust their direction of movement in such cases. In animals, social interactions may occur in such cases. Conversely, people and animals behave differently in groups than when they are travelling alone. They adapt their direction and speed and pay attention to a certain distance to the surrounding group members. To take these cases into account, the motion model can be extended by an interaction model. If the interactions are rare, the motion model can, for example, be extended by a term as soon as the objects come close. Since the term uses additional resources, it makes sense to define it only for such cases. For example, a pairwise Markov random field (MRF) can be used to penalize overlapping of individual targets when determining the joint state \mathbf{S} [KBD04].

The linear motion model can also be extended to minimize the probability of collisions [PES+09]. For this purpose, it is assumed that the persons being tracked keep a certain distance, based on the knowledge of the current position and speed of the other persons. Based on this

2.4. Object Modeling

assumption an energy function is defined and its parameters are trained using examples. When tracking, this energy function is then minimized to determine the most probable trajectory of the targets.

Detection

Visual detection describes the task of finding one specific or multiple objects of the same class in an image. As mentioned in Chapter 2, detection is a basic element of tracking, because for successful tracking the target objects must be found in as many images of the sequence as possible. Therefore the tracking result depends significantly on the quality of the detection. For example, the tracking procedure would have to deal with gaps in the sequence if the detection fails. On the other hand, false positives can be generated by the detector, where detections of objects that are actually not present are generated. All this has a significant impact on the quality of the final trajectories found.

In addition to detection, the second important component of tracking is the assignment of the detections to the targets. Depending on the method used, the system either searches for the next position for one of the tracked targets (detection free tracking, see Section 2.2), or it first lists all detections independently of the targets and then assigns them to one of the targets or discards them again later (detection based tracking, see Section 2.3). In the first case the object model of the target can be used to search for the target, while in the second case a kind of objectness [ADF12] has to be evaluated, which distinguishes all possible target objects from the background. In the formal definition of the tracking problem in Section 2.1, the detections are listed as observations \mathbf{O} , which represent possible positions of the targets, but can also contain false detections. The generation of the observations \mathbf{O}_t in a given image I_t in the t -th time step is basically a classification, where all possible positions P are evaluated with a quality-function f . The position at which the quality function returns the highest value is then marked as a successful detection, or the detection fails if a certain threshold value is not reached.

3. Detection

$$\mathbf{O}_t = \underset{P \subseteq I_t}{\operatorname{argmax}} f(P) \quad (3.0.1)$$

There are many different types of detection, which differ in the recognition features used and correspondingly in the quality function. However, the big challenge in detection is always the large search area, since the quality-function has to be evaluated in all possible object positions. Thereby, it depends mainly on the complexity of the features to be evaluated.

Performance As described in Section 2.4.1, the appearance model of the searched targets can be structured very differently, but in most cases simple rectangular bounding boxes are used to define the search area. In such cases, a search window of the same size must be used or the search may have to be performed several times with different search windows (or differently scaled input images) to cover changes in size of the target. The simplest method to search for the possible positions is an exhaustive search over the entire image area (sliding window). An important point in the selection of the quality-function is therefore the efficiency with which it can be evaluated, which in turn contrasts with the complexity of the evaluation. With traditional detection methods, which rely on sliding windows as search methods, the required efficiency was achieved, for example, by exploiting the mathematical structure of the feature [VJ01] or by executing different complex quality-functions in a cascade [KMM12]. Another approach is the optimization of the subwindow search [LBH08].

Hand-crafted vs. learned Features Since the visual appearance of searched objects is extremely diverse, early detection methods based on static templates or hand-crafted features quickly reach their limits. The solution for the traditional detectors were enhancements with machine learning methods where e.g. linear classifiers are trained to distinguish positive from false positive detections based on a training set [VJ01; DT05; FGM+09]. With the successful introduction of neural networks, the classical methods of machine learning were replaced by methods based on neural networks. However, the design of the network architectures follows the previously established approach of a combination of region suggestions and a subsequent classification [SEZ+13; GDD+14]. In the same way, object detection

is also solved with segmentation networks [DHL+16; UCF+16]. A detailed overview of the development in the field of object detection has been put together by Zou et al. [ZSG+19].

3.1 Traditional Detectors

Individual targets can usually be recognized and identified by their appearance and individual visual characteristics. However, if all objects must be found (based on their class), their visual appearance is much more diverse and the object model must be defined more generally as a consequence.

One of the first successful approaches to generate a general representation of features of a given class was the eigenfaces for face recognition by Sirovich and Kirby [SK87] and its application by Turk and Pentland [TP91]. The general representation of human faces was generated in advance on a training dataset of face images. Assuming that the relevant features such as eyes and mouth are approximately at the same positions in the training images, the individual pixels can be interpreted as dimensions in a high-dimensional space, where each face-image is defined as a point in space, according to its pixel-values. A principle component analysis (PCA) can then be used to identify the components that best represent the variance of different faces in the distribution of all faces. The M features with the largest variance (or the eigenvectors with the largest eigenvalue) then span a M -dimensional *face space*. On this subspace, a quality function can then be defined to evaluate individual image sections. To determine whether the selected section contains a face, the linear combination of the pixels of the image section with the M selected eigenfaces is calculated. The coefficients can then be used to calculate the positioning in face space and accordingly determine how similar the current image section is to a face.

The processing of concrete pixel values and their relative positioning to each other within the window, is very sensitive to illumination effects and viewing angle changes. Even with a slight rotation of the image, the faces end up at completely different positions in face space. To abstract from the actual pixel values of the images and detach from the evaluation of the exact match in positioning, Viola and Jones used [VJ01] features

3. Detection

based on rectangular Haar wavelets filters to detect faces. These filters respond to the characteristic differences in brightness between different facial segments and can be evaluated very quickly due to their simple structure (see Figure 3.1a). This allowed for high accuracy at moderate computational cost. Although the authors limited themselves to the most significant four features, a complete evaluation of all combinations of these rectangular filters on individual subsections of the image region being evaluated would have required over 180k evaluations. The major contribution of Viola and Jones' work is therefore the introduction of a classification cascade.

They do this by using each of the features as a weak classifier to generate multiple ensembles and thus strong classifiers using the AdaBoost meta-algorithm. Since in normal images in the general case there are much less image parts showing faces than image parts with background, in the first stages of the cascade classifiers were used, which can sort out non-face images with a high accuracy and a low false positive rate. As these classifiers only need to use a few features, they can be evaluated very quickly and thus directly sort out a large proportion of the image sections. Only when these classifiers pass a section, it is processed and passed through in the cascade of more complex classifiers. This means that only a few image sections are examined with greater effort. The idea of using a cascade to sort out as many image sections as possible was also adopted in later work. In TLD (Tracking-Learning-Detection) [KMM12] the variance of the gray levels in the image section is evaluated in a first step to eliminate homogeneous background regions. If the image section contains enough variance, it is evaluated by an ensemble of classifiers, which compares the match of the current image section with an object model learned from previous successful detections. Only in the last stage, the complete image section is compared with a nearest neighbor classifier using the Normalized Correlation Coefficient (NCC) with stored representations of the object (see Figure 3.1b).

Since the appearance of an object can change massively due to changes in illumination and occlusions, the comparison based on the complete image section containing the object is not only difficult in terms of performance, but the dependence of the individual pixels on each other is usually not very meaningful. In such cases, detectors that do not consider

3.1. Traditional Detectors

the entirety of the objects (global appearance modeling) but evaluate individual parts and regions or special points of interest (local appearance modeling) provide better results. Examples of this can be found in related tasks such as object recognition or image registration. Instead of tracking the motion of objects, these tasks involve estimating camera motion or retrieving objects in images from different camera positions. If the angle of view of the camera changes, the displayed objects are often subject to large visual changes, so that it is particularly important here to identify the image areas that have a high degree of uniqueness and thus recognizability.

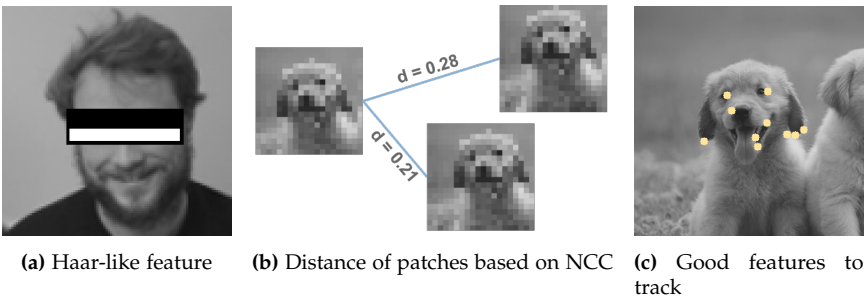


Figure 3.1. Visualization of examples of traditional detection methods. (a) Rectangular feature to evaluate the difference in brightness between the eye area and the upper cheek [VJ01]. (b) Stored representation and its distance to potential detections (with NCC) [KMM12]. (c) Points with high redetection probability for tracking [Shi+94].

To reconstruct the movement of the camera in relation to the depicted scene, it is sufficient to identify any object-independent points in the scene whose position can be reliably found in further images of the same scene. In particular, vertices were identified, since they can describe camera motion in two directions, while edges do not reveal motion in the edge direction. [HS+88]. If the distances between camera poses are small, feature points can be tracked more easily. For this, Thomasi and Shi [TK91; Shi+94] studied the relationships between the trackability of image patches (or features) and the choice of image patches (see Figure 3.1c). On the one hand, the shifts of brightness gradients can be used to estimate the shift of image-patches between two consecutive images. On the other hand,

3. Detection

the perspective distortions have to be removed with the help of an affine transformation in order to reliably track the image patches over longer sequences. This allows a meaningful comparison between the currently found image-patch and the original template. Since the affiliation of the selected features to individual objects is difficult to determine, the resulting KLT tracker (Kanade–Lucas–Tomasi) based on this is rather used for estimating the camera movement with respect to the imaged scene.

In order to recognize and track individual objects, however, features can be selected on the objects in order to find them again in another image. It is particularly important that the selected features can be recognized even under large changes in viewing angle. Brown and Lowe [BL02] have used groups of interest points, which allow the transformation of an image area into a canonical frame by their geometric arrangement. The image section projected into the frame then serves as a descriptor, which is thus robust against rotation and scaling. The individual interest points are also based on scaling-invariant descriptors and enable the retrieval of prominent points even under strong changes in viewing angle or illumination [Low+99]. The interest point groups can be used not only to estimate camera movements, but also to find objects by generating features from a template and matching the corresponding features in the search image. With the combination of multiple interest-points to one objects-descriptor, the recognition of objects becomes robust against the influence of occlusions.

Part models, which solve object recognition via flexible constellations of parts, work in a similar way. The advantage is that the parts can be recognized individually and thus a large variability in the appearance of the objects can be modeled. Since the composition of the objects from the parts is learned based on training data, not only the appearance of the individual parts can be learned, but also the positioning and the probability of the presence of a part. The parts are mostly defined as contiguous regions in the image. These can be determined either based on features such as histogram of gradients [DT05] (see Figure 3.2b) or visual saliency [KB01] as used by classical detectors, or with segmentation methods based on visible contours [GLA+09; FH04] (see Figures 3.2c and 3.2d). Based on the selected regions, object models can be defined to learn the searched object classes. The model parameters contain appearance information

3.2. Detection with Neural Networks

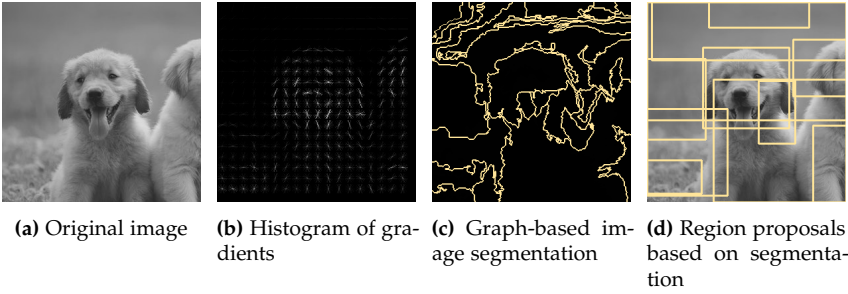


Figure 3.2. More visualizations of traditional detection methods applied to an example image (a). (b) Histogram of gradients [DT05] for feature description. (c) Graph based segmentation for part models [FH04], and (d) region proposals based on the segmentation from (c).

about the individual parts but also about relative positioning of the parts among each other or to a root part. The parameters of the model can then either be statistically modeled with probabilities [FPZ03; FH05] or learned [GLA+09; FGM+09].

3.2 Detection with Neural Networks

Traditional detectors initially used features hand-selected by researchers that were specific to the objects they were searching for (e.g., eigenfaces). Gradually, the manual features were then replaced by learned features, as these can statistically extract the true significance of individual features. With the increasingly significant performance of neural networks in the field of image classification [KSH12], their use as feature extractors in object detection methods also became interesting.

3.2.1 Deep Neural Networks

As with all other machine learning techniques, the use of neural networks aims to replace the hand-engineered features and extract the most important features directly from the raw data. This is because the almost infinite

3. Detection

variability of natural input data, cannot be processed without a meaningful internal representation. Deep neural networks are a combination of individual modules connected in series, each of which learns an internal input-output function and thus transforms the passed representation into a higher and more abstract representation. Despite the simple structure of the individual modules, which only learn a non-linear mapping of input to output, the network as a whole can learn extremely intricate functions [LBH15].

Although the basic idea and the techniques used had been known for decades, ways had to be found to adapt the parameters of the many layers connected in series by means of back-propagation according to their contribution to the final output [RHW86; Sch15]. Another important milestone was the adaptation of the techniques to graphics hardware, which could accelerate the computation by massive parallel execution [RMN09]. This allowed the use of larger data sets [DDS+09] and deeper network architectures, which led to the first successful applications and the breakthrough of the technique [KSH12].

Convolutional Neural Networks In classical implementations of neural networks, the individual neurons were defined as the weighted sum of all neurons in the previous layer. Each neuron thus received its input from all units of the previous layer (fully connected). However, this leads to two problems when processing images and thus the individual pixels as input data. First, the networks are not invariant to transformations of the input data and second, the spatial distribution of the information in the image is ignored [LHB+99]. *Convolutional networks* address these two problems by limiting the spatial perception of individual neurons and connecting them only to neurons in the small neighborhood in the previous layer. By forcing the units of a layer to share weights, feature detectors are created that operate independently of the position of features in the image. If the units of a layer are evaluated sequentially, it is similar to convolving the input with the weights. The resulting feature map can then be used by subsequent layers to assemble it with further feature maps into higher ordered features. [LBB+98]. This way of processing images with layers of neural networks, has enabled important breakthroughs in applications such as image classification [LBB+98; KSH12].

3.2.2 Different Approaches with Neural Networks

As with traditional detectors, there are also different approaches to visual detection with neural networks. *Detection networks* mimic the classic two-phase detection approach by first extracting regions of interest (RoI) and classifying them individually in a second subsequent step. Later, architectures were added that perform area extraction as well as classification in a combined step. However, all these methods are limited to the classical rectangular bounding boxes (see Figure 3.3a). In contrast to this, there are methods that approach the detection of individual objects via segmentation, and can thus generate pixel-precise representations of the objects. For example, *segmentation networks*, which assign a class to each pixel in the input image, can be used to apply and extend the approaches of traditional segmentation. In the task of *semantic segmentation* the pixels are classified individually and each pixel is assigned a class. All objects of the same class receive the same label (see Figure 3.3b). In order to also detect the individual objects, the semantic segmentation must be enhanced by an *instance segmentation* (see Figure 3.3c). This combination was described in [KHG+19] as *panoptic segmentation*, where the semantic segmentation estimates the class and the instance segmentation distinguishes the individual objects. If specific parts of the objects are defined instead of object classes, the segmentation networks can also act like part models and the searched objects can be composed as a collection of key points (see Figure 3.3d).

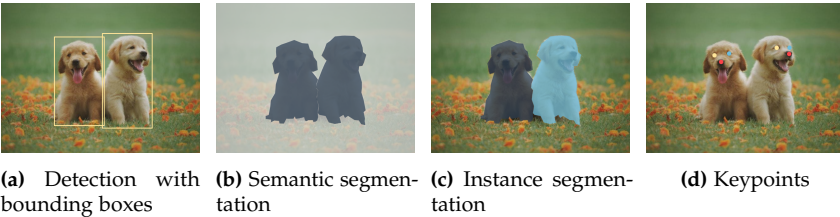


Figure 3.3. Examples of different approaches used for object detection with neural networks. (a) Classical bounding boxes based on region proposals. (b) Semantic segmentation by assigning each pixel to a class. (c) Instance segmentation by assigning each pixel to an object instance. (d) Key point detection to find features and subsequently assign them to instances.

3. Detection

3.2.3 Detection Networks

The use of a neural network for the detection of objects was proposed in [GDD+14] with the *R-CNN*. The authors combined a convolutional neural network with region proposals which were retrieved from a selective search algorithm [UVG+13]. The network was based on the AlexNet architecture [KSH12] (one of the first classifier networks to win a computer vision competition), and was used as a feature extractor for a downstream Support-Vector-Machine classification. This approach outperformed the *OverFeat*-network [SEZ+13], which exploits the translation invariance of the convolutions to efficiently search the input image for objects as with a sliding window. A method based also solely on neural networks was proposed by [STE13] who trained an AlexNet-like architecture to predict multiple binary masks, which can be combined to bounding boxes for all visible objects in the scene. In later works [EST+14; SRE+14] the *MultiBox*-architecture developed out of it, which is able to output the coordinates of bounding boxes directly. This popularized the idea of the *Regional Proposal Network* (RPN), which is built into the successors of the R-CNN approach *Faster R-CNN*. In *Faster R-CNN* [RHG+15] the network uses a shared base network to generate region proposals (defined as displacement relative to predefined anchor-boxes) and afterwards classify the sparse set of proposed regions of interest. The two parts of the network are trained thereby end to end in an alternating fashion (see Figure 3.4a). To simplify learning and speed up prediction, there are also approaches that replace the classification on each of the proposed regions with a single fully convolutional network [DLH+16].

This two-stage procedures are countered by the one-stage procedures, where a single convolutional network is used to do the localization and classification in a single step simultaneously. With YOLO [RDG+16] the input image is divided into a $S \times S$ grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts a set of bounding boxes with the position of the object relative to the grids center, the size of the box and a confidence score. In addition each grid cell also predicts a conditional probability for all classes (see Figure 3.4b). The *Single Shot Detector* (SSD) [LAE+16] works very similar to this. At each position of the output, a set of default boxes with different

3.2. Detection with Neural Networks

aspect ratios and sizes is evaluated (as in Faster R-CNN). The output of the network is then the displacement of the object bounding box to one of the default boxes as well as a probability distribution to which class the object in the box belongs (see Figure 3.4c).

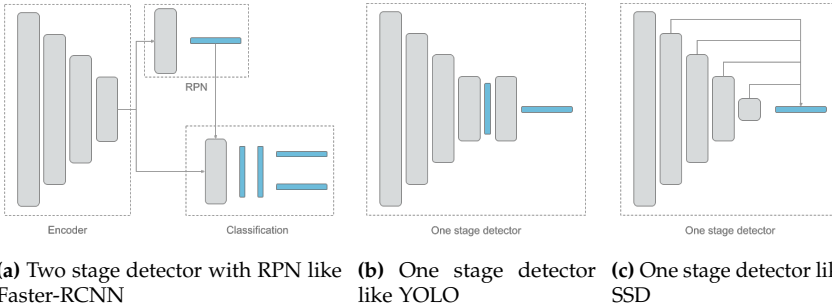


Figure 3.4. Schematic representation of different network architectures for object detection. (a) A two stage approach with RPN and Classification-Head [RHG+15]. (b, c) One stage detection as in [RDG+16] and [LAE+16].

Unfortunately, all this works only on the bounding boxes of the targeted objects (which may also include a large part of the background as described in Section 2.4.1). To obtain pixel-precise detections, the methods have to be extended. One way to combine the object scoring of individual regions with a segmentation mask is shown in *DeepMask* and its successor *SharpMask* [PCD15; PLC+16]. The network is presented with image patches and outputs the likelihood of the patch being centered on a full object, together with the appropriate mask of the primary object. Another approach ([LQD+17]) is based on the *Instance-sensitive FCN* [DHL+16] but deploys a RPN in addition. On each proposed region the instance-sensitive featuremaps are evaluated in an end-to-end manner. The authors of *Fast R-CNN* [Gir15] and *Faster R-CNN* [RHG+15] also proposed a network which is able to produce segmentation masks on the detections which they called *Mask R-CNN* [HGD+17]. This work outperformed all existing, single-model entries on the task of instance segmentation in the COCO suite of challenges [LMB+14].

3. Detection

3.2.4 Segmentation Networks

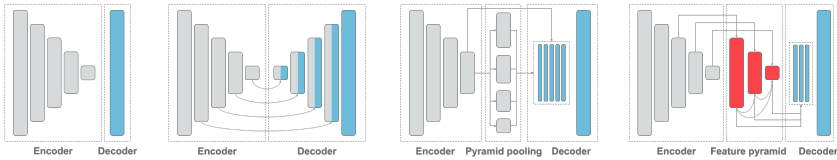
Neural networks for classification are normally designed to capture clues from all areas of the input image to infer the meaning of its content. To accomplish this integration of the whole input their architectures include pooling layers to decrease the size of the feature-maps thereby condensing and propagating the relevant information through the network. This enlarges the receptive field and help to reduce the memory requirements. This omission of unimportant features also allows the development of deeper and more complex network architectures.

While this reduction of dimensionality is desirable and essential for the task of image classification, the spatial information is discarded completely in the final fully connected layers, making pixel-wise classification impossible. So for the task of pixel-wise segmentation [LSD15] proposed a fully convolutional network (FCN) which is based on the basic architectures of the proven and well-established classification networks. They state that the fully connected layer in the final classification head can be viewed as convolutional layers, whose kernels cover their entire input regions. This makes the output of their fully convolutional network equivalent to the output of the original classification network on overlapping input patches, but with the computation highly amortized over the overlapping regions of those patches. With the convolutional layers only, the spatial information is reduced but not discarded so that it can be recovered with upsampling. Such upsampling naturally leads to coarse segmentation results. Therefore the authors introduced so called *skip-layer fusion* where they added links that combine the final prediction layer with lower layers with finer strides. This combination of fine layers and coarse layers enables the model to make local predictions that respect global structure. For the upsampling operation the authors used deconvolution layers initialized to bilinear upsampling. This means the network can rely on the classical bilinear upsampling technique but adjust the upsampling-operation parameters while training.

With the basic classification network, followed by an upsampling part (see Figure 3.5a) the FCN is a classical example of an encoder-decoder-architecture (this combination is often also called auto-encoder). The encoder part of the network learns to transform the image into a latent

3.2. Detection with Neural Networks

variable which has lost spatial dimensionality, but encodes all the contained information in its many featuremaps. The decoder part learns to infer from the describing features to reconstruct the underlying structure of the image by upsampling the spatial resolution and reducing the feature maps. SegNet [BKC17] is another good example of such encoder-decoder-network. The key contribution here was the use of skip connection for pooling layers, which transferred the indices from the pooling-operations in the encoder to the associated unpooling operations in the decoder part. The most generic encoder-decoder-architecture for segmentation networks was presented in [RFB15] with the U-Net. Here the encoder-decoder structure is build symmetrically and the corresponding steps in the encoder and decoder are connected with skip connections before/after the down/up-sampling is applied (see Figure 3.5b). This generic architecture can be build with any classification network, by mirroring its structure in the decoder part and replace the pooling operations with upsampling operations.



(a) FCN with simple upsampling. (b) U-Nets generic encoder-decoder-architecture. (c) Pyramid Scene Parsing Network (PSPN) with four pyramid levels. (d) Feature pyramid networks (FPN)

Figure 3.5. Schematic representation of different architectures of segmentation networks with encoder and decoder parts (depiction adopted from [Yak19]). (a) Fully convolutional network with simple upsampling [LSD15]. (b - d) U-Net, PSPN and FPN with different strategies for processing the latent representation [LSD15; ZSQ+17; LDG+17].

Another way to optimize the receptive field for the segmentation task are dilated convolutions (see Appendix A). With their spreading of the kernel they can be used to increase the receptive field and incorporate more contextual information without decreasing the resolution. In [YK16] these dilated convolutions are used to build a context module which can

3. Detection

be plugged at any position into existing segmentation architectures. A combination of 7 layers of 3x3 dilated convolutions with different dilation factors gives a module which has the same input as output vectors, but aggregates contextual information. With this module plugged into different architectures, the authors could increase their performance consistently. The dilated (or atrous) convolutions are revisited in the DeepLab v3 [CPS+17] paper, where a *Atrous Spatial Pyramid Pooling* consisting of four parallel atrous convolutions with different atrous rates is applied to the feature maps of the (also adapted) classification network. Such *pyramid pooling modules* were introduced in the Pyramid Scene Parsing Network [ZSQ+17]. The authors of this network found, that the classical FCNs failed to incorporate context information, which lead to missclassification of individual objects. To incorporate the global context the *Pyramid Scene Parsing module* was introduced. It consist of four pyramid levels with different sizes which are fed with a downsampled (via max-pooling) version of the final feature maps each. Through the different poolings and a subsequent convolution to a feature dimension of one, the pyramid layers contained condensed information about the features inside their respective scope (*global prior*). For the final dense prediction the pyramid layers are upsampled and concatenated with the original final featuremaps from the base network (Figure 3.5c). A very similar approach is taken by *feature pyramid networks* (FPN) [LDG+17], which take the features of the encoder in different layers and evaluate them separately in the decoder in order to subsequently combine the predictions (Figure 3.5d).

Earlier versions of the DeepLab [CPK+14; CPK+18] network combined the classical segmentation technique of a (fully connected) conditional random field (CRF) with a neural network to refine the course results of the encoder-network in a post-processing step. Other researchers even integrated the CRF into the neural network [ZJR+15] to enable end-to-end learning of the complete system.

Instance Segmentation Based on the semantic segmentation the next level of scene understanding is to delineate the individual instances of the detected classes. Detecting individual objects is normally in the scope of detection networks (see Section 3.2.3) but there are techniques to extend the

3.2. Detection with Neural Networks

semantic segmentation task to be instance-aware. In [ZSF+15] a network is used to predict the object affiliation of each pixel together with the depth ordering of the objects. The architecture is comparable to the one in the FCN-Paper but the result in the final featuremap now define for each pixel the affiliation to the objects instead of its class. The problem was reduced to a maximum of 5 objects plus the background with a strict depth ordering on the vertical axis. To enlarge the output, multiple image-patches are merged with a Markov Random Field.

Another way to segment instances is presented in [DHL+16]. The final score featuremap is here replaced by a small set of instance-sensitive featuremaps, each of which is responsible for a relative body-part of the object. One example from the paper uses 9 channels in the final featuremap representing a 3x3 grid of relative body-parts of the objects. This is motivated by the fact that the critical part of object discrimination in the output of a FCN-Network are the parts where different objects are conjunct. With the particular featuremaps for the touching parts the network is able to learn the discrimination of the objects in these parts of the image. To output the individual instances the objects only need to be assembled from the different body-part-featuremaps.

A similar idea is proposed in [UCF+16] where a FCN-network with three output-heads is defined. In addition to the classical semantic classes, the network also outputs a learned depth map as well as an *instance center direction*. The desired output of this featuremap is for each object pixel the direction towards its corresponding center discretized to a set of classes. The *instance center direction* combines information about an instance's boundary with the location of its visible center in a way which is well suited for the pixel-wise prediction of a FCN-network. The direction classes form a star-pattern at all the centers of predicted objects, and can be found with template-matching methods easily. The idea of learning pixel-wise the direction of the objects center was also presented in [LLW+17], where for each pixel the position of the top-left corner, the center and the bottom-right corner of the bounding box was learned. Together with a learned per-class instance count, the pixel could be clustered with an arbitrary clustering method in a post-processing step. As in this approach the correct clustering depends on the correct prediction of the number of instances [DNV17] used a discriminative loss function to map the pixels to points

3. Detection

in a high dimensional feature space so that pixels belonging to the same instance lie close together while different instances are separated by a wide margin. The points in the feature space can be easily clustered afterwards without a need for a known instance count. This simple architecture prove to be as powerful as much more complex ones.

The task of instance segmentation was approached for example also with many different types of recurrent neural networks. With their recurrent cells these networks incorporate a memory which enables them to work on one image in a sequential manner. In [RT16] the output of a FCN-network is fed into a recurrent layer which produces a segmentation of one of the instances in the image together with a score indicating the confidence of the network. The recurrent cells then update their state and the same image is fed into the network again. The network outputs another segmented instance until the confidence score drops below a certain level in which the model stops. In a later work [RZ17] this architecture was extended with an additional external memory which stores the already segmented instances. Based on this memory an attention mechanism selects a bounding box in which a subsequent segmentation network performs the segmentation.

Tracking

Given the definition in Section 2.1, the objective of the multi-object tracking is to find the sequential state of all the targets that best matches the observations made up to time t . So the solution can be found by performing MAP (maximum a posteriori) over the conditional distribution of the sequential states, given the observations:

$$\hat{\mathbf{S}}_{1:t} = \underset{\mathbf{S}_{1:t}}{\operatorname{argmax}} P(\mathbf{S}_{1:t} \mid \mathbf{O}_{1:t}) \quad (4.0.1)$$

There are a lot of different approaches to solve the above MAP problem, but they can coarsely be categorized into *online*- and *offline*-approaches. In the *online* processing mode only the observations up to the current frame are taken into account, enabling real-time tracking. Contrary to *offline* tracking, the observations over the complete sequence are gathered and processed subsequently. This makes it possible to solve even more complicated occlusion situations, since theoretically an optimal global solution can be found. Depending on the processing mode, the task of data association can either be solved with a probabilistic approach or using deterministic optimization.

4.1 Probabilistic Approach

The observations \mathbf{O} depend on the real state of the objects \mathbf{S} . This can be interpreted as the image captured by the camera depends on the state of the targeted objects in the scene. This dependency can be modeled as a Dynamic Bayesian Network (DBN), a graph of random variables and their conditional dependencies [Bis06], where the multi-object state \mathbf{S}_t and the observations \mathbf{O}_t are modeled by two random variables. In the

4. Tracking

tracking situation, the state of the objects changes over time and thus the observations. So the *Bayesian Network* is extended over the complete sequence of images with time index $t \in \{1, \dots, T\}$. The extension follows a first order *Markovian assumption*, so each configuration \mathbf{S}_t depends only on the configuration in the previous time-step \mathbf{S}_{t-1} and the observations are conditionally independent. The resulting graph is depicted in Figure 4.1.

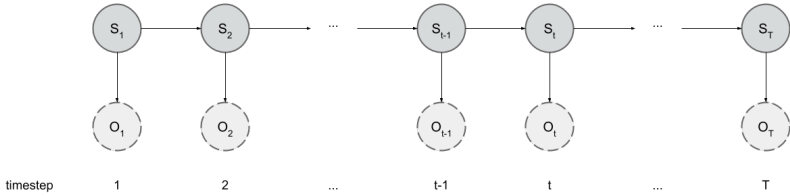


Figure 4.1. The Bayesian Network consisting of two random variables (configuration and observation) and their dependency is extended as a Markov chain (figure adopted from [Smi07]).

In each time-step the goal of the tracking is to gain the current joint state of the objects, based on all observations up to the current time $P(\mathbf{S}_t \mid \mathbf{O}_{1:t})$ (*posterior distribution*). The computation is done in a two-step iterative procedure using *bayesian filtering*. In the first step the new state of the objects is predicted based on a *motion-model* $P(\mathbf{S}_t \mid \mathbf{S}_{t-1})$ of the targets (see Section 2.4.2). Subsequently the *posterior distribution* is updated using an *observation model* $P(\mathbf{O}_t \mid \mathbf{S}_t)$ [DFB+99].

Prediction The predictive probability distribution $P(\mathbf{S}_t \mid \mathbf{O}_{1:t-1})$ is obtained by integration over all possible previous states $P(\mathbf{S}_{t-1} \mid \mathbf{O}_{1:t-1})$ weighted by the *motion model* (or *state evolution*) $P(\mathbf{S}_t \mid \mathbf{S}_{t-1})$:

$$P(\mathbf{S}_t \mid \mathbf{O}_{1:t-1}) = \int P(\mathbf{S}_t \mid \mathbf{S}_{t-1})P(\mathbf{S}_{t-1} \mid \mathbf{O}_{1:t-1})d\mathbf{S}_{t-1} \quad (4.1.1)$$

Thereby it is assumed, that the current state \mathbf{S}_t is only dependent on the previous state \mathbf{S}_{t-1} (Markov property).

4.1. Probabilistic Approach

Update The observation model $P(\mathbf{O}_t | \mathbf{S}_t)$ incorporates the current observation and gives the probability of observing the current observations \mathbf{O}_t given the current state \mathbf{S}_t . The new *posterior distribution* $P(\mathbf{S}_t | \mathbf{O}_{1:t})$ is obtained using *Bayes Theorem*:

$$P(\mathbf{S}_t | \mathbf{O}_{1:t}) = \frac{P(\mathbf{O}_t | \mathbf{S}_t)P(\mathbf{S}_t | \mathbf{O}_{1:t-1})}{P(\mathbf{O}_t | \mathbf{O}_{1:t-1})} \quad (4.1.2)$$

Substituting Equation (4.1.1) into Equation (4.1.2) yields the final *recursive bayesian filtering distribution*:

$$P(\mathbf{S}_t | \mathbf{O}_{1:t}) = c^{-1}P(\mathbf{O}_t | \mathbf{S}_t) \int P(\mathbf{S}_t | \mathbf{S}_{t-1})P(\mathbf{S}_{t-1} | \mathbf{O}_{1:t-1})d\mathbf{S}_{t-1} \quad (4.1.3)$$

Whereby the constant c represents the denominator $P(\mathbf{O}_t | \mathbf{O}_{1:t-1})$ as the observations are independent from each other.

Equation (4.1.3) combines the prediction and update phase, starting from the initial state \mathbf{S}_0 which is assumed to be given. The prediction phase combines the posterior distribution from the previous time-step $P(\mathbf{S}_{t-1} | \mathbf{O}_{1:t-1})$ with the motion model $P(\mathbf{S}_t | \mathbf{S}_{t-1})$ and therefore gives a prediction of the current state based on the previous posterior distribution and the motion model of the tracked targets. In the update phase the current observation \mathbf{O}_t is incorporated and the likelihood $P(\mathbf{O}_t | \mathbf{S}_t)$ is used to measure how well the new observation fits the predicted state. The posterior distribution is then updated accordingly.

A common implementation of the formal definition of the probabilistic tracking approach is the particle filter which was introduced for visual tracking by Isard and Blake in the *condensation* algorithm [IB98]. Particle filters use a set of N weighted samples (or particles) $\{(S_t^i, w_t^i); i \in 1 \dots N\}$ to approximate the posterior distribution from the previous time-step $P(\mathbf{S}_{t-1} | \mathbf{O}_{t-1})$. Each particle represents a hypothesis about the (multi)-object state \mathbf{S}^i , with its weight $w^i \in [0, 1]$. The classical implementation uses *importance sampling* for the approximation (see Figure 4.2). In the prediction step N new samples are drawn from a proposal distribution $Q(\mathbf{S}_t) \triangleq \sum_{i=1}^N w_{t-1}^i P(\mathbf{S}_t | \mathbf{S}_{t-1}^i)$, which gives a higher probability of selecting samples from the posterior distribution at time $t - 1$ with large weights.

4. Tracking

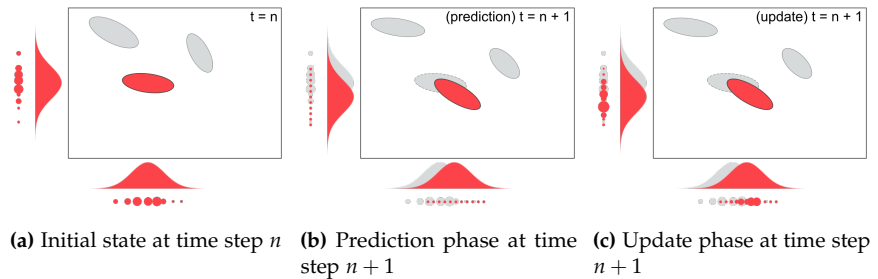


Figure 4.2. Schematic depiction of a particle filter. (a) The initial state at time step n with the posterior distribution of the red target and its approximation as weighted particles. (b) Prediction phase with the motion model applied to the new particles. (c) Update phase with the evaluation based on the observation likelihood.

Then the new hypothesis are predicted using the motion model where Q is the mixture density with the motion model $P(\mathbf{S}_t | \mathbf{S}_{t-1}^i)$ as mixture components [Smi07]. In the update step the new samples \mathbf{S}_t^i are assigned a weight w_t^i according to the evaluation of the observation likelihood at its configuration:

$$w_t^i = P(\mathbf{O}_t | \mathbf{S}_t^i) \quad (4.1.4)$$

In the *condensation* algorithm Isard and Blake showed that particles can be used to approximate a multimodal probability distribution to estimate the parameters of the contour of searched objects and thus track the objects. To counter the problems with multi-modal distributions that occur when tracking multiple objects, Vermaak et al. [VDP+03] introduced a mixture particle filter (MPF) that models the distribution with a mixture model. Thus, each component of the distribution is represented with its own particle filter, which is part of the mixture model. This approach was then extended to the boosted particle filters by Okuma et al. [OTD+04] with the addition of an Adaboost detector. Cai [CFL06] proposed the use of a separate filter for each target and later works adopted this schema and combine the individual trackers for the different targets with classifiers for each target [SCZ+08] or use the detection confidences to improve the tracking [BRL+09].

4.1. Probabilistic Approach

Another efficient variant of the particle filter is based on an approximation by Markov Chain Monte Carlo (MCMC) [KBD04]. In this case the particles have uniform associated weights $w_t^i = \frac{1}{N}$. In order to approximate a distribution from which examples can be drawn, the weighting is not used this time. Instead, a chain of states is created (Markov chain). Each state contains a valid joint configuration at time t S_t^i with the property that the frequency of occurrence of this state corresponds to the probability of the targeted posterior distribution (Equation (4.1.3)). This Markov chain can be generated e.g. with the Metropolis-Hasting algorithm [Has70], which approximates a distribution as a random walk algorithm by exploring the state space in small steps. Each new configuration in the chain S^* is created from the last configuration in the chain S^{n-1} by applying the *motion model* (or *state evolution*) $P(\mathbf{S}_t | \mathbf{S}_{t-1})$. The first element of the chain is thereby copied from a seed sample S^0 from the last time step. An acceptance ratio is then used to determine how likely the new configuration would be observed. For this purpose, the observation model is evaluated and the acceptance ratio is defined as $a = \min(1, \frac{P(\mathbf{O}_t | \mathbf{S}_t^*)}{P(\mathbf{O}_t | \mathbf{S}_t)})$. If the acceptance ratio a is greater than 1, the new sample is added. Otherwise, the new sample is added with the probability a . If the new sample is rejected, a copy of the previous sample S^{n-1} is added instead. The key to the efficiency of this approach is the use of *single component Metropolis-Hastings*, in which the state of only one object in the joint configuration is changed at a time. With caching of previous likelihood evaluations and canceling terms in the Metropolis-Hastings algorithm, the necessary computational effort can be reduced massively.

To accommodate a variable number of targets the joint state-space need to be variable in its dimension and therefore the sampler needs to construct a Markov chain that jumps dimensions. So the Metropolis-Hastings algorithm need to be adopted accordingly by defining a set of moves that make the change in dimension [Gre95]. Thereby each defined move needs to be reversible so that that the random walk can revert back to the previous state in a later move (*reversible jumps*). If the observations are based on the output of a detector which naturally gives the number of targets, the appearance of a new target increases the dimensionality (*birth*) and

4. Tracking

the leave of a target decreases it (*death*) accordingly. The sampling of the traditional MCMC algorithm to find a configuration for one of the targets (*update*), does not change the dimensionality but is reversible of course [KBD05]. In each iteration of the Metropolis-Hasting algorithm a move is selected (via a uniform distribution) first and then the target to be changed is selected according to the *single component Metropolis-Hastings* to form a new joint configuration. Depending on the move type the acceptance ratio needs to be calculated in an adapted version.

Such (RJ)MCMC trackers were successfully applied to the tracking of people [SGO05], cars [BC08] and insects [KBD05; MBJ17].

4.2 Optimization based Approach

In the optimization based approaches the problem is solved in a global way by building all possible trajectories over the complete sequence of T images and then calculating the best fitting representation $\mathbf{S}^*_{1:T}$ given all observations $\mathbf{O}_{1:T}$. The state of the targets is inferred here from the observations which are assigned to the target at each time step t . This *data association* defines the trajectory of the i -th object as an ordered list of observations $\mathbf{T}^i = (o_1^i, \dots, o_T^i)$. From such single trajectory hypotheses an association hypothesis $\mathbf{S}_{1:T}$ is defined as a set of single trajectory hypotheses $\mathbf{S}_{1:T} = \{\mathbf{T}^k\}$. The objective of the optimization is then to maximize the posteriori probability of $\mathbf{S}_{1:T}$ given the set of available observations $\{o_{1:T}^n\}$ [ZLN08]:

$$\begin{aligned} \mathbf{S}^*_{1:T} &= \operatorname{argmax}_{\mathbf{S}_{1:T}} P(\mathbf{S}_{1:T} \mid \mathbf{O}_{1:T}) \\ &= \operatorname{argmax}_{\mathbf{S}_{1:T}} P(\mathbf{O}_{1:T} \mid \mathbf{S}_{1:T}) P(\mathbf{S}_{1:T}) \\ &= \operatorname{argmax}_{\mathbf{S}_{1:T}} \prod_n P(o_{1:T}^n \mid \mathbf{S}_{1:T}) P(\mathbf{S}_{1:T}) \end{aligned} \tag{4.2.1}$$

The space of all the possible trajectories built from the available observations is huge, but can be reduced by the fact that each observation can only belong to one trajectory and therefore the single trajectory hypotheses can

4.2. Optimization based Approach

not overlap with each other:

$$\mathbf{T}^k \cap \mathbf{T}^l = \emptyset, \forall k \neq l$$

If the motion of the targets is not coupled (resp. independent) Equation (4.2.1) can be decomposed into:

$$\begin{aligned} \mathbf{S}_{1:T}^* = \operatorname{argmax}_{\mathbf{S}_{1:T}} \prod_n P(o_{1:T}^n | \mathbf{S}_{1:T}) \prod_{\mathbf{T}^k \in \mathbf{S}_{1:T}} P(\mathbf{T}^k) \\ \text{s.t. } \mathbf{T}^k \cap \mathbf{T}^l = \emptyset, \forall k \neq l \end{aligned} \quad (4.2.2)$$

The term $P(o_{1:T}^n | \mathbf{S}_{1:T})$ acts as a likelihood function, modeling the likelihood of an observation $o_{1:T}^n$ being a true detection or a false positive. $P(\mathbf{T}^k)$ is modeled with a Markov chain, defining the *entry*-, *exit*-, and *transition*-probabilities. With proper definitions for such probabilities the problem can be transformed into a standard assignment problem by defining a transition matrix and solving for the optimal assignment with the Hungarian algorithm [HWN08]. Or the problem can be cast into a min-cost / max-flow problem and solved with an appropriate min-cost flow algorithm [ZLN08; BFT+11]. Of course, there are also alternative formulations of the problem, for example as energy minimization [MRS13].

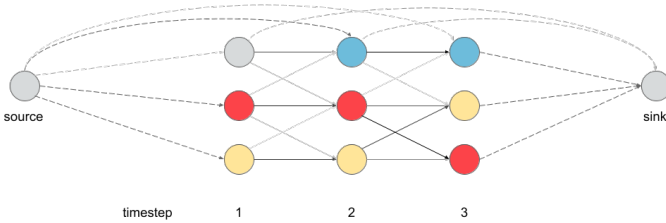


Figure 4.3. Tracking of three objects as min-cost/ max-flow problem. The association is solved by the assignment of transition probabilities as costs to the edges of the flow-graph. (figure adopted from [BFT+11])

In the MAP formulation of the tracking problem (Equation (4.2.1)) the term $P(o | \mathbf{S})$ acts as a likelihood-function of the observation o being made given a joint state \mathbf{S} . Given an appearance model of the target as a template, a simple approach is to search the area around the last detected

4. Tracking

position in a sliding window manner. This "brute force" optimization will return the local maximum of the likelihood function $\operatorname{argmax}_{\mathbf{S}} P(o | \mathbf{S})$ and therefore the state-parameters with the highest probability of representing the target. For a single rat this technique was successfully applied in [FLB11]. They use an animal model consisting of a histogram of gradients (HOG), a histogram of intensities and motion and edge density to define a weighted cost function to evaluate the likelihood of the observation. The coarse localization is then followed by a pose refinement based on the detected edges.

If multiple targets are present, the observations need to be assigned to the correct target (*data association*). Over the complete sequence of T images this will form the association hypothesis $\mathbf{S}_{1:T} = \{\mathbf{T}^k\}$ consisting of multiple trajectory hypotheses which in turn are defined as an ordered list of observations for the i -th target $\mathbf{T}^i = (o_1^i, \dots, o_T^i)$. An association hypothesis therefore states which observation belongs to which target. To find the best hypothesis or rather to maximize the posteriori probability of $\mathbf{S}_{1:T}$ given the set of available observations $\{o_{1:T}^n\}$ the problem can be transformed into a bipartite matching problem, where observations are linked between successive frames. In Equation (4.2.2) for example, the term $P(\mathbf{T}^i)$ can be modeled with a Markov chain [ZLN08] which uses *transition probabilities* $P_{link}(o_t^i | o_{t-1}^i)$ to define the linking:

$$\begin{aligned} P(\mathbf{T}^i) &= P(\{o_1^i, \dots, o_T^i\}) \\ &= P_{entr}(o_1^i) P_{link}(o_2^i | o_1^i) \dots P_{link}(o_T^i | o_{T-1}^i) P_{exit}(o_T^i) \end{aligned} \quad (4.2.3)$$

The *transition probabilities* are often based on some spatial distance metric (like euclidean) and the similarity of the linked observations based on the object model (see Section 2.4). Setting up a cost matrix with the pairwise *transition cost* the optimal solution of the assignment-problem can be found with the hungarian algorithm [Kuh55]. Examples for bipartite matching can be found in publications about pig tracking [MMP+17] and tracking of insects and larvae [SAT+17; RBO+17]. In Equation (4.2.3) the trajectories are assumed to span over the whole sequence of T images, which of course is normally not the case. This oversimplification is used here to keep the indices in the formula manageable. A useful method to deal with interrupted trajectories is hierarchical matching, where in a first

4.2. Optimization based Approach

step obvious assignments are used to create short tracklets, which are then linked in later steps with bipartite matching to form longer trajectories [HHR13].

To solve the association of indistinguishable and translucent targets in the event of heavy overlap, the optimal assignment need to be solved on a pixel-wise level. If the observations are given as identities of the targets before the overlap happens, the association of the pixels in the overlapping area can be formulated as an energy function. In [FDG+14] the tracking of *Drosophila* larvae is solved with such energy optimization problem. The motion of the larvae is modelled in a spatiotemporal neighborhood by the flow (\mathbf{f}) of masses (\mathbf{m}) between the pixel. The masses define the identities or, in the overlapping areas, the mixture of them. If the objects are isolated, the association of the pixel is observable ($\mathring{\mathbf{m}}$) and if the association is to be determined (in the overlapping areas), the mass is modelled with a latent variable ($\mathring{\mathbf{m}}$). The energy optimization problem is then defined as:

$$\underset{\mathbf{f}, \mathring{\mathbf{m}}}{\operatorname{argmin}} E(\mathbf{f}, \mathring{\mathbf{m}}, \mathring{\mathbf{m}}, \mathbf{w}) \quad (4.2.4)$$

The energy is a weighted sum of terms which favour solutions of the assignment where the association is spatially smooth and temporal consistent. The weights \mathbf{w} are learned from annotations, so that the correct interpretation of the training data receives lower energy. With some standard techniques for shaping the problem to a convex one, it can be solved efficiently by linear programming. However, such methods are usually applied only on the short sequences where overlap of animals has been detected.

Another approach to solve tracking with an optimization method is to define an error function which is used by an optimization algorithm to find the optimal mapping to the tracking targets. We presented such an approach in [BTK18] and it is described in more detail in Chapter 7. Using iterative and evolutionary methods, the optimization algorithm searches for the global minimum of the error function and thus the assignment to the targets that best fits the current observations in the image information.

Evaluation Metrics

In order to be able to assess the quality of various detection and tracking methods, comparative data is required in which the objects searched for are reliably marked (ground truth). This annotated data is often created manually and is therefore also subject to subjective falsification by the annotator. If ground truth data are available, the results of a procedure can be evaluated on the basis of defined metrics and thus a direct comparison of different procedures can be carried out.

5.1 Metrics for Detection

Detections usually have no temporal reference, so their evaluation can be done separately in each image. Given the ground-truth information of the N depicted objects in the t -th frame $(s_t^1, s_t^2, \dots, s_t^N)$ and the corresponding detections $(o_t^1, o_t^2, \dots, o_t^M)$ as defined in Section 2.1, the results of the detection can be classified in four categories. Any detection which matches with an object marked in the ground truth data is counted as *true positive* (TP). If an object is included in the ground truth data but is not found by the detector, it is a *false negative* (FN). The second problematic case is a false detection where the detector finds an object where there is none. These *false positives* (FP) are especially problematic for subsequent tracking, as it can confuse the assignment of detections to existing tracks. The fourth case are the *true negatives* (TN) and thus all areas of the image in which correctly no objects are detected. Since all possible image sections without contained objects would have to be evaluated, this case is hardly considered. Therefore, the choice of these four categories for the evaluation of object detection tasks is not uncontroversial [Pow07] and yet it is used in most publications. The assignment between ground truth positions and

5. Evaluation Metrics

observations must always be unambiguous, so that if there are several matching detections, the one with the highest agreement is chosen and the other detections must be evaluated as *false positives*. Such a greedy assignment can be achieved e.g. with the Hungarian algorithm.

Intersection over Union

To decide whether a detection matches a ground truth marker, the overlapping image areas of both marks are evaluated. The standard method for this comparison is the *intersection over union* (IoU). It is based on the Jaccard index to evaluate the similarity of sets and is therefore also well suited for the comparison of objects in two-dimensional images where the pixels are considered as elements of the sets. It is normalized (0-1) and scale invariant. Since the results of object detectors are classically defined as rectangular bounding boxes, the intersection over union is often defined with their areas, but polygons or pixel-precise segmentations can also be compared (see Figure 5.1 for a graphical representation). Given a ground-truth segmentation s_i and its bounding box B_{s_i} and likewise a detection o_i with its B_{o_i} the IoU is defined by the overlapping area of the segmentations (or the two boxes), divided by their area of union:

$$IoU(s_i, o_i) = \frac{\text{area}(s_i \cap o_i)}{\text{area}(s_i \cup o_i)} \quad (5.1.1) \quad IoU(B_{s_i}, B_{o_i}) = \frac{\text{area}(B_{s_i} \cap B_{o_i})}{\text{area}(B_{s_i} \cup B_{o_i})} \quad (5.1.2)$$

The final evaluations to *true positives* TP or *false positives* FP are then defined by a threshold value of the IoU. If the IoU is above the threshold (typically 50%), the detection is considered as TP and as FP otherwise.

The most important criterion is first and foremost the detection of all objects present and thus the highest possible true positive rate. To increase this, a detector must generate as many detections as possible. Of course, this has the side effect that the false positive rate also increases, because the detector suddenly detects objects in many things. On the other hand, suppressing detections will increase the false negative rate,

5.1. Metrics for Detection

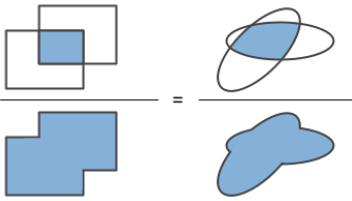
$$\text{Intersection over Union} = \frac{\text{Area of intersection}}{\text{Area of union}} = \frac{\text{Diagram 1}}{\text{Diagram 2}} = \frac{\text{Diagram 3}}{\text{Diagram 4}}$$


Figure 5.1. Example of *Intersection over Union* (IoU) for a case with bounding boxes and one with pixel-wise calculation.

because existing objects will not be marked. To control the detection rates, some detection methods can additionally calculate a confidence for each detection, which allows a subsequent filtering of the results. In this way, a balanced ratio between false positives and false negatives can be found.

Precision and Recall

To evaluate this ratio, there are two other metrics that evaluate the number of correctly found objects (TP) compared to the other two categories (FP and FN). The first is *Precision* which rates the percentage of correct positive detections over all detections. A detector that does not generate false positives therefore has a high precision.

$$\text{Precision} = \frac{|TP|}{|TP| + |FP|} = \frac{|TP|}{\text{all detections}} \quad (5.1.3)$$

The second metric is the *Recall* which evaluates how many of the actually existing objects were found. A detector that does not miss any objects has a high recall.

$$\text{Recall} = \frac{|TP|}{|TP| + |FN|} = \frac{|TP|}{\text{all ground truths}} \quad (5.1.4)$$

As described earlier, the two values are often in contradiction, so for a meaningful evaluation, both metrics must be evaluated together. The easiest way is a weighted average of both values which is also called the F1-Score:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5.1.5)$$

5. Evaluation Metrics

Another way of a combined evaluation is the *Precision-Recall-Curve*, where the data points of both metrics are plotted in a two-dimensional graph and linearly interpolated while the detections are added one by one. If the detections are sorted by their confidence, the precision is very high (probably as high as 100%) when the first detection is inserted, and then gradually decreases (with fluctuations) as the first FP detections are added. Since the Recall counts in its formula all correct detections (TP) divided by all GT objects, the Recall value will continuously increase when adding more detections. A good detector should have a high precision (few false positives) while it also has a high recall because it finds all available objects (few false negatives). So the curve of a good detector will stay as long as possible at a high precision, while the recall will gradually increase.

Average Precision

Since the precision-recall curve often oscillates, direct comparison of different detectors based on the curve alone is difficult. Therefore, in the major detection challenges [LMB+14; EEV+15], an *averaged precision* (AP) is used as a metric. The averaging is done over all data points in the precision-recall curve and thus calculates the *area under the curve* (AUC). To interpolate the curve, the precision steps are flattened by setting the precision for recall r to the maximum precision obtained for any recall $r' \geq r$. Now the rectangular areas under the curve can be easily summed. If different categories are to be detected, the AP for the classes is calculated individually and then averaged (*mean average precision* or mAP).

The adding of the individual detections in the calculation of the Precision-Recall-Curve is done by their confidence. Therefore, the curve or the calculation of the Average Precision can only be performed meaningfully if the detection method used can calculate a confidence for the individual detections.

Panoptic Quality

For the task of *panoptic segmentation* Kirillov et al. [KHG+19] also proposed a metric called *panoptic quality* (PQ). It is very similar to the F1 Score (see Equation (5.1.5)), but takes into account the special characteristic that

each pixel can only be assigned to exactly one object. It first matches the predicted segments with the ground truth segments and afterwards calculates a score based on the matches.

Since each pixel can only be assigned to one object, the predicted segments cannot overlap. Therefore it can be shown that there can be at most one predicted segment for each ground truth segment, with an intersection over union (IoU) of strictly greater than 0.5 [KHG+19]. Each ground truth segment s_i for which there is such a matching predicted segment o_i counts as a *true positive* (TP). Predicted segments that do not sufficiently overlap any ground truth segment count as *false positives* (FP) and uncovered ground truth segments count as *false negatives* (FN). The PQ is defined as:

$$PQ = \frac{\sum_{(s_i, o_i) \in TP} IoU(s_i, o_i)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|} \quad (5.1.6)$$

5.2 Metrics for Tracking

In the metrics for the detections in the previous chapter, it was evaluated whether all existing objects in the image were covered by one of the detections. However, the identities of the objects were not taken into account. The task of the tracking methods is the *data association*, i.e. to assign the detections to the correct individuals. Depending on the tracking method, the detections are either given (Detection-Based Tracking) or the detections are generated during the tracking (Detection-Free Tracking). The result of the tracking is a list of trajectories, each of which replicates the movement of an object over time. Formally, the trajectory of the i -th object over all T time steps is an ordered list of observations $\mathbf{T}^i = (o_1^i, \dots, o_T^i)$.

Given the sequential ground truth states of all the objects $\mathbf{S}_{1:t} = (\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_t)$, the sequential states $s_{i_s:i_e}^i = (s_{i_s}^i, \dots, s_{i_e}^i)$ define the true trajectory of the i -th object, where i_s and i_e are respectively the start and end of the trajectory. If the target is only temporarily occluded, the true trajectory of an object can of course also consist of several shorter trajectories. It is therefore important that a unique target ID is always present in order to be able to identify the target over the entire sequence. Likewise, the estimated trajectories may contain gaps if, for example, the detections are erroneous.

5. Evaluation Metrics

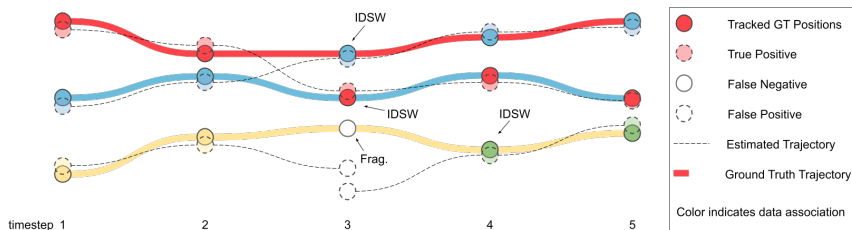


Figure 5.2. Example of a tracking result for evaluation. The assignment of the detections to the ground truth positions is based on the IoU. If the assignment changes along a ground truth trajectory, an identity switch is counted. If a ground truth trajectory is interrupted (e.g. by a false negative detection) a fragmentation is counted.

With the true and estimated trajectories defined in this way, some new performance measures can be defined that evaluate the quality of the estimated trajectories. Since the quality of the tracking depends on the underlying detections, the first step is the data association of the observations to the ground-truth positions. Based on the overlap, the *true positives*, *false positives* and *false negatives* are determined like described in the section on metrics for detection (see Section 5.1). But to account for the special temporal component of tracking, the highest match is not strictly evaluated in the assignment, but the temporal assignment is also considered. In concrete terms, this means that a ground truth trajectory that was already matched with a specific estimated trajectory in the previous time step is also matched with this trajectory in the current time step if the overlap of the corresponding detection is above the IoU threshold. Unlike the classical definition of the association in detection metrics, this matching is done even if another detection would have a higher overlap [MLR+16]. Based on this detection to ground-truth association, it can then be determined for the ground-truth trajectories how well they were reproduced by the estimated trajectories. For example by counting the *identity switches* (IDSW) as well as the fragmentation along the trajectory (see Figure 5.2). Both measures are explained in more detail in the following.

There are many different metrics for evaluating tracking methods, but according to the authors of MOTChallenge [MLR+16] (a benchmark

for multiple object tracking), the following have proven useful and have gained acceptance in the literature.

Multiple Object Tracking Accuracy

The most commonly used metric is *Multiple Object Tracking Accuracy* (MOTA), which evaluates both the quality of detections and the consistency of the mapping with respect to the target ID (in terms of identity changes). The MOTA is defined as:

$$MOTA = 1 - \frac{\sum_t |FN|_t + |FP|_t + |IDSW|_t}{\sum_t |GT|_t} \quad (5.2.1)$$

where t covers all time steps of the sequence to be evaluated and $|GT|$ describes the number of ground truth positions. Since MOTA essentially penalizes any error in detection or association, the result can also become negative, precisely when the number of errors in the tracking process is greater than the number of ground truth positions present.

Multiple Object Tracking Precision

The *Multiple Object Tracking Precision* (MOTP) shows the average agreement of all observations successfully assigned in tracking with the ground truth positions. Similar to *panoptic quality*, it evaluates the exact value of the overlap instead of a discrete decision from a threshold value:

$$MOTP = \frac{\sum_{t, (s_i, o_i) \in TP_t} IoU(s_i, o_i)}{\sum_t |TP|_t} \quad (5.2.2)$$

It evaluates the ability of the tracker to estimate the positions independently of their correct assignment to the targets [BES06], and is therefore more dependent on the quality of the detection results.

Track Quality Measures

Since it is difficult to assess the quality of a tracking result in all its variations using a single number (such as MOTA or MOTP), additional quality measures are often reported as well. For example, each individual ground truth trajectory can be evaluated in terms of the extent to which

5. Evaluation Metrics

it has been recovered from the estimated trajectories. Classical categories are *mostly tracked* (MT), *partially tracked* (PT), and *mostly lost* (ML), where a trajectory is considered mostly tracked if it is recovered more than 80% and mostly lost if it is recovered less than 20%. All values in between are considered partially tracked. It should be noted that this metric is defined to only evaluate whether the GT positions along the trajectory were successfully tracked, but not whether the ID remains the same.

Since the described classification neglects the purity of the ground-truth trajectory, the identity-switches (IDSW) are also recorded. An *identity-switch* is counted in each time step in which the associated ID of the estimated trajectory differs from the last ID associated with the ground truth trajectory (see Figure 5.2 for examples). Similarly, *fragmentations* (FM) are counted if a ground truth trajectory with a matching assignment in the last time step does not have a matching assignment in the current time step.

Part II

**Detection and Tracking of
Pigs**

Applications

In the previous part, a broad overview of different techniques used for tracking animals was given. This part introduces my contributions to this topic in the special case of pigs in livestock environments. As described in Section 1.1, tracking animals in confined environments poses a number of challenges. On the example of pigs in breeding farms most of this challenges can be witnessed. At the beginning of a fattening iteration, the young piglets are playful and bursting with energy. They investigate every alteration in the surrounding and react to disturbance with panic. Later when the animals are older and have grown accordingly, the available space is limited so that the animals stand and lie close to each other. At this time it comes also increased to the displacement actions, with which animals are moved by the actions of other animals. In the dormant phase the pigs literally stack on each other. Robust detection of the individual animals respectively tracking them over time is an ambitious task therefore. Nevertheless the evaluation of the behaviour of individual pigs is of great value for scientists and farmers, since the welfare of the complete group depends on the good behaviour of all animals.

The task of detection and tracking of the pigs is encountered in two ways. The first way, described in Chapter 7, is using a global optimizer to find the best guess of the positions based on the positional information of the previous time step and the current image of the video. Therefore this approach can only work on the continuous video and not on single images. Parts of the description in the thesis are based on my publications about this approach ([BTK15; BTK18]). The second method, described in Section 8.1, leverages the power of deep learning to detect the poses of the pigs in individual images. The resulting detections are then combined with a tracking by detection method (see Chapter 8). Details about the

6. Applications

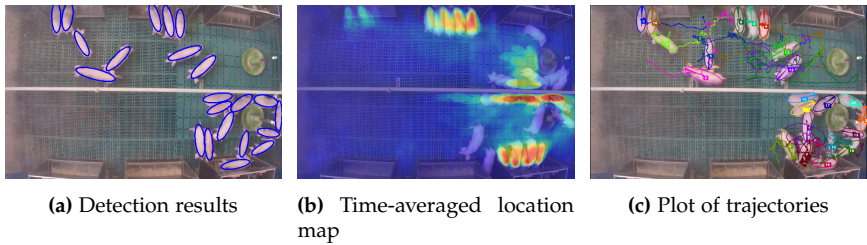


Figure 6.1. Example images, depicting possible applications of the proposed methods. (a) Detection of all the pigs in the pen. (b) Heatmap of time-averaged location in the pen. (c) Tracking of individual pigs and their movements.

detection with neural networks are already published in [BGT+20].

6.1 Problem Description

As described in the general introduction of this thesis (see Chapter 1), video surveillance can help answer various questions in the area of behavioral research in livestock animals. The videos used in this work originate from such behaviour-studies of agriculture scientists. By recording the pigs over days, the scientists try to conclude if the overall well-being, the occurrence of injuries and the general activity level depends on the environmental influences or the behaviour of individual animals. The goal then is, to identify factors which will help to increase the well-being of the animals and the overall productivity of the farm in the future. To identify and recognize individual animals, the piglets are sometimes marked with color patterns on their back, or colored ear tags. The scientists are looking for pattern in the recorded activity and compare them against the final assessment of the condition of the animals. As the recording-periods are over days, the manual evaluation of the activity of the group respective individual animals is a tedious job. To facilitate the processing of more video footage, an automatic approach would be helpful, in which the position and activity of the individual animals is detected. The goal of the methods, proposed and evaluated in this work therefor is, to provide robust detection of the pigs under all conditions (see Figure 6.1a). Based

on the detected positions of the animals, their activity and use of the barn equipment can then be determined by, for example, averaging the location over time (Figure 6.1b) or combining them to form trajectories as described in Chapter 4 (see Figure 6.1c). Accordingly, the methods described here have already found their way into studies in which the position of the animals was determined with the use of camera recordings [KKA+20; GMZ+20].

6.2 Related Work

The need for robust tracking methods is also evident in the large number of studies and scientific papers in this area. In the first part of this work, general techniques for detecting and tracking multiple objects in videos were presented. Since tracking confined animals presents its own unique challenges, many specialized techniques have been developed and studied to address these challenges.

6.2.1 Pigs

In conventional pig farming, aggressive behavior occurs repeatedly, depending on the size of the group [ANB+04]. In addition to the behavior of individual pigs, environmental factors such as litter and the right temperature also have an influence on the welfare of the animals. Methods for tracking animals in video sequences were already presented in the late 1990s. According to the technical conditions at that time, the procedures were rather rudimentary. McFarlane and Schofield [MS95] used difference images to recognize the individual animals and a narrow search window around the last position to track the animals between the individual frames. Tillett et al. [TOM97] demonstrated a similar tracking procedure, modeling only a single pig with a point distribution model. A more complex object model that takes into account spatial extent as well as other features such as color was used by Ahrendt et al. [AGK11] to track three pigs over a period of 8 minutes. Based on the point cloud of a depth sensor, Mittek et al. [MPP+16; MPC+18] used three-dimensional ellipsoid models to approximate the orientation and position of the animals. The tracking then

6. Applications

follows a classical EM clustering to fit the ellipsoid models to the point cloud of the next time step. To deal with the severe conditions of various lighting and occlusions in the barn, Zhang et al. [ZGY+18] combined several techniques to track pigs reliably. The detection of an object detection network was combined with an animal-individual track box based on a discriminative correlation filter. The online learned descriptor allowed individual identification of the animals. To avoid tracking errors, a data association based on the Hungarian algorithm was additionally applied.

One of the biggest problems in detecting and tracking pigs in video images is the correct visual separation of individual animals. Just by accurately determining the position of the animals, a lot of useful information can be obtained. For example, in their work, Kashiha et al. [KBH+13; KBO+13] show the detection of animals using binary segmentation followed by ellipse fitting. This allows, for example, the drinking behavior (stay at the drinking troughs) to be recorded. If the animals are individually marked with color patterns for certain experiments, these can also be detected. To distinguish pigs from the background, Nilsson et al. [NAÅ+14] used different color spaces to learn a descriptor based on training images which can predict whether a pixel belongs to a pig or to the background. If the animals are close to each other, they are often detected as a common object. To detect a meaningful separation in such cases, Ju et al. [JCS+18] use bounding boxes, which are suggested by a detection network. If these do not match the segmentation based on a depth sensor, the segmentations are separated with artificial holes based on skeletal tracings. Also using a depth sensor and an infrared image, Sa et al. [SCL+19] have demonstrated robust segmentation entirely without machine learning, which also works with a wide variety of lighting situations during day and night. Object detection networks are used by Nasirahmadi et al. [NSE+19] to classify the current posture (sitting, lying, standing) of individual pigs. Based on the represented postures, conclusions can be drawn about the activity level of the whole group. Psota et al. [PMP+19] also use neural networks to recognize the position of pigs in the pen. Instead of classical bounding boxes, the authors rely on single keypoints (ears, neck, tail) to recognize the animals.

In addition to the location of the individual animals, the patterns in which the animals are distributed in the space can also provide information

about their well-being. Using a global threshold over the brightness of the pixels, Shao and Xin [SX08] measured the compactness of the distribution of the detected animals as well as other parameters in correlation with environmental factors such as temperature in the barn. They were also able to infer the general activity in the barn via changes in the segmented pixels. Kashiha et al. [KBO+14] measured activity somewhat more precisely. In addition to binary segmentation, they introduced detection of individual animals to determine the number of active animals in the group. Nasirahmadi et al. [NRH+15b; NHE+16b] also used ellipses to detect the individual animals in the image. This allowed them to use Delaunay triangulation to classify the different patterns in the animals' lying behavior on the one hand, and the occurrence of certain contacts between animals, such as mounting, on the other.

Instead of using the feature of whereabouts or activity, it is of course also possible to classify directly on the behavior searched for. Viazzi et al. [VIO+14] use the motion history image to detect aggressive behavior with a Linear Discriminant Analysis (LDA). Using a Support Vector Machine, Lee et al. [LJP+16] trained a detector that not only detects aggressive behavior based on single-animal detection, but also classifies it in a downstream step.

6.2.2 Other Livestock or Laboratory Animals

In this thesis, pigs are considered in particular, but there are also many papers on other livestock. Sergeant et al. [SBF98] showed an early introduction of computer vision techniques to track poultry in the barn. Using heuristically created thresholds, they were able to separate the animals from the background and any visible shadows. Curve profiles were then used to separate closely spaced birds into individual segmentations and match them with greedy matching between successive time steps. Using a particle filter and an ellipse model per animal, Fujii et al. [FYT+09] also successfully tracked poultry. To reinitialize lost trackers, an additional tracker was introduced that scans the entire search space for uncovered foreground pixels. Also using a particle filter, Morais et al. [MCP+05] tracked and counted fish. The detection was based on a learned mixture of Gaussians to compute a multi-blob likelihood function. Bees are also a

6. Applications

good example of the difficulties with many similar targets that are densely crowded. Poiesi and Cavallaro [PC14] have developed a detector that can distinguish densely packed targets based on brightness gradients and an ellipse model. The joint-state of all targets is iteratively refined with NMS and MCMC and then the targets are tracked with a greedy graph-based approach.

Cows are also large animals that require a lot of attention and are considered accordingly in the studies. Cangar et al. [CLG+08] use an active shape algorithm on edge detection to fit a 2D points model to pregnant cows in video images. This allowed them to evaluate their movement patterns just before calving. Kim et al. [KCL17] use a thermal imaging camera to visually separate individual cows with topographic surface segmentation and track them with a velocity model. Using a landmark detector network and a downstream cow detector, Guzhva et al. [GAN+18] have used a simple probabilistic tracker to track individual cows over time.

In addition to livestock, laboratory animals are another area of application for tracking methods in video sequences. For insects such as ants, Khan et al. [KBD05] developed a (RJ)MCMC particle filter that models the interactions between animals with a Markov Random Field (MRF). Using a structured-learning approach, Fiatschi et al. [FDG+14] tracked the identities of semitransparent larvae in short video sequences. Their energy function was selected with learned weights using annotated sequences such that the association of individual pixels to insects in space remained smooth and consistent over time. To enable tracking over longer sequences, Risse et al. [RBO+17] demonstrated an entire framework for tracking drosophila larvae. This uses different methods for detection and tracking. Since the experimental setup requires a defined background, detection can use simple blob and contour detectors. Tracking is then solved using either the Hungarian algorithm or a greedy assignment. For hatched *Drosophila* flies, Sirigrivatanawong et al. [SAT+17] have presented a very similar approach. After segmentation based on background subtraction, the individual flies are detected in the image and their orientation is determined. Then, the motion vectors are determined using a Kalman filter and the targets are assigned over the different images using the Hungarian algorithm. Michels et al. [MBJ17] have also successfully applied

the (R)MCMC approach to *Drosophila* larvae. However, other than Khan et al, they introduced a much more complex object model that can also model the deformations of the insects.

In the observation of mice and rats, there is also some work aimed at reliable tracking of the animals. Branson and Belongie [BB05] use a multi-stage particle filter to track the contours of mice. With this, they combined a blob tracker with edge detection results to fit spline models to the detected animals. Pistori et al. [POM+10] combined a blob detector with a K-Means clustering method to automatically adjust their object model to the current scene. If the animals are clearly separated, the blob detector applies; otherwise, the segmented pixels are clustered. Tracking was implemented with one particle filter per animal. Giancardo et al. [GSH+13] tracked several mice in images from a thermal imaging camera. The mice clearly stand out from the background in temperature, so that the detection of individual mice only had to be adjusted for touching mice using a watershed algorithm. The tracking was then solved using the Hungarian algorithm, where mismatches were detected and corrected using a shape model and a temperature model of the individual animals. Tracking of a single mouse was performed by Farah et al. [FLB11] in a side view of the cage using a sliding search window in combination with a boundary refinement procedure. For this, Geuther et al. [GDF+19] trained a neural segmentation network that can reliably highlight the mouse from the background.

6.3 Data Sets

All videos in the data sets used here were not recorded specifically for this thesis, but were generated independently for two separate projects. Table 6.1 list some statistics on them.

The first dataset contains 1410 images showing the same pen (1.61 x 2.8 m) with 12 piglets on three different days. No special preparations were made to enhance the lighting or to improve the distinguishability of the animals from the background, but to reduce the amount of storage needed, only four images per second were captured in a highly compressed format. In addition as the original intention was to analyze the images by humans,

6. Applications

Table 6.1. Information about the two data sets. Since the images are from different studies, they differ in some respects.

Data Set	#1	#2
Resolution [px]	720 × 540	1280 × 800
Number of images	1410	1000
Includes night vision	✗	✓
Temporal resolution [fps]	4	10

the individual piglets were all marked with painted patterns on their backs (four different colors, three different symbols) to allow individual identification. Immediately after application, the color was clearly visible, but later it faded noticeably. In Figure 6.2, some example images from the data set are shown.

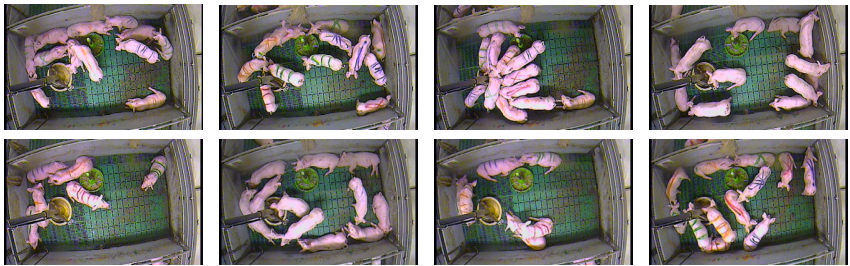


Figure 6.2. Some sample images from the first data set [BTK18]. Some piglets were marked with color patterns for individual identification for the original study.

The second data set consists of images from five different cameras each covering two 5.69 m² pens with a maximum of 13 animals each. The recordings of this data set covered a period of four months. From all the available videos 1000 frames were randomly selected. This data set contains normal color images from the daytime periods but also night vision images with active infrared illumination from the night periods. In addition, the cameras occasionally switched to night vision mode during the day due to dirty sensors. Figure 6.3 depicts sample images from the second data set, showcasing some of the challenges of working in pigsties.

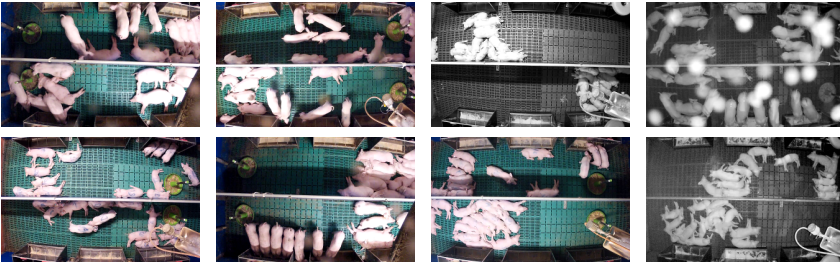


Figure 6.3. Some sample images from the second data set [BGT+20]. The example shows the dense grouping of animals and the distortions during active night vision caused by dirt on the lens.

All videos have in common, that the cameras are mounted under the ceiling to cover the whole pen. This perspective prevents most of the mutual occlusion between the animals.

6.3.1 Synthetic Data

The second method presented in this thesis (see Section 8.1) uses neural networks for the detection of the pigs. Since the accuracy and robustness of neural networks significantly depends on the amount of training data [SSS+17], synthetic images of pigs were generated in addition to the recorded videos. A theoretically infinite number of the synthetic images is available and when training the neural networks they can be mixed in a certain ratio among the training data. The goal thereby is to massively increase the variance in animal positioning and allow the network to generalize. Studies on the use of synthetic training data have shown that its use can be quite reasonable. A photorealistic replica of reality is not at all important and does not lead to better results than simpler images [MIF+18]. In fact, even with synthetic data, it is important to include the inaccuracies and imperfections of real photography (distortions, image noise) [LBZ+17; MIF+18].

Using a mock-up pigsty and a 3D model of a pig in different postures, scenes were recreated with the free modeling software Blender, which

6. Applications



Figure 6.4. Examples of the synthetically generated training images. The 3D models of the pigs are randomly placed and the environmental conditions are randomly set.

resemble the known structures and layout of the pens in the two data sets. For this purpose, classical barn floor elements such as grid floors or concrete slabs are covered with realistic textures and a random number of 3D models of the animals are placed. When rendering the synthetic images, all parameters can be adjusted as desired. Thus, the color shades of the textures, or even parts of the textures can be adjusted or exchanged. Likewise, the camera can be positioned and pointed randomly over the pen. Light sources and shadow-casting dummy objects can be used to vary the lighting of the scene. Examples of the images generated in this way can be seen in Figure 6.4. Since the poses of the pigs are known in the computer generated images, the ground truth data can be calculated automatically. Even if the synthetic images do not achieve a realistic representation, any number of images with different environmental conditions (noise, exposure, occlusion) can be generated without human time expenditure.

6.3.2 Data Preparation

When processing large amounts of data with learning-based detection techniques, data preparation is of great importance. Theoretically, convolutional neural networks can work with matrices of arbitrary size, but in order to ensure the numerical stability of the optimization and to take advantage of special hardware acceleration (e.g. GPUs), there are some limitations. GPUs now have special processing cores for computing matrix operations which are the main part of the mathematical operations in

neural network training. Moreover, these cores can work massively in parallel, so that several images can be processed simultaneously. When optimizing with the gradient descent methods, mini-batches are mostly used (the updates of the weights are calculated based on the averaged loss of several images) [Rud16], so that the images are processed together in batches in the training process. Therefore, the images are scaled to a uniform resolution and cropped in a preprocessing step. Furthermore, it is crucial for the quality of the optimization to normalize the input data (i.e. scale it so that the mean value is close to 0 and the variance is within a fixed range). This ensures more stable gradient updates and thus leads to a better convergence of the optimization [LBO+12].

The quality of the training data is also crucial for the subsequent outcome of the procedures. This includes, on the one hand, consistent annotation (see Section 6.5), but also the removal of confounding factors that could falsify the training but also the later evaluation. For example, parts of an adjacent compartment can be seen in the images of data set #1 (see Figure 6.2). Since only the animals of the compartment depicted in the center of the camera's field of view were to be examined, the non-relevant parts were masked.

In order to be able to safely evaluate the results, a part of the annotated data is reserved as a validation dataset during the training. After each epoch (i.e., the processing of all training images), the network is evaluated on these validation images to monitor the training progress. This helps to detect classic problems such as *overfitting*, where the network adapts its weights too much to the training data and thus loses its ability to generalize. Just like the validation dataset, another part of the data is retained as a test- or evaluation-dataset and not fed to the network during training. The evaluation of the different networks on only this test set prevents a falsification of the results, because in the creation process and training process of the method the validation set has already been strongly incorporated (e.g. in the choice of the hyperparameters). The split of the presented data sets into training, validation and test images is shown in Table 6.2.

As described in the previous section, the generalization of neural networks can be increased by the variance of the training data provided. Besides adding synthetic data, there is another method to increase the

6. Applications

Table 6.2. Split of the available data into training, validation and test data. In addition, the number of day and night vision images are listed.

Data Set	#1	#2 (Total)	#2 (D)	#2 (N)
Train	950	606	361	245
Validation	261	168	96	72
Test	199	226	108	118
Total	1410	1000	565	435

amount of training data. With *data augmentation*, the available real images (and correspondingly the annotation) are modified and processed in the training in addition to the original images. The augmentation included different distortions, affine transformations, and color changes (e.g., grayscale to simulate active infrared illumination) and increased the amount of training images by a factor of 10. The augmentation was performed with the *imgaug* library [JWC+20].

6.4 Animal Model

In order to detect and track the pigs, the next step is to define a reasonable object model (see Section 2.4). This model can combine different characteristics such as shape, appearance, motion patterns, etc.

Unfortunately the appearance of individual pigs is barely distinguishable as seen in the example images from the used data sets (Figures 6.2 and 6.3). Even under optimal illumination, pigs in exactly the same age and from comparable stature can look exactly the same from above. In combination with overexposed image-parts the appearance is not a reliable source of information. While there are also some sequences where the pigs are marked with color patterns, these patterns are constantly changing as the color fades or blurs and are therefore also not useful for identification.

The identification of individual pigs is therefore not possible via the color. However, since the animals stand out visually quite clearly from the background, the color still remains for the detection of the animals in general. Based on the separation of animals and the background, the

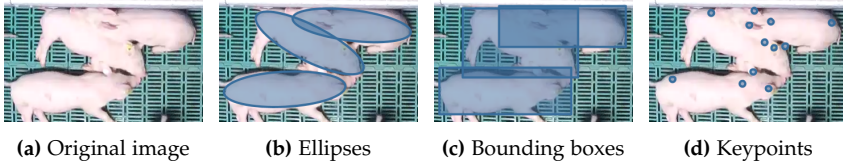
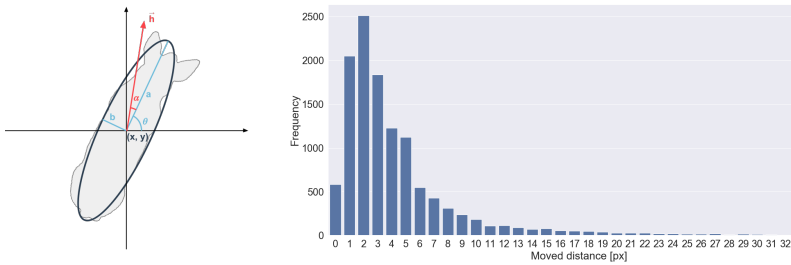


Figure 6.5. Visualization of different types of detection approaches on an example image part (a). Ellipses (b) give a much better approximation than the classic bounding boxes (large overlap) (c) or keypoints (d) where the affiliation to the individual animals has to be resolved afterwards. [BGT+20]

animals can then be described e.g. with a contour model. Many researchers who study the behaviour of pigs, have used ellipses as a simple shape-model for the pigs [MS95; KBH+13; NRH+15b]. In images like used in this work, where the pigs are captured with cameras from above, the ellipse approximates the body of the pig sufficiently, while exhibiting a simple parametrization with just five parameters (the position of the centroid (x, y) , the two main-axis lengths (a, b) and the rotation of the ellipse θ , see Figure 6.6a). Of course do the ellipses fail to cover all the pixels of a pig, if the pig’s posture is curved or individual body parts stick out (Figure 6.5b) but on the other hand is the exact identification of the contours of the individual pigs often not possible, as the pigs stand and lay closely together if the space in the pen is rare. The rotation of the ellipses also avoids that too many background pixels are assigned to the object model, which would be the case e.g. with classic bounding boxes (Figure 6.5c). And even if the ellipses can’t completely represent the outer dimensions of the animals, they still offer the possibility to roughly capture the size of the animals. Other appearance models such as keypoints would require a subsequent association and grouping of the individual points to the animals to achieve this (Figure 6.5d). Therefor the choice for ellipses is a good trade-off between effort and accuracy. They are also very easy to draw (due to the two main axes) and this is important, because in order to evaluate the presented methods, the searched positions of the animals in all images must be drawn by hand as described in the following Section 6.5. Due to the positioning of the camera on the ceiling and the perspective from above, it is difficult to tell if an animal is standing or lying down.

6. Applications



(a) Five Parameters of an ellipse and the motion vector \vec{h} .

(b) Histogram of motion vector length.

Figure 6.6. Visualization of the five ellipse parameters (position of the centroid (x, y) , the two main-axis lengths (a, b) and the rotation of the ellipse θ). The motion vector \vec{h} is measured relative to the heading of the ellipse (a). The distribution of the motion vector length in data set #1 (b) shows the low average movement of the animals between two consecutive images.

This information is not captured in the animal model, but the activity of the animals (active or resting) can be determined later by tracking based on the change in position.

6.4.1 Motion Analysis

While the color information within the ellipses contains little information, the description of position and orientation is hopefully unambiguous enough to make an association between detections in two consecutive images. With the hand-generated ground truth data, it can be shown that in data set #1, for example, the average distance the animals move between two images (resp. the length of the motion vector $|\vec{h}|$) is only 4.4 pixels (see Figure 6.6b). Thus, even with the low temporal resolution of the videos of 4 frames per second (fps), the distance to the last position is a reasonable assignment criterion.

In Section 2.4.2, motion models were introduced, which can be used to predict future motions based on past motions and thus support the

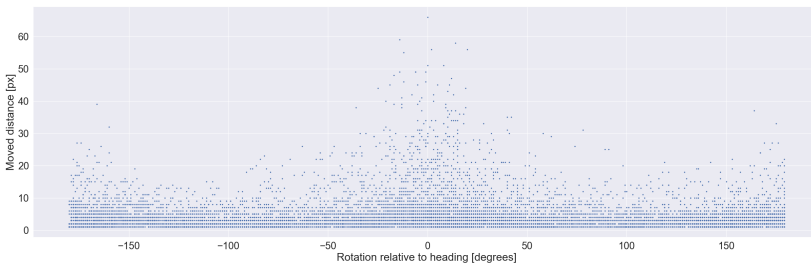


Figure 6.7. Evaluation of motion statistics on data set #1. The scatter plot of all the movements with their distance and their direction (relative to the heading of the pig) reveals a high noise rate for the position and a nearly uniform distribution of the direction of the movement.

association. However, the evaluation of the ground truth data shows that the movement of the animals does not follow a meaningful trajectory, but is rather evenly distributed over all directions. Figure 6.7 shows a scatter plot of all the movements with their distance and their direction relative to the heading of the pig. The plot reveals a high noise rate for the position (most of the movements have only distances of a few pixel). It also shows, that the movements in the direction of the heading (around 0 degrees) also as a reverse motion (around ± 180 degrees) are dominant, but for all other directions there is a high level of movements too. This means the pigs move perpendicular to their heading, which is quite unusual. A possible explanation would be that the animals are exposed to the effects of displacement due to the limited space available. Likewise, extended movements with larger distances seem not to be possible either. These two circumstances prevent the individual animals from generating movement vectors which could be used with a motion model to help (re-)identification.

Depending on the ambient temperature and available space, the activity of the animals also changes. Thus, the animals can move close together and even crawl (huddle) over each other when it is cold, or lie stretched out wide and far away from each other when they are warm [SAV+12]. Although this and the fixed space per animal make behavior difficult

6. Applications

to predict, factors such as natural displacement can be modeled as an interaction model (see Section 2.4.3).

6.5 Ground Truth Data

To evaluate the performance of the methods and to create training data for the neural networks, the positions of the pigs in all images of the data sets need to be labeled by hand. The ellipses are fully defined by their two main axis, so the labeling can be done by drawing two straight lines, outlining the two axis. From the first line, the length of the pig and its rotation can be discerned, and the midpoint of it, is the center of the ellipse. Then the second line only gives the width, but do not change the other parameters anymore. However, it is important to always start the first line at the head to capture the orientation of the animal. With this technique the drawing of an ellipse can be accomplished within under a second.

As always when data is labeled by hand, a good definition is needed how to deal with ambiguous data. In case of occlusions for example, a human annotator is able to conclude the overall position of the occluded part of the animal. This enrichment of the visible image data with the three-dimensional model is problematic, as the raw pixel-values of the occluded parts tell a different story which can be contradictory for the algorithms. On the other hand, the ellipses must be able to overlap in order to capture all the pixels belonging to the animal (see Figure 6.8a). When projecting non-transparent objects, each pixel is associated with exactly one object or the background. If animals and thus the captured ellipses overlap, it is therefore mandatory that the order of the ellipses with respect to the camera's visual axis is also captured and stored (see Figure 6.8b).

Unfortunately, a problem arises when reconstructing the positions from the saved ellipses, because the overlapping and drawing of the ellipses in the already mentioned depth sorting, results in new contours that cannot necessarily be mapped with an ellipse. This leads to the problem that the ellipses generated by the detection methods often do not correspond exactly to the ellipses generated by hand. It is important to account for this fact in the evaluation, for example by painting the ground-truth ellipses

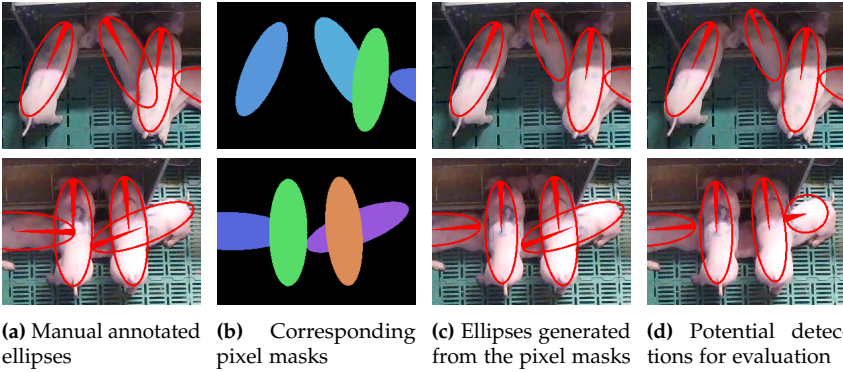


Figure 6.8. Two examples showing the problem of overlapping ground truth ellipses (a). The filled part of the ellipses shows the identified orientation of the animals. The pixel masks with correct depth sorting (b) and the ellipses obtained from them (c) allow a better comparison with potential detections (d). [BGT+20]

into a mask image and fitting subsequent ellipses into the resulting pixel masks of the individual animals (see Figure 6.8c).

Overlapping animals are one of the main problems in detection. With the perspective from the cameras on the ceiling, most occlusions can be avoided, but of course the animals sometimes crawl over one another. In the two data sets, the frequency of overlaps can be accurately quantified using the ground truth ellipses available.

Table 6.3. Evaluation of the overlap of individual animals based on the manual annotations.

Data Set	No. images	No. ellipses	Overlap 10%	Overlap 20%
#1	1410	15671	1174	552
#2	1000	22959	1321	623

Table 6.3 lists how many of the ellipses are overlapped by other ellipses at 10% and 20% of their own mass, respectively (measured in pixels). According to the ground truth ellipses an overlap of 10% occurs in the two data sets in 7.5% resp. 5.75% of the registered animals. These numbers

6. Applications



Figure 6.9. Example of automatic ground truth generation on synthetic images. The 3D objects are rendered for the input image (a) and simultaneously projected into a label image (c) (where identity is represented by different greenish color).

confirm the suitability of the camera perspective for meaningful detection of individual animals.

In the synthetic data the ground truth positions were inferred from a projective mapping of the scene and an object-based coloring of the pixel. Instead of drawing the texture of the 3D objects, the identity of the animals is set as output value in the render pipeline, creating a pixel-exact mask image. These pixel masks can then be converted into ground truth annotations by fitting an ellipse (see Figure 6.9 for an example).

6.5.1 Ground Truth for Tracking Evaluation

To evaluate the performance of the presented tracking methods and their ability to preserve the identity of the targets, the identity of the animals was also partially recorded during the generation of the ground truth data. In data set #1 there are two sequences of 500 contiguous images each and in data set #2 there is one sequence of 500 contiguous images. These three sequences are referenced in the following as *tracking sequence #1 - #3*.

Detection-Free Tracking

The introduction to this thesis addressed some of the problems associated with detecting and tracking animals in confined environments (see Section 1.1). Besides the many difficult circumstances, however, there is usually the one advantage that the number of animals in the camera image is known and does not change over time. Based on the images in Data Set #1 and the analysis of the motion patterns performed in Section 6.4, this chapter presents a method for detection-free tracking.

The motion evaluation in Section 6.4.1 shows that the distance a pig travels between two consecutive images is very small in most cases. Or, in other words, the temporal resolution of the videos is high enough to capture the pigs' movements without contradiction (see Figure 6.7). This can be exploited to find the unknown state-parameters by refining the parameters of the successfully estimated pose in the previous frame, as shown in Figure 7.1. With a starting point for the first frame, the entire sequence can then be processed automatically without further user input. This procedure is quite consistent with the approach on *detection-free tracking* (see Section 2.2), in which the state of the tracked objects is continuously updated throughout the sequence.

In the description of the animal model (see Section 6.4) it was already noted that the coloring of the individual animals is not sufficient to distinguish them individually. In this case, the object model must manage the identity of each animal by position alone. Using the ellipses as the object model, the position, size, and orientation of an animal can be described in just 5 parameters. In order to enable the tracking of the animals, a comparison function is needed, which evaluates a new observation in its quality and enables an assignment to the target object. Without the unique identification via color, the only remaining option is to use motion

7. Detection-Free Tracking

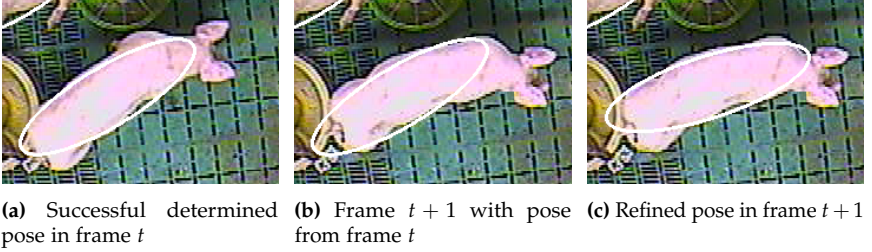


Figure 7.1. Example of the fragility of the categorical segmentation in case of strong overlaps. If the center of the animals is not visible, the segmentation cannot provide meaningful information about the hidden animal (b). The instance segmentation, on the other hand, does not have this problem (c). [BTK15]

analysis, so that the assignment is made based on motion. Given the fact, that the temporal resolution is high enough, the assignment can therefore be based on the proximity to the previous position.

Let $\mathbf{S}_t = (s_t^1, s_t^2, \dots, s_t^N)$ be the joint state of the N targets in the t -th frame of the image sequence. The object-model of each target s_t^i in the t -th frame contains the position, size and orientation described by a vector of five ellipse-parameters $\Theta_{i,t} \in \mathbb{R}^5$. When the next frame is processed, the parameter vector should be updated or refined, based on the current image I_t and the parameter-vector with the last valid position $\Theta_{i,t-1}$. Given a quality-function $f(I, \mathbb{R}^5, \mathbb{R}^5) \rightarrow \mathbb{R}$ which evaluates a parameter-vector based on the appearance of the target in the current image, the refinement can be seen as a classical optimization task

$$\Theta_{i,t}^* = \operatorname{argmax}_{\Theta \in \mathbb{R}^5} f(I_t, \Theta, \Theta_{i,t-1}) \quad (7.0.1)$$

with $\Theta_{i,t}^*$ as the vector of parameters describing the position and pose of the target in the current image best.

The formula looks a lot like the formula in traditional detection methods, where all possible positions were evaluated in an exhaustive search (see Equation (3.0.1)). Here, however, the search range can be massively restricted, since the distance to the old position should be minimized. So in this chapter an approach is presented which assigns an individual tracker to each target. This tracker maintains an internal model, capturing

7.1. Detection and Tracking with Analysis by Synthesis

the actual state of its target. In each iteration the state of the model is updated according the new video-frame to match the depicted scene best. A suitable method for such situation is the approach of *analysis by synthesis*. Given an object \mathcal{X} observed by a physical capturing system which produces the output-signal \mathcal{Y} , the task is to determine the unknown original information \mathcal{X} . Since the input-signal \mathcal{X} cannot be observed directly, the physical process of the capturing needs to be modeled and inverted. Thus this kind of problem is known as an *inverse problem*.

In real world scenarios an inversion often is impossible so a forward approach like *analysis by synthesis* is used to bypass this problem. As the name suggests, analysis by synthesis refers to an analysis process where the observed output-signal \mathcal{Y} is compared to a generated one ($\hat{\mathcal{Y}}$). For this a generator is used which uses a guessed signal $\hat{\mathcal{X}}$ to generate its output $\hat{\mathcal{Y}}$. As the input-parameters of the generated signal are known, comparing the observed signal to different generated signals will hopefully result in a good approximation $\hat{\mathcal{X}}$ of the observed input-signal \mathcal{X} .

7.1 Detection and Tracking with Analysis by Synthesis

To apply the concept of analysis by synthesis three components are needed. 1) A model of the observed object, 2) a generator to simulate the capturing process and 3) a suitable quality-function to allow comparison.

In the task of pig detection and tracking the pigs are modelled with the five ellipse-parameters $\Theta_{i,t} \in \mathbb{R}^5$. For each new input image of the sequence, the generator can access the old position of the searched animal through the stored object model and adjust it based on the new image information. It thereby simulates the imaging process to evaluate whether the newly estimated parameters of the object model match the observed new image content or rather whether the pixels in the image at the estimated position represent a pig. For this, it uses the fitness function, which allows to assess the quality of each guess. Based on the values of the fitness function, the generator then iteratively improves its guesses until convergence. Unfortunately such fitness-function tend to be multimodal. Even worse it is often non-convex and discontinuous as it is only based on the image

7. Detection-Free Tracking

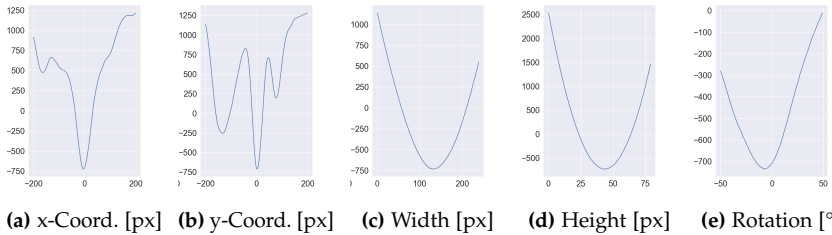


Figure 7.2. Plots of the five degrees of freedom in the fitness-function, showing the multi-modality in the first two dimensions (position of the proposed ellipse).

data which is noisy and may include compression artifacts. In addition constraints are needed to prevent the models from degeneration, which also can result in breaks or sharp edges in the function's shape. All this leads to a complex and unknown function where the only information about function values can be obtained by sampling the function at certain points in the parameter-space. To give an idea, Figure 7.2 shows a plot of the fitness-function (defined in Section 7.1.1) evaluated at different positions by sweeping each of the five parameters separately.

An optimization on such function is referred to as *black-box optimization* as no knowledge about the function can be used for proper optimization-parameter tuning. An algorithm which has shown great performance¹ in the domain of randomized black-box search techniques is Covariance Matrix Adaptation - Evolution Strategy (CMA-ES) [HO01]. It uses an internally maintained probability distribution to randomly choose points in the parameter-space where the given fitness-function is to be evaluated. The gathered fitness-function values are then used to update all the distribution-parameters trying to maximize the probability of finding the best solution. As the name suggests, CMA-ES follows an evolutionary strategy. So the points where the fitness-function is evaluated are treated as a population, from which the best entities are selected to spawn new offsprings in the next generation. To optimize the positioning of the next generation the internal probability distribution is updated accordingly. A deeper inspection on the functionality of CMA-ES is given in the

¹see 2009 Black-Box Optimization Benchmarking Competition (BBOB)

Appendix B.

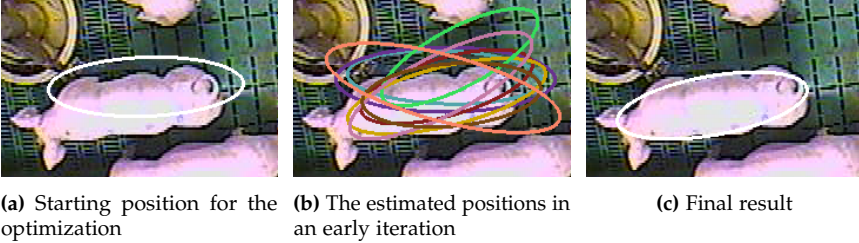


Figure 7.3. Illustration of the optimization. The optimizer starts with the last valid position, which is slightly off (a). In each iteration samples are evaluated and the internal state is updated (b). After convergence, the final result is found (c). [BTK15]

The following describes an approach which was published by me in [BTK15; BTK18]. In this implementation a separate CMA-ES-optimizer is initialized once with a manually labeled starting-position for each target in the first frame of the video sequence. In the subsequent frames each optimizer starts with the last known valid position of its target. If a pig had moved, this position would be slightly off, so the optimizer samples from the fitness-function to find a new better guess of an ellipse covering the pig (see Figure 7.3). In addition to the value of the fitness function, the (old) positions of the remaining animals are also taken into account to account the natural displacement of the animals and to prevent overlaps.

7.1.1 Fitness-Dunction

As this approach uses a fairly abstract model the generator does not need to reproduce the original image-capturing. Instead it just generates an ellipse parameter vector $\Theta \in \mathbb{R}^5$ and evaluates the fitness-function to assess its guess. The fitness function evaluates the image content of the current input image, as well as the last known positions of all currently tracked animals. The actual returned fitness value is then a weighted sum of these individual components. Formally, this results in the following definition

$$fitness_value = f(I_t, \Theta, \mathbf{S}_{t-1}, i) \quad (7.1.1)$$

7. Detection-Free Tracking

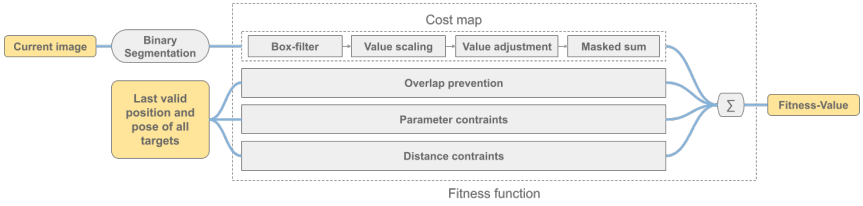


Figure 7.4. Overview flow-chart of the proposed fitness-function. The resulting fitness value is a weighted sum of four components.

with the current input image I_t , $\Theta \in \mathbb{R}^5$ as the ellipse to evaluate, $\mathbf{S}_{t-1} = (s_{t-1}^1, s_{t-1}^2, \dots, s_{t-1}^N)$ being the last known joint state of all N targets and i as an index for the target currently under consideration.

Since this is visual detection and tracking, the pixels of the input image are the main source of information. Unfortunately, the pixel values do not allow a unique identification, but they do allow to distinguish the animals from the background (see Section 6.4). So the goal of the optimizer should be to cover as many pixels of a pig as possible with a synthetic ellipse mask \mathcal{M} (created with the ellipse parameters) while keeping the percentage of background pixels low. To prevent the ellipse-mask in such a greedy approach from growing unnecessarily and occupying several animals, the parameters of the ellipse are constrained in the process and overlapping with other animals is also penalized. Using the information provided, the fitness function can be roughly divided into four components (see Figure 7.4).

The CMA-ES optimizer draws its estimates from a probability distribution, so some samples may violate the defined constraints of the ellipse model. To avoid a distortion of the distribution parameters updated by the samples, the unsuitable samples can be discarded and resampled. In the same manner samples that turn out to be bad guesses can be invalidated early by the fitness-function. To do so, the fitness-function returns a high negative number and skips subsequent steps, which helps also to speed up the evaluation.

7.1. Detection and Tracking with Analysis by Synthesis

Cost Map To identify the pixels that represent the pigs, the input image must be preprocessed and segmented into two classes. Such binary segmentation can be achieved by various methods and is used in many related works, as mentioned in Section 6.2. Binary segmentation is usually computed on the intensity values of pixels only. However, although gray levels are sufficient for the human eye to distinguish animals from the background, simple adaptive thresholding fails here. The brightness values of the pen partitions and the pigs are too similar (see Figures 7.5a to 7.5c). Similarly, motion based approaches such as background subtraction suffer problems because the animals are nearly textureless and move too little to produce distinct motion patterns (Figure 7.5d). By additionally considering the color, the segmentation can be refined to a point where it can be used as a basis for the next steps. Also masking of known barn areas that cannot be reached by animals or more complex methods based on machine learning can be used (Figure 7.5e). In this case, the assignment of the pixel to the foreground- resp. background-class is learned supervised from examples (see Section 3.2.4).

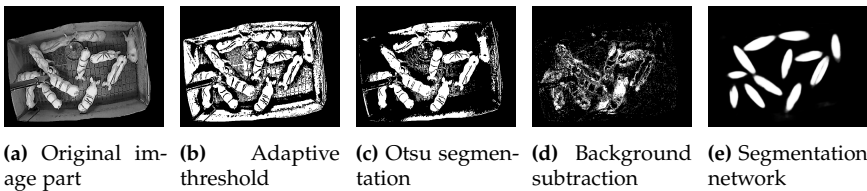


Figure 7.5. Results of different approaches for binary foreground segmentation. Classical threshold-based approaches have difficulty with barn equipment (b, c), while movement-based approaches fail because of the animals' lack of structure (d). Segmentation networks can overcome these difficulties (e).

After the binary segmentation the targets will normally form big clusters of segmented pixels except for disturbances and noise. Therefore, a segmented pixel is more likely to belong to a pig if there are multiple selected pixels in its vicinity. Single scattered segmented pixels, on the other hand, are supposed to be less relevant. A simple way to utilize this fact is to convolve the image with a quadratic box kernel of size k pixel. This will sum the pixel in the neighbourhood, so that the pixels-value

7. Detection-Free Tracking

are higher as the pixel is surrounded with more segmented pixel. Areas with a high probability of belonging to a pig thus have a pixel value of k^2 (Assuming pixels of the foreground class have a value of 1).

By summing the pixels of this cost map \mathcal{C} that are covered by the current ellipse mask \mathcal{M} , the optimizer gets some initial useful feedback on how well the estimated ellipse maps the position of a target:

$$\mathcal{L}_{cm} = \sum_{(i,j) \in \mathcal{M}} \mathcal{C}(i,j) \quad (7.1.2)$$

To further penalize background pixel coverage, the cost map is normalized and scaled to a range $[-c, c]$ with $c \in \mathbb{N}$, so pixel with no or few segmented pixel in the neighbourhood get high negative values, the clustered pixel get high positive values. Figure 7.6 shows an example of a cost-map generation as described in [BTK18] with a binary segmentation based on grayscale histogram equalization and thresholding based on color. The cost-map is normalized and scaled to a range of $[-255, 255]$ afterwards.

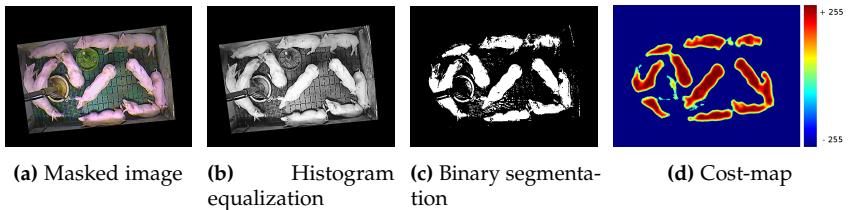


Figure 7.6. Cost-map generation as in [BTK18]. Due to tight masking of the input image (a) the binary segmentation based on histogram equalization (b, c) is a reasonable foundation for the generation of the cost-map (d).

Because the ellipses cannot exactly reproduce the body of the pigs, there are always inaccuracies, especially around the edges. Therefore, the mask can be extended by dynamic weightings. For example, the center of the ellipse is weighted normally, but the outer perimeter is only included in the sum with a reduced weight. This gives the final cost-map part of

7.1. Detection and Tracking with Analysis by Synthesis

the fitness value as:

$$\mathcal{L}_{cm} = \begin{cases} \sum_{(i,j) \in \mathcal{M}} \mathcal{C}(i,j) \cdot \mathcal{M}(i,j) & \text{if } \sum_{(i,j) \in \mathcal{M}} \mathcal{C}(i,j) \cdot \mathcal{M}(i,j) > 0 \\ \text{invalid} & \text{otherwise} \end{cases} \quad (7.1.3)$$

If the value of \mathcal{L}_{cm} is not positive the current guess is invalidated immediately.

Overlap Prevention Although the pigs also crawl over each other, in most cases the animals can be expected to displace each other (see the animal model in Section 6.4). Therefore, the fitness function prevents the overlap of the estimated pose with the known poses of the other animals \mathbf{S}_{t-1} . Even if the last known poses of the other animals are from the previous time step, the small movement of the animals justifies this comparison. The overlap is evaluated as a percentage over the area of the currently evaluated ellipse. If the overlap exceeds a threshold value (e.g. 30%), the current estimate is immediately invalidated. Otherwise, the overlap (measured in pixels) is calculated linearly and penalized by subtracting the overlap value from the total fitness value.

$$\mathcal{L}_{overlap} = \begin{cases} \text{invalid} & \text{if } \sum_{s \in \mathbf{S}_{t-1}} \mathcal{M} \cap \mathcal{M}_s > \text{threshold} \\ \sum_{s \in \mathbf{S}_{t-1}} \mathcal{M} \cap \mathcal{M}_s & \text{otherwise} \end{cases} \quad (7.1.4)$$

Discarding ellipses with too much overlap is motivated by the fact that a detection of the occluded animals at the bottom is rather unlikely, since they are hardly visible. If no ellipse with a sufficiently large fitness value is found during the optimization, the target is considered lost and detected again later by a recovery step (see Section 7.1.3).

Parameter Constraints To prevent the ellipse parameters from deviating from reasonable values, average values are calculated over the different dimensions of all targets at the start of the algorithm. Since the ellipses can appear in all possible image positions and in any rotation, especially the two main axes (or the length and width of the animals) have to be restricted in a reasonable way. On the one hand, the two values are in a certain relation and on the other hand, deviations from the average size of the

7. Detection-Free Tracking

animals are rather unlikely. For the initially registered pigs the joint state $\mathbf{S}_t = (s_t^1, s_t^2, \dots, s_t^N)$ at time-step t consist of N ellipses with parameters $s_t^i = (x_t^i, y_t^i, w_t^i, h_t^i, r_t^i)$ describing its position (x, y) , its dimensions (w, h) and its rotation (r) . To ensure temporal smoothness of the dimensions (w, h) , a moving average over the last time steps is calculated for each target (\bar{w}, \bar{h}) . Deviations from this average are penalized in the same way as deviations from the average of the initial values over all animals:

$$\mathcal{L}_{constr} = |\bar{w} - w|^2 + |\bar{h} - h|^2 + |\bar{w}_{all} - w|^2 + |\bar{h}_{all} - h|^2 \quad (7.1.5)$$

with $\bar{w}_{all} = \frac{1}{N} \sum_{i=1}^N w_0^i$ and $\bar{h}_{all} = \frac{1}{N} \sum_{i=1}^N h_0^i$.

Distance Constraint The last term of the fitness function incorporates the motion model of the animals and maintains the assignment of the animals across the time steps. The plain Euclidean distance between the previous position and the current guessed one penalizes too large steps in the movement and thus prevents a wrong assignment:

$$\mathcal{L}_{dist} = \|(x, y) - (x_{t-1}, y_{t-1})\|_2 \quad (7.1.6)$$

If the current guess was not invalidated in one of the stages, the overall fitness function formula results in a weighted sum of its terms:

$$f(I_t, \Theta, \mathbf{S}_{t-1}, i) = w_0 \cdot \mathcal{L}_{cm} - (w_1 \cdot \mathcal{L}_{overlap} + w_2 \cdot \mathcal{L}_{constr} + w_3 \cdot \mathcal{L}_{dist}) \quad (7.1.7)$$

with weights $w_0, \dots, w_3 \in \mathbb{R}$. All weights are determined heuristically and need to be adjusted to the scaling of the cost-map.

7.1.2 Update Iteration

As mentioned earlier, CMA-Es is an evolutionary algorithm, and in each iteration a population of guesses is generated and evaluated so that the next generation inherits the parameters of successful entities. By randomly scattering the entities of the new generation, CMA-ES can handle non-continuous or multimodal functions without getting stuck in local maxima.

7.1. Detection and Tracking with Analysis by Synthesis

After each iteration, the internal probability distributions are updated based on the fitness values of the current generation such that the positions converge at the maximum of the fitness function. If no sample with a value above a minimum threshold can be found, the optimization terminates unsuccessfully after a defined number of iterations. Following the definition of detection free tracking, an independent tracker (consisting of the ellipse model and the optimizer) is created for each target. These trackers are then sequentially executed one after the other on each new input image, sharing common knowledge in the form of the last known positions of all targets and updating them accordingly. The order of execution is defined by the last fitness value to prevent good detections from being disturbed by inferior ones.

Optimizing the parameters of all targets at once has been tested too but gave inferior results due to the very large dimensionality of the joint problem.

7.1.3 Recovery

Despite the good perspective from the ceiling of the barn, occlusions of individual animals may occur (see Table 6.3). Barn equipment such as automatic feeders and piping can block the camera's view. Animals can also crawl over each other, completely obscuring the animals below. As a result, an animal may not contribute any information to the image content and thus cannot be detected. To model these cases, the optimizer has to maintain an additional state per target, which stores whether the target can currently be actively tracked or not. If a tracker cannot find the target, it is marked as lost and the tracker starts in recovery mode in the next time step.

Each CMA-ES optimizer draws its guesses from a probability distribution over the five-dimensional search space on the possible values of the ellipse parameters. The search radius of the optimizers in the image is mainly determined by the first two parameters of the ellipse (center). By changing the distribution parameters, the search space can therefore be artificially expanded to encompass the entire image. Accordingly, the population size must be increased to increase the density of the sampling of the fitness function. In addition the *parameter-* and *distance constraint*

7. Detection-Free Tracking

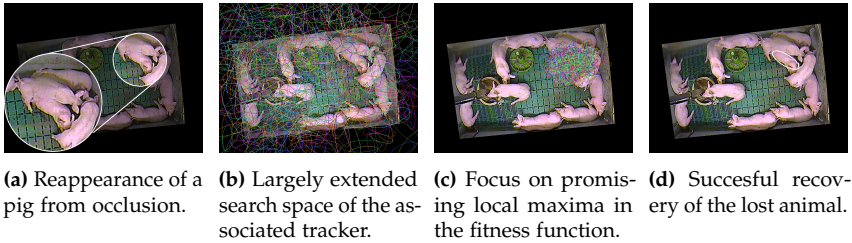


Figure 7.7. Visualization of the recovery process as described in [BTK18]. By expanding the search area (b), local maxima can be found quickly (c), indicating untracked animals.

are suspended to deal with a possibly new appearance of the reappeared target. Since all other active trackers have already been evaluated (see previous Section 7.1.2) and due to the overlap prevention, a reappeared animal will produce a strong local maximum in the fitness function.

Although resuming tracking through recovery mode works well, the simultaneous reappearance of two animals can result in an identity switch. To reliably find animals, the deviations of the new position from the previous position of the animals are suspended in the evaluation of the fitness function. Therefore, the unambiguous assignment cannot be guaranteed, especially when the affected animals are close to each other.

7.2 Evaluation

Detection-free tracking combines detection with tracking. However, since the quality of tracking strongly depends on the quality of detections, detection and tracking should be evaluated separately. Only in this way is it possible to point out the influence of the quality of the detection results on the quality of the tracking results.

The weights defined in Equation (7.1.7) to balance the different elements of the fitness function were determined heuristically. In all subsequent evaluations, they were set as follows: $w_0 = 0.001$, $w_1 = 0.2$, $w_3 = 0.025$, $w_4 = 1.5$.

7.2.1 Evaluation of Detection

To evaluate the quality of the detections, their correspondence to the positions of the animals from the manual annotation is compared. It is important to remember that the optimization process starts on the last detected position and therefore an evaluation of individual images is not possible. In this section, therefore, the detections on the tracking sequences #1-#3 (see Section 6.5.1) are evaluated (in contrast to the evaluation of the detection results in Section 8.3.1.) Furthermore, in this procedure, subsequent errors cannot be avoided for a target that has been incorrectly detected once, which can further falsify the result.

As described in Section 5.1, the metrics for detection evaluate the overlap of the ground truth objects that are present with the detections that are found in each image separately. There is no temporal correspondence or target identity preservation and the mapping is greedy, such that the optimal bipartite association is chosen. For the evaluation, the metrics *precision* and *recall* are used, as well as the *F1 score* combined from the two values. Since these metrics follow a discrete classification, the *panoptic quality* (PQ) is also listed, which additionally includes the degree of overlap.

In the method described in this chapter, the quality of the detections highly depends on the quality of the binary segmentation, because the *cost map* and thus the underlying data for the optimization is built on it (see Section 7.1.1). Although the described method of cost-map generation is quite robust against errors and holes in the segmentation, larger errors can massively influence the detection result. If barn areas are wrongly marked as animals, or animals are not recognized due to shading, the detections are distorted consequently. As already described above, a binary segmentation can be generated with a range of methods of varying complexity. So in order to assess the influence of the binary segmentation, three different evaluations are made. In the first method, referenced as *Color*, the segmentation is generated as described in [BTK18] by histogram-equalizing the gray value image and then additionally excluding pixels based on color information (e.g. to separate the greenish stable floor). In the second method *SN*, a segmentation network is trained, as described in Section 8.1.2, to separate the animals from the background. And as a base-

7. Detection-Free Tracking

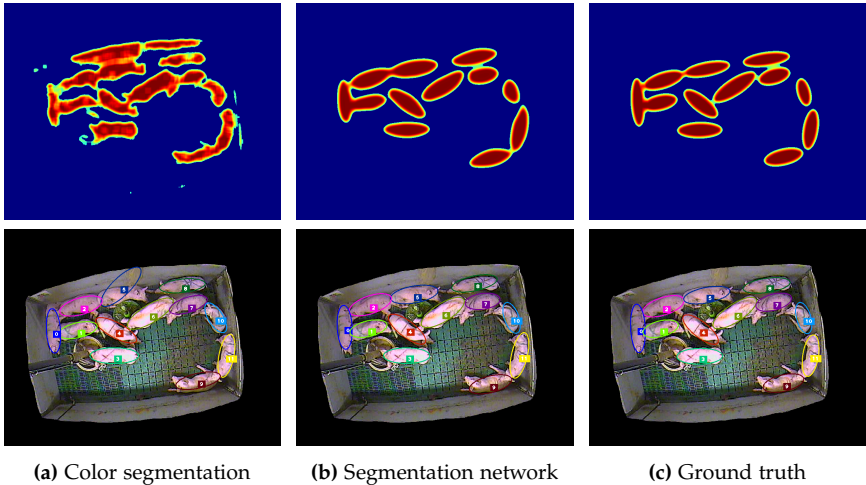


Figure 7.8. Visualization of the detection results based on the corresponding cost maps. (a) The simple binary segmentation based on the color information results in errors in the cost map and misleading information, which ultimately cause erroneous detections. (b) The much better segmentation done with the neural network does not show these problems and is close to the segmentation with ground truth data (c).

line, an additional evaluation is made where the binary segmentation is based on the manual annotated ground truth ellipses (*GT*). This shows the performance of the method under an optimal segmentation and therefore defines the upper bound. To demonstrate the influence of the cost-map on the accuracy of the detection, Figure 7.8 shows the generated cost-maps for all three mentioned methods for the generation of the underlying binary segmentation for an example image. Depending on the quality of the cost map, it can be clearly seen that errors in detection occur, e.g. when barn equipment is falsely detected and marked as parts of the animals.

Table 7.1 lists the results for the three tracking-sequences. For the calculation of the metrics the classification into successful detection was evaluated with an *IoU* threshold of 0.5, as it is common in the large detection benchmarks [LMB+14]. As already shown visually in Figure 7.8, the quality of the binary segmentation is crucial for the quality of the

detections. This also becomes clear in the evaluation of the metrics, because the results of the binary segmentation with the segmentation network are very close to the results based on the ground truth data, while the errors with the segmentation on color values are larger.

Table 7.1. Detection results of the proposed detection-free tracking method. Only the overlap of detections and ground truth positions and no target identity is evaluated. For all metrics the threshold for the *IoU*-Score was set to 0.5.

Sequence	Segment.	Precision \uparrow	Recall \uparrow	F1 \uparrow	PQ \uparrow
#1	Color	0.7526	0.7498	0.7512	0.5587
#1	SN	0.9892	0.9797	0.9844	0.8684
#1	GT	0.9806	0.9712	0.9759	0.8737
#2	Color	0.8092	0.7965	0.8027	0.6155
#2	SN	0.9860	0.9718	0.9788	0.8806
#2	GT	0.9869	0.9830	0.9850	0.9037
#3	Color	0.9162	0.9161	0.9161	0.6444
#3	SN	0.9607	0.9511	0.9559	0.7612
#3	GT	0.9923	0.9775	0.9848	0.8699

But even with optimal segmentation based on ground truth data, not all positions can be detected without error. This is mainly due to the fact that the animals overlap and may therefore occlude each other (see Section 6.5). Table 6.3 listed the proportion of overlapping ground truth ellipses for the underlying data sets. To illustrate the relationship between detection results and animal overlap, Figure 7.9 plots the evaluations along with a normalized overlap value for tracking sequence #1. The overlap (measured in pixels) was determined per image using the manual annotated ellipses and is normalized over the entire sequence. Thus, it does not represent an absolute value. It is easy to see that the detection rates decrease significantly as soon as the overlaps increase. At the end of the sequence, one animal is hidden by housing equipment, which was not evaluated as an overlap here, but still leads to false negatives.

From the data presented in Table 7.1 and Figure 7.9, it can be concluded that the binary segmentation based on the neural network provides an almost perfect pre-stage for the subsequent detection. Nevertheless, the method fails in complex situations in the barn, when the animals interact heavily and the positions of the animals are no longer recognizable

7. Detection-Free Tracking

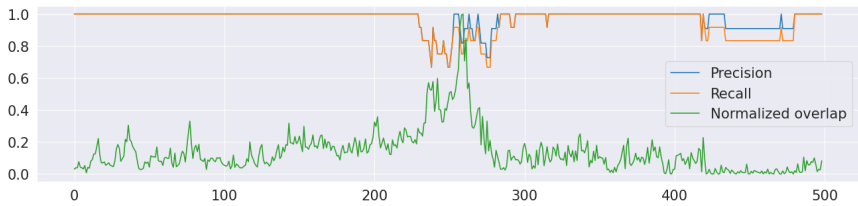


Figure 7.9. Plot of relationship between detection accuracy and animal overlap for the 500 frames of tracking sequence #1. Overlap is calculated as a normalized value across the sequence.

unambiguously.

7.2.2 Evaluation of Tracking

The tracking of the animals is evaluated according to the metrics presented in Section 5.2. In the evaluation of the detections it became already clear how big the influence of the binary segmentation on the final result is. Therefore, the evaluation of the tracking is also performed with all three segmentation methods presented in the previous section. It should be noted, however, that the number of successful (true positives) or failed (false positives) detections used to calculate the metrics is computed slightly differently in this case, as temporal consistency is now prioritized (for details, see Section 5.2). Since CMA-ES is a randomized procedure, the evaluation experiments for tracking are repeated 10 times each. The results are listed in Tables 7.2 and 7.3 with mean values over the 10 runs as well as the standard deviation. The influence of binary segmentation can also be clearly identified in the evaluation of tracking. The jump in accuracy between the classical color segmentation and the segmentation based on neural networks is large, while the difference between the results based on the learned segmentation and the tracking on ground truth data is rather small.

Table 7.2. Tracking results of the proposed detection-free tracking method according to the metrics presented in Section 5.2. The evaluation is performed separately for three different segmentation methods. All experiments are repeated 10 times each, so the reported values are the mean with the corresponding std.

Sequence	Segment.	MOTA \uparrow	IDSW \downarrow	Frag \downarrow	MOTP \uparrow
#1	Color	0.4110 \pm 0.0559	187.4 \pm 22.0	94.6 \pm 11.2	0.7427 \pm 0.0043
#1	SN	0.9486 \pm 0.0199	44.2 \pm 5.6	21.1 \pm 2.3	0.8664 \pm 0.0005
#1	GT	0.9415 \pm 0.0084	38.8 \pm 6.3	18.8 \pm 3.1	0.8947 \pm 0.0007
#2	Color	0.5823 \pm 0.0741	106.3 \pm 20.1	53.5 \pm 10.1	0.7678 \pm 0.0032
#2	SN	0.9394 \pm 0.0327	44.4 \pm 31.0	20.8 \pm 14.1	0.8768 \pm 0.0058
#2	GT	0.9590 \pm 0.0046	24.8 \pm 3.2	12.1 \pm 1.4	0.9148 \pm 0.0015
#3	Color	0.7622 \pm 0.0613	177.7 \pm 12.8	90.8 \pm 6.9	0.7043 \pm 0.0008
#3	SN	0.8997 \pm 0.0288	129.2 \pm 16.9	63.5 \pm 8.5	0.7974 \pm 0.0011
#3	GT	0.9535 \pm 0.0363	44.6 \pm 36.6	20.8 \pm 16.6	0.8818 \pm 0.0034

Table 7.3. Continuation of the evaluation from Table 7.2 with the metrics *mostly tracked* (MT), *partially tracked* (PT) and *mostly lost* (ML). For details see Section 5.2.

Sequence	Segment.	MT \uparrow	PT \updownarrow	ML \downarrow
#1	Color	0.4000 \pm 0.0726	0.5917 \pm 0.0583	0.0083 \pm 0.0250
#1	SN	0.9833 \pm 0.0333	0.0167 \pm 0.0333	0.0000 \pm 0.0000
#1	GT	0.9917 \pm 0.0250	0.0083 \pm 0.0250	0.0000 \pm 0.0000
#2	Color	0.5833 \pm 0.1054	0.3917 \pm 0.0990	0.0250 \pm 0.0382
#2	SN	0.9667 \pm 0.0408	0.0333 \pm 0.0408	0.0000 \pm 0.0000
#2	GT	1.0000 \pm 0.0000	0.0000 \pm 0.0000	0.0000 \pm 0.0000
#3	Color	0.7909 \pm 0.0891	0.2090 \pm 0.0891	0.0000 \pm 0.0000
#3	SN	0.9000 \pm 0.0182	0.1000 \pm 0.0182	0.0000 \pm 0.0000
#3	GT	0.9636 \pm 0.0273	0.0364 \pm 0.0273	0.0000 \pm 0.0000

7.2.3 Runtime

The probability that the optimizer, when searching for the current position of a target, makes a proposal that matches the real position as closely as possible, increases with the number of attempts that it can evaluate. On the other hand, each evaluation of the fitness function requires computing power and thus influences the runtime of the method.

In CMAES, the number of function evaluations is significantly influenced by the size of the population. If the size of the population is small, more iterations are needed to achieve a significant improvement of the

7. Detection-Free Tracking

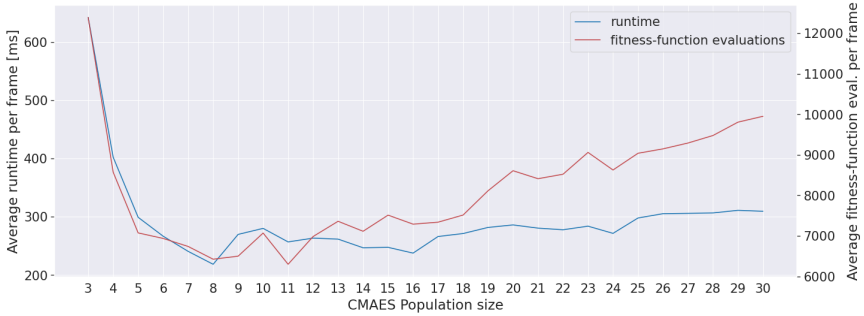


Figure 7.10. Plot of average runtime and number of fitness-function evaluations over the population size of the CMA-ES optimizers.

internal parameters. If it is large, the count of fitness-function evaluations increases and therefore also the runtime. On the other hand, more individuals may help the optimizer converge faster which will result in less evaluations. The author of CMAES proposed the optimal population-size λ for l parameters as $\lambda = 4 + \lfloor 3 \ln l \rfloor$ [Han09]. For the five-dimensional fitness-function this gives an optimal population-size of 8 which was used in this work. To confirm the choice of population size, the influence of the size on the number of executed fitness-function evaluations as well as the average runtime per frame is shown for tracking sequence #1 in Figure 7.10. With approx. 220 ms per frame the runtime is minimal at this point which makes it even suitable for online evaluation of videos with low framerates (as data set #1). In [BTK18] it is also shown that at the optimal population size on the detection rate reaches its peak and further increasing the population size does not increase the detection accuracy.

As the individuals of the population are evaluated independently in each iteration, this can be done in parallel via multi-threading. In Figure 7.10 local minima at $\lambda \in \{8, 16, 24\}$ can be seen in the runtime curve, which indicate optimal processing of this configuration with the test system used. The system used was a quad-core CPU with *Hyper-Threading* (Intel Core i7-4790), which can optimally handle multiples of 8 threads.

Detection-Based Tracking

In the previous chapter, a method for tracking pigs based on detection-free tracking was presented. In this chapter, detection-based methods are shown in order to compare the different approaches. As described in Chapter 2, the success rate of such a tracking method depends of course significantly on the quality of the detector, therefore the focus in this thesis is on detection (see the next Section 8.1). For tracking, three established methods are used (see Section 8.2).

8.1 Detection of Pigs with Panoptic Segmentation

Traditional detectors based on hand-selected features have been increasingly outperformed by detectors based on learned features. Although the latter requires the generation of a training dataset (at least in the classical supervised case) and thus involves extra work, the adaptability and generality of these solutions is helpful to address specific target domains. In Section 3.2.2 different approaches were presented, how object detection with neural networks can be realized. These include detection networks, which produce region proposals in the form of bounding boxes, and segmentation networks, which estimate the assignment of each pixel. In Section 6.4 about the *animal model* of pigs, the advantage of ellipses as contour models compared to classical bounding boxes had already been shown. Therefore, an approach is presented here in which the individual animals are detected using a pixel-wise *panoptic segmentation*, where the segmentation is subsequently used to fit the ellipse models. The proposed method can be varied modularly from a binary segmentation to an in-

8. Detection-Based Tracking

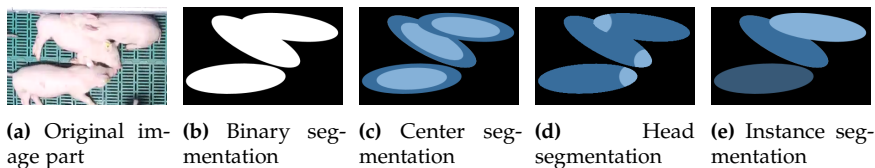


Figure 8.1. Visualization of the different steps of pixel segmentation for an example image (a). The binary segmentation (b) distinguishes only between foreground and background. The center segmentation can be used to separate the individual animals (c) or the classes are defined to identify body parts (d). Or the network is trained to directly tell the affiliation of the pixels to the individual animals (e). Images from [BGT+20]

stance segmentation including orientation detection and has already been published by me in [BGT+20].

8.1.1 Framework

Pixel-precise segmentation can be performed at different levels of complexity. The simplest type is a foreground-background separation, which only distinguishes between object and non-object (see Figure 8.1a). Such binary or foreground/background segmentation is for example used in the fitness function of the optimization based approach (see Section 7.1.1). Based on this segmentation the optimizer can rate its guess of the pigs' positions. With additional added classes, further information can be extracted, such as the labeling of individual body parts (see Figures 8.1c and 8.1d). This can either be used directly to detect the individual animals, or the output of such a categorical segmentation can be included in a combined method as additional information (for example, to detect the orientation of the animals). Another possibility is the implementation of an abstract representation, in which the individual pixels are embedded in a high-dimensional feature space and positioned based on the prediction. The learned pixel embedding serves as input for a clustering postprocessing step, which groups the pixels belonging to the individual animals (see Figure 8.1e).

All described segmentation types can be generated with the presented framework. The binary and categorical segmentation are described in

8.1. Detection of Pigs with Panoptic Segmentation

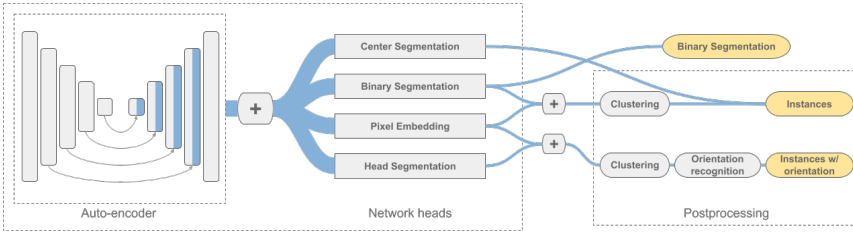


Figure 8.2. Schematic representation of the used modular framework [BGT+20]. The auto-encoder produces an intermediate representation (U-Net architecture depiction adopted from [Yak19]), which is passed through different heads. The final network output is processed afterwards to yield the desired results.

Section 8.1.2, the instance segmentation based on pixel embedding in Section 8.1.3.

Architecture

The different types of segmentation differ only in the dimensionality of the output of the network, which leads to a very high reusability of large parts of the processing pipeline. The input images are first processed by a segmentation network, resulting in an intermediate representation that has the same spatial dimension as the input image. This intermediate representation does not yet contain the final prediction of the network but consists of a set of feature maps (see Section 3.2.1). Depending on the desired output, the feature maps are then combined with a network header and a suitable loss function. The different heads consist of convolutional layers and activation functions to produce the requested output directly or with a subsequent post-processing step. The heads are not mutually exclusive, so that the outputs of several heads can also be combined in post-processing. A high-level overview of the proposed framework is depicted in Figure 8.2.

The first part of the pipeline is an arbitrary encoder-decoder network (or auto-encoder). In Figure 8.2 the depicted architecture is a U-Net [RFB15] which can be build from almost every classification network. The classification network is used as the encoder and extended with a symmetric

8. Detection-Based Tracking

decoder and skip connections. But as described in Section 3.2.4 there are also many other architectures which are based on the outputs of a classical encoder. So apart from the actual segmentation architecture, the encoder part can also be selected arbitrarily and thus benefit from the latest developments in the field of classification networks.

Encoder-Backbone

The classification networks which are the base for the segmentation architecture are called *base network* or *backbone*. The original FCN-Paper used the famous VGG [SZ14] and GoogLeNet [SLJ+15] architectures as backbones as those networks achieved high performance in the *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) [RDS+15]. The main novelty in both approaches was the use of blocks, which are build from convolutional layers with small-sized filter kernels. Stacking this blocks results in an architectures which is modular and less complex, while preserving a large receptive field. The combination of multiple convolutions with small filter-kernels instead of one convolution with a large filter also reduced the parameter count and thus the memory consumption significantly, which allowed deeper networks. While in VGG the convolutions where placed in a sequential order, the *Inception-Module* from the GoogLeNet was designed with parallel paths. These path have different receptive fields (1x1 up to 5x5 filters) and their results are concatenated in the end. In addition one parallel path also pools the features as pooling has proven to be essential for the success of convolutional neural networks. Later descendants of this architecture [SVI+16; SIV+17] optimized the inception-modules further in terms of computational complexity by factorization of the individual convolutions. The different versions of the *Inception-Module* are described in detail in the the original paper [SLJ+15].

Although the use of such blocks allowed for deeper networks, more layers don't lead necessarily to more performance. In [HZR+16] the authors showed that in fact deeper networks perform worse. This is in so far as surprising as, as the authors note, any shallower network can be transferred to its deeper counterpart by adding simple identity layers. Nevertheless, the optimization algorithms are not able to find this optimal solution. The authors therefore propose the *deep residual learning framework* in order to

8.1. Detection of Pigs with Panoptic Segmentation

exploit the potential of the deeper networks. By skip-connections over single blocks of layers, the underlying mapping function is converted into a residual mapping function. This allows the optimizers to easily train very deep architectures. Combinations of the ideas of the *Inception-Module* and the *residual learning framework* led to more advanced architectures like Inception-ResNet [SIV+17] and ResNeXt [XGD+17], which further pushed the capabilities of such networks in the task of classification. While ResNet uses skip-connections over single blocks of layers to allow the information to be forwarded unchanged, the authors of the *Dense Convolutional Network* (DenseNet) [HLV+17] found that allowing information to be interchanged by all layers in a block to be beneficial. So the L layers in a block are connected with $\frac{L(L+1)}{2}$ connections, but in contrast to ResNet the results are never combined through summation but by concatenation. As the authors state, with this the network can build up *collective knowledge* and the final classifier can make its decision based on all feature-maps in the network. They found the improved flow of information and gradients throughout the network to make them easy to train, especially with deeper network architectures as each layer has direct access to the gradients from the loss function and the original input signal.

Following this development, the experiments for semantic segmentation on the pigs were carried with an U-Net architecture with different recent classification backbones. Other architectures were tested as well, but showed comparable or worse performance (see ablation studies in Section 8.3.1).

Weight Initialization

The right initialization of the weights of a neural network is crucial for its performance. When the calculated outputs are passed from one layer to the next through the network, good initialization of the weights ensures that they do not leave the range of numerical stability. Values that are too large or close to zero would quickly leave the representable ranges or lead to exploding or vanishing gradients in the backward pass. In such a case, successful convergence is very unlikely if not impossible. The main findings for the choice of the weights in convolutional neural networks were that a normal distribution with a specific standard deviation and mean is beneficial [LBO+12; GB10]. Another very important finding about

8. Detection-Based Tracking

the weight characteristics of the different layers in deep networks was presented in [YCB+14]. The authors studied the idea of *transfer learning* [Ben12] and investigated the possibility of reusing weights trained on one of the big image datasets for a new task. The authors explain that regardless of the architecture, the learning process and the training images used, the networks always show a similar characteristic of the weights, with the first layers imitating coarse texture filters and later layers learning problem-related features. Therefore it can be useful to train the network on a large general dataset like ImageNet [DDS+09] and later fine-tune it with a lower learning rate on the actual dataset. In the special case of the U-Net which is built from the encoder and decoder part where the encoder is a common classification network, pretrained weights for them are often available. This avoids training the network on the often very large data sets, which can sometimes take a few days. In this case, however, it must be taken into account that only the encoder part has predefined weights and the symmetric decoder is randomly initialized. Training this construct can wreck the pretrained weights in the encoder because of the large gradient updates triggered by the nontrained weights in the decoder. Therefore it can be conducive for the probability of fast convergence to freeze the encoder for some time, while the decoder adjusts.

8.1.2 Categorical Segmentation

In categorical segmentation, a class is assigned to each pixel of the output image. This categorization can be chosen arbitrarily fine (number of classes to be assigned) and thus be used for answering different questions. To make statements about the class association, segmentation networks produce different representations of their prediction depending on the activation function and the number of feature maps in the last layer. These raw predictions (*logits*) are then fed into a normalization function to produce a vector of (normalized) probabilities with a value for each possible class.

Binary Segmentation

Binary segmentation distinguishes between only two classes. Similar to the classic foreground-background segmentation, the goal is to decide for each pixel whether it belongs to a pig or to the background. Or more formally for each pixel x_i , the network predicts a probability $p(x_i)$ to which the pixel belongs to a pig (with the corresponding opposite probability $(1 - p(x_i))$ that the pixel belongs to the background). Despite the simplicity of the task and the low amount of information resulting from the segmentation, binary segmentation is the basis of many classical approaches to pig detection [MS95; SX08; KBO+13; NRH+15a]. Since the binary segmentation can be produced in parallel with other outputs, it is also suitable as a support for more complex tasks. For example computationally-intensive post-processing can be applied to pixels only that actually represent the animals as shown in Section 8.1.3.

The training data consist of pairs of input images X with a corresponding binary label image y (see Figure 8.1b), where each pixel in the label image is a binary variable y_i , indicating whether the pixel belongs to the background (value 0) or to a pig (value 1). The output of the network \hat{y} is an image with the same spatial resolution as the input and one color channel containing the prediction of the network for each pixel. To produce this output, the *binary segmentation head* is composed from a single convolutional layer with only one output featuremap and a subsequent *Sigmoid* activation function which squashes the output in the range $[0, 1]$.

A commonly used loss function for binary classification is the *cross-entropy* or *log-loss*, which is defined as a sum over the N pixel of the image:

$$L = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i) \quad (8.1.1)$$

Cross-entropy loss increases as the predicted probability diverges from the actual label and thereby helps the optimizer to converge. A short introduction on how loss functions work and how neural networks are trained is given in Appendix A.

8. Detection-Based Tracking

Center Segmentation

The binary segmentation can be used to identify the areas where pigs are located, but is unsuitable for the discrimination of the individual animals. To achieve this, the segmentation can be extended by additional classes. The newly added classes encode then information about the affiliation of individual pixels to individual animals. In Section 3.2.4, some such approaches were presented in the paragraph on instance segmentation. *Instance center direction* of Uhrig et al. [UCF+16], for example, uses the different classes to work out the outlines of the searched objects. For each object in the scene, the center point is determined and starting from this, all pixels belonging to the object are divided into classes, depending on the angle of their position relative to the determined center point. As the direction of border-pixel of two touching objects are contrary regarding their direction to the objects center, the border will be outlined clearly. The authors of the original work used nine classes (eight angular ranges of 45° each and one background-class).

Here a slightly simpler approach of a distance-based classification is proposed. It encodes the distance to the pigs center in discrete steps with the three classes *background*, *outer edge of an animal*, and *inner core of an animal* (see Figure 8.1c) whereby the inner-core area is just a scaled down version of the original ellipse. Since the output of the network is still interpreted as a probability distribution, the training data must be encoded as categorical labels if there are more than two classes. The probability distribution is represented in the label images with a one-hot encoded vector y_i at each pixel, indicating one positive class and two negative classes. Correspondingly, the *center segmentation head* has three featuremaps and is equipped with a *Softmax*-activation function. This ensures that the predicted values of the network \hat{y}_i at each pixel are transformed into a valid and normalized probability distribution over the three classes. The loss-function is just a generalized multi-class version of *cross-entropy* and sums over the N pixel and the number of classes C :

$$L = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^C y_{i,j} \cdot \log(\hat{y}_{i,j}) \quad (8.1.2)$$

The probabilities can be afterwards converted into an image containing

8.1. Detection of Pigs with Panoptic Segmentation

the discrete class-labels at the pixel-positions by the *argmax* operation. In the prediction, each blob of pixels with the *core* label can be interpreted as the representation of one individual pig. If two or more animals stand close together, the pixels of the class *edge* form a joint area, but by enlargement of the ellipse-shaped cores the boundaries can be determined (see Figure 8.1c).

Head Segmentation

By the choice of the defined classes and their semantic meaning, different problems can be solved. Thus, categorical segmentation can also be used to obtain additional information about the detected pigs. One problem with ellipses as an animal model is the symmetry with respect to the major axis. After an ellipse is fitted to the segmentation of each animal in post-processing, there is a 180 degree ambiguity with respect to the orientation of the animal. However, during manual annotation, the orientation was correctly captured because the major axis was always drawn starting at the head of the animal (see Section 6.5). In order to recover this information, categorical segmentation can be used to distinguish between background, the head and the trunk of the animals (see Figure 8.1d). However, if the animals are close together, the segmentation of the body parts merges. This means that the individual pixels can no longer be assigned to the animals. In combination with the instance segmentation, the information can still help to determine the orientation of the detected animals.

8.1.3 Instance Segmentation

Even though detection with categorical segmentation works in principle, there are situations where detection of certain pigs is not possible. A particularly demanding situation for the segmentation of the individual animals are the resting phases in which the animals lie down close to each other. As the animals crawl over each other due to the confined space, very complex occlusion patterns are common. When the bodies of two pigs cross each other, several unconnected areas are created for the covered animal. Such an example is shown in Figure 8.3.

To effectively force the network to accurately map each pixel to the

8. Detection-Based Tracking

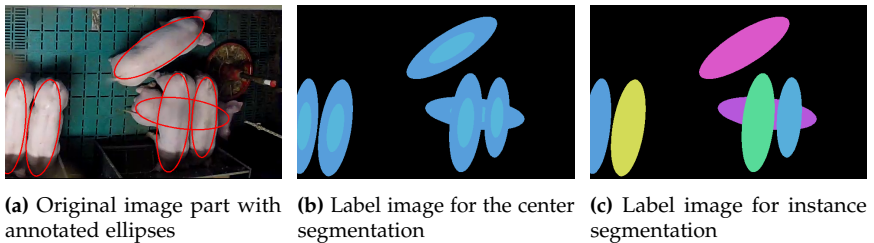


Figure 8.3. Example of fragility of center segmentation in the presence of heavy overlap. When the center of the animals is not visible, the segmentation cannot provide meaningful information about the hidden animal (b). While instance segmentation (c) can theoretically provide unambiguous assignment in such cases.

associated animal, De Brabandere et al. [DNV17] proposed a loss function that maps each pixel in the image to a position in a high-dimensional feature space (or pixel embedding). A high-dimensional feature space makes sense because the occlusion is actually created by projecting the three-dimensional space into the two-dimensional camera image. So just extending the feature space by only one dimension can already help to resolve the occlusion by separating the pixels of the affected pigs from each other in depth (i.e., the third dimension). With more dimensions there are even more possibilities to group the related pixels. Generally speaking, the network learns to place all pixels belonging to a certain object as close to each other as possible. In this way they form clusters, which can be found in post-processing with classical clustering methods. In order to avoid ambiguities, an additional requirement is that the clusters of different objects in the feature space are as far apart as possible (see Figure 8.4). The idea of this two push and pull forces was proposed by Weinberger and Saul [WS09] to learn appropriate distance metrics for clustering.

De Brabandere et al. added an additional term, which prevents the degeneration of the representation in the feature space. In total, the loss function combines three terms:

1. **Variance term** The variance term penalizes the spatial variance of the pixel embeddings belonging to the same instance. For all pixels that belong to the object (according to the annotated data), the mean is

8.1. Detection of Pigs with Panoptic Segmentation

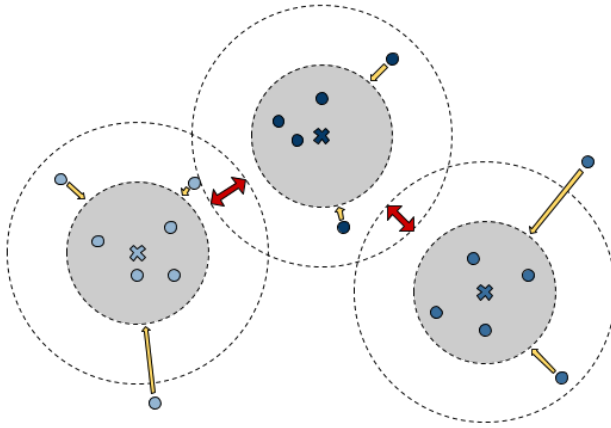


Figure 8.4. Illustration of the forces acting on the pixels to form the clusters (image adopted from [DNV17]): The variance term (yellow arrows) is used to pull the pixels towards the cluster center (crosses) while the distance term (red arrows) ensures a minimum distance between the different clusters. Both forces are only active as long as the threshold values are not reached (inner circle for cluster variance and outer circle for distance) [BGT+20]

calculated and then its distance of all object-pixels is evaluated. This forces the points in the feature space to cluster.

2. **Distance term** The distance term keeps the calculated means of the clusters at a distance.
3. **Regularization term** The regularization term keeps the expansion of all points in the feature space within limits and prevents them from drifting apart.

During the manual annotation, the ellipses were stored in such a way that the exact assignment is possible for each pixel even in the case of overlaps (see Figure 8.3c). Based on this data, the respective associated pixels can be determined and evaluated for all objects in the image.

For a d -dimensional feature space, the number of feature maps of the last convolutional layers of the *pixel embedding* network head is also

8. Detection-Based Tracking

d , matching the dimensionality of the feature space. Since the spatial resolution of the network output corresponds to that of the input image, a d -dimensional vector y_i can be extracted along the feature maps axis for each pixel and interpreted as a point in the feature space. Since the points do not represent probabilities, but should use the space without constraints, a final activation function is no longer necessary. Following the definition of [DNV17], these points are then evaluated in the loss function. In each training image there are C objects (the pigs and the background) and for each of the objects $c \in C$, the set of corresponding pixel positions N_c is known. The points y_i predicted by the network at the corresponding pixel positions form a cluster in the pixel embedding, which has its mean at the point μ_c . The clusters are then evaluated according to the three terms described above, whereby the loss is hinged to be less constrained in the representation. The pixel embeddings of the objects don't need to converge to exactly one point but should reach a distance below a threshold δ_v . In the same way, the distance between two different mean embeddings need only be greater than or equal to the threshold δ_d . This is mapped with the hinge-function $[x]_+ = \max(0, x)$. With this, the three terms are given as:

$$L_{var} = \frac{1}{C} \sum_{c=1}^C \frac{1}{N_c} \sum_{i=1}^{N_c} [\|\mu_c - y_i\| - \delta_v]_+^2 \quad (8.1.3)$$

$$L_{dist} = \frac{1}{C(C-1)} \sum_{\substack{c_A=1 \\ c_A \neq c_B}}^C \sum_{c_B=1}^C [2\delta_d - \|\mu_{c_A} - \mu_{c_B}\|]_+^2 \quad (8.1.4)$$

$$L_{reg} = \frac{1}{C} \sum_{c=1}^C \|\mu_c\| \quad (8.1.5)$$

where $\|\cdot\|$ is the L1 norm. The final loss function L with weights α, β and γ is given as:

$$L = \alpha \cdot L_{var} + \beta \cdot L_{dist} + \gamma \cdot L_{reg} \quad (8.1.6)$$

Post-Processing

To recover the individual pixel segmentations from the pixel embedding, a post-processing step is needed to find the clusters in the embedding space. As the loss-function (Equation (8.1.6)) forces the related points to move as close together as possible in the high-dimensional feature space, the clusters can be found using a density-based clustering method like HDBSCAN [CMS13]. It starts with a thinning of the non-dense areas. Then, the dense areas are linked to a tree, which is converted into a hierarchy of linked components. Thus, a condensed cluster tree can be created by the parameter of minimum cluster size, and from this tree, the final flat clusters can be extracted. The results are cluster-labels for all the pixels in the image or a special “noise” class if the pixel does not belong to a dense area in the feature space. Based on this cluster labels the individual pigs can be extracted. The background naturally forms the largest cluster and can therefore be easily identified and treated separately. For the remaining clusters, an ellipse need to be found as the object model to represent the found pig. For this, all the pixels belonging to a cluster are marked and an ellipse is fitted with the algorithm of Fitzgibbon [FF+96] ¹.

Combined Segmentation

Even if clustering methods like HDBSCAN are highly optimized, images have large amounts of data points due to their two-dimensional shape. For example, cameras with HD resolution already have up to one million points in pixel embedding. To accelerate post-processing steps such as clustering with HDBSCAN, the information contained in the base network can be evaluated and combined in parallel with different network heads.

For this, the different heads are connected in parallel to the output of the auto-encoder, sharing the base network and updating its weights simultaneously. The two heads are trained with the appropriate loss functions and the network thus produces two outputs at the same time, without much additional computational overhead. A remaining problem is balancing the heads, so adjustments to the weights in the base network help the optimization of both problems equally. A possible solution to this

¹Implemented in the OpenCV library [Bra00]

8. Detection-Based Tracking

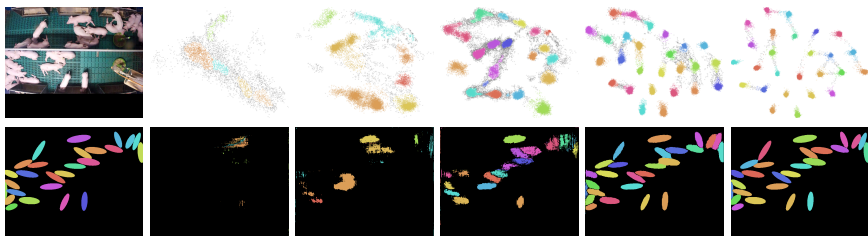


Figure 8.5. Visualization of the clustering of the combined segmentation. The original input image is shown on the left with the ground truth label below it. The top row depicts the two-dimensional embedding space. The bottom row shows the corresponding binary segmentation and cluster assignment. Snapshots are taken after 1, 2, 3, 10, and 80 gradient updates. The network was trained exclusively on this one input image to produce the results shown here for illustration.

problem is, for example, to weight the gradient updates according to the uncertainty of the loss functions [KGC18].

To speed up the clustering, a binary segmentation as described in Section 8.1.2 can be used, so that only the pixel embeddings of the animals have to be evaluated. All pixels of the background (which make up a large part of the image) can then be disregarded during clustering. Another way to incorporate additional information is head segmentation (see Section 8.1.2). After the ellipses are detected based on the instance segmentation, the orientation for each animal can be determined individually. This is done by overlaying the ellipses as masks over the output of the head segmentation and evaluating the predicted pixels of the class *head* at both ends of the ellipse major axis. By merging the two classes *head* and *trunk*, the head segmentation can additionally be easily transformed into a binary segmentation, so that the previously described speed-up of the clustering is also possible in this case. Figure 8.5 shows an example of the distribution of pixels in a two-dimensional embedding and the clustering applied to the binary segmentation.

8.2 Tracking by Detection

To combine the detections found by the neural network classical *tracking by detection* methods can be applied. As described in Section 6.4 neither an appearance nor a meaningful motion model was applicable. In addition, there are often occlusions because the pigs crawl on top of each other or parts of the pen protrude into the camera's field of view. Such occlusions result in false negative responses of the detector and need to be handled by the tracking algorithm.

The tracking of the pigs is performed using three different approaches and evaluated according to the metrics described in Section 5.2. All three methods are optimization-based methods (see Section 4.2) and show approaches of different complexity. As a baseline for the evaluation of the tracking, the temporally local assignment of detections between two consecutive frames (bipartite matching) is given (Section 8.2.2), as it is used in many of the related works (see Section 6.2). The second approach described in Section 8.2.3 is based on the idea of *Unified Hierarchical Multi-Object Tracking* (UHMOT) [HHR13] and extends the bipartite matching hierarchically to a global optimization context. Also operating globally is the third method (Section 8.2.4), which builds a complete graph of all possible trajectories and then uses min-cost-max-flow optimization [ZLN08] to find a reasonable solution.

All three methods try to link the available detections in different images in such a way that meaningful trajectories of the individual animals are created. Using the formal definition from Section 4.2, this can be defined as the assignment hypothesis $\mathbf{S}^*_{1:T}$ (Equation (4.2.1)) that best explains the detections present over all T frames.

8.2.1 Transition Costs

To transform the formal MAP problem into a concrete optimization problem, instead of using the *transition probabilities* $P_{link}(o^j | o^i)$ between two detections (see Equation (4.2.3)), the negative log probabilities $-\log(P_{link}(o^j | o^i))$ are used as *transition costs*. Just as the *transition probabilities* determine how likely two detections are to represent the same animal at different points in time, they serve as a metric that quantifies

8. Detection-Based Tracking

the association of two detections.

Due to the lack of an appearance model (see Section 6.4), tracking is based here only on the pigs' geometric pose information. This is of course only possible if the positions do not have any ambiguity from a too low temporal resolution, but as already stated in Chapter 7 about the detection-free tracking approach, the data sets used meet this requirement. The transition cost between two detections is determined based on the similarity of their model parameters. In addition to the five ellipse parameters, other features such as the spatial and temporal distance of the two detections are considered for the evaluation. Thereby, dependencies between the parameters also arise, because larger changes of the parameters are more likely to be expected with a larger temporal distance, for example, than between two directly successive images. Finding the optimal weighting of these different parameters is a classic machine learning problem and can be approached with a variety of known methods. Using the available ground truth positions and the associated target IDs, a binary classification problem is therefore set up to distinguish correct mappings between pairs of detections from incorrect mappings. Each data point in the classification problem consists of the difference information of the two detections, as well as the frame-gap, which is the temporal distance between the two detections in the sequence. The feature vector of a pair of detections consists of the *Intersection over Union* (IoU), the difference in the ratio of length to width of the ellipses, the difference in position, and the difference in rotation. A *gradient boosted decision tree* [Fri01] is used as a classifier, which generates weak classifiers based on individual elements of the feature vector and then determines the weighting of the individual classifiers to fit the predictions to the given labels of the training data set. The classifier's training data is thereby generated by evaluating all possible pairs of detections over a time window (e.g., 5 frames) for a randomly selected portion of the annotated images and labeling them as correct or incorrect matches according to the available target IDs.

As formally described above, the negative log-likelihood is used to transform the binary classification problem into a transition cost function. Thus, instead of the discrete prediction of the trained classifier, its computed probability that a data point belongs to the class *correct assignment* is queried.

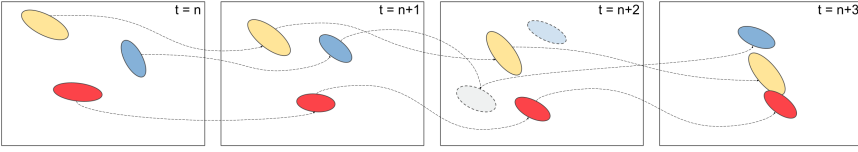


Figure 8.6. Example of bipartite matching between successive frames. In time step $n + 2$ a false assignment occurs, which is caused by a false-negative detection (faded blue ellipse) and a false-positive detection (faded gray ellipse) occurring simultaneously.

8.2.2 Bipartite Matching between Frames

With the transition cost metric, the transition from one frame to the next is essentially a simple weighted assignment problem on the available detections and can be solved using graph theory methods. Formally, a bipartite graph $G = (V, E)$ with edges E and vertices V is defined, whose vertices are split into two sets U_1 and U_2 such that $U_1 \cup U_2 = V$ and $U_1 \cap U_2 = \emptyset$. The two sets correspond to detections in the two consecutive frames and for this bipartite graph holds:

$$\forall \{u_1, u_2\} \in E : (u_1 \in U_1 \wedge u_2 \in U_2) \vee (u_1 \in U_2 \wedge u_2 \in U_1) \quad (8.2.1)$$

If the bipartite graph is fully constructed so that each detection in the first frame is connected to each detection in the second frame by an edge, each of these edges can be associated with the corresponding transition cost between the connected detections. The cost of the edges can then be described by a cost matrix C whose entries $c_{i,j}$ are the cost of matching detection i in the first frame with detection j in the second frame. The optimal mapping is then a complete unique assignment of all detections where the total cost is minimal. Formally described, an assignment matrix X is introduced with $x_{i,j} = 1$ iff the i -th detection in the first frame is assigned to the j -th detection in the second frame. The optimization problem for the optimal assignment is then given by:

$$\min \sum_i \sum_j c_{i,j} x_{i,j} \quad (8.2.2)$$

Such assignment problems with quadratic cost-matrices can be solved with

8. Detection-Based Tracking

the Hungarian method [Kuh55]. Due to detection errors or occlusions, the number of detections in two consecutive frames is not necessarily equal, so this optimization problem with its rectangular cost-matrix is an extension of the classical linear assignment problem [BL71]. Similarly, it is important to delete edges from the graph whose cost is above a threshold (pruning) to avoid unreasonable assignments. Otherwise, due to the unique assignment, after the unambiguous detections are assigned, all remaining detections would somehow be linked even if this clearly contradicts the motion model of the animals. Figure 8.6 depicts such a case to be prevented, where the simultaneous occurrence of a false negative and a false positive in a single frame leads to a wrong assignment.

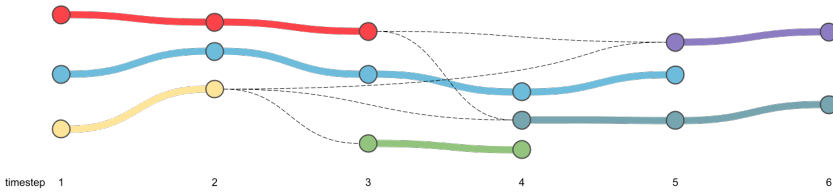
The assignments can then be concatenated frame by frame across the entire sequence to form the final trajectories of the animals. Due to the local optimization from frame to frame, errors in the detection have a correspondingly strong effect on the final result. If a detection is missing in one of the frames, one of the existing trajectories inevitably ends and if the animal is detected correctly again in the next frame, a new trajectory is started with the result that the identity of the target is lost.

8.2.3 Global Bipartite Matching

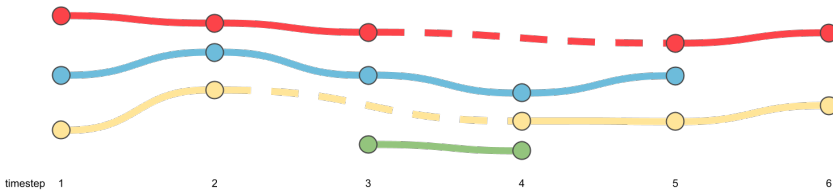
The deficiencies of local optimization can be addressed with a hierarchical approach by dividing the tracking problem into several stages [HHR13]. In a first stage, the associations are performed locally as described in the previous section, generating meaningful but possibly interrupted trajectories (called *tracklets*).

To close the gaps, the start and end points of all tracklets found in the first stage are processed in a second stage and an attempt is made to merge them to longer trajectories (see Figure 8.7). For this the conditions of the optimization problem are relaxed, so that also end points of tracklets can be linked with start points of other tracklets, which do not start in the directly following frame. When creating the transition cost classifier (see Section 8.2.1), the temporal differences of the detections (frame-gap) are already included in the calculation, so the same transition cost function can be used in the second stage. Tracklets whose endpoints are in the same time step cannot be merged because the tracking procedure should not

8.2. Tracking by Detection



(a) Generated *tracklets* after the first stage of bipartite matching.



(b) Final trajectories after the second stage. Dashed lines mark the successfully closed gaps.

Figure 8.7. Graphical illustration of the hierarchical application of bipartite matching. In the first step, local linking is performed between pairs of images. In the second step, tracklets are also linked across multiple time steps to close the gaps.

change the underlying detections. Likewise, no interpolated detections are artificially added along newly added linkages (as in [HHR13]). This second stage could be run multiple times by increasing the threshold for the maximum temporal distance between tracklets to be merged. However, this did not produce any significant improvements, so here the second stage was applied once without any frame-gap constraint.

Theoretically, there would be further steps to refine the results. For example, in the second stage, only tracklets that have a minimum length could be considered. This would prevent that single false-positive detections are woven into trajectories that are actually correct, if they happen to have a gap in the corresponding frame. Such additional steps were also not considered here in order to avoid increasing the complexity of the individual procedures.

8. Detection-Based Tracking

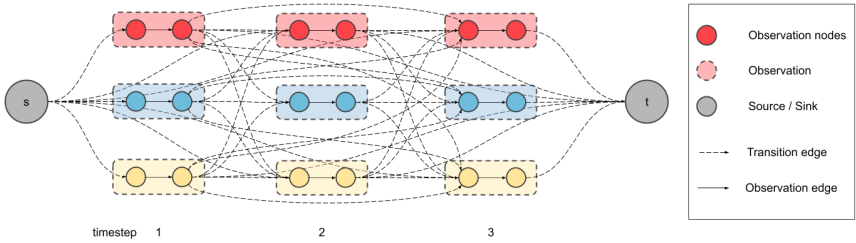


Figure 8.8. Example of a flow network for the min-cost flow method. Each observation is modeled internally with two nodes and is connected to all observations in previous and following time steps (within a time window).

8.2.4 Minimum-Cost Flow

The assignment can be solved even more holistically if all detections are linked to all theoretically possible successors in the following frames (in a specified time window). Thus, a graph with all possible trajectories over the complete sequence is created as described in [ZLN08]. By building such graph, the problem of finding the best fitting representation $\mathbf{S}^*_{1:T}$ given all observations $\mathbf{O}_{1:T}$ can be transformed in a min-cost-max-flow graph optimization problem. Let $G = (V, E)$ be the graph with edges E and vertices V and $c : E \rightarrow \mathbb{R}^+$ defines a capacity-function, then $N = (G, c, s, t)$ is called a *flow network* with two distinguished nodes $s \in V$ and $t \in V$ called *source* (s) and *sink* (t). Given a flow function $f : E \rightarrow \mathbb{R}^+$ and a cost function $a : E \rightarrow \mathbb{R}^+$, the cost of sending a flow over an edge is defined as $a \cdot f$. The *minimum-cost flow* problem then consists of sending a fixed amount of flow through the network (from source to sink) while minimizing the cost:

$$\min \sum_{e \in E} a(e)f(e) \quad (8.2.3)$$

Such *minimum-cost flow* problem can be solved with the *simplex*-algorithm [Orl97], *search-trees* [Tar97] or a *k-shortest path solver* [BFT+11].

With the definition of the flow network, multi-object tracking can now be represented as follows. Each trajectory is represented by a flow path in the network and the amount of flow between source and sink corresponds

8.2. Tracking by Detection

to the number of objects in the tracking problem. An example flow network is depicted in Figure 8.8. By representing each detection by two nodes connected by a single edge (*observation edge*), the constraint that each detection is assigned to only one trajectory can be easily implemented by assigning all observation edges a capacity of 1. The flow conservation constraint states that the amount of incoming flow must be equal to the amount of outgoing flow. Thus, only one of the incoming edges and one of the outgoing edges of each detection can be assigned a flow. More formally this can be written as ([ZLN08]):

$$f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j}, \quad \forall i \quad (8.2.4)$$

whereby the following 0-1-indicators are used:

$$\begin{aligned} f_{en,i} &= \begin{cases} 1 & \exists \mathbf{T}^k \in \mathbf{S}_{1:T}, \mathbf{T}^k \text{ starts from } o_i \\ 0 & \text{otherwise} \end{cases} \\ f_{ex,i} &= \begin{cases} 1 & \exists \mathbf{T}^k \in \mathbf{S}_{1:T}, \mathbf{T}^k \text{ ends at } o_i \\ 0 & \text{otherwise} \end{cases} \\ f_{i,j} &= \begin{cases} 1 & \exists \mathbf{T}^k \in \mathbf{S}_{1:T}, o_j \text{ is right after } o_i \text{ in } \mathbf{T}^k \\ 0 & \text{otherwise} \end{cases} \\ f_i &= \begin{cases} 1 & \exists \mathbf{T}^k \in \mathbf{S}_{1:T}, o_i \in \mathbf{T}^k \\ 0 & \text{otherwise} \end{cases} \end{aligned} \quad (8.2.5)$$

Each unit of flow sent from the source s to the sink t represents a trajectory and the total cost along the path corresponds to the log-likelihood of the association hypothesis. As in bipartite matching (Section 8.2.2), the transition cost function defined in Section 8.2.1 can be used to set the transition cost $c_{i,j}$ between two detections. However, to prevent the trivial empty zero-cost flow as a solution to the minimization problem, each observation o_i is represented by two nodes, connected with an edge with negative *observation cost* c_i . As a result, the total cost of a trajectory can become negative if it combines as many detections as possible.

Using the indicators in Equation (8.2.5) and starting from Equation (4.2.2), the cost to sent the flow through the graph is defined as ([ZLN08]):

8. Detection-Based Tracking

$$\begin{aligned}
\mathbf{S}_{1:T}^* &= \operatorname{argmax}_{\mathbf{S}_{1:T}} \prod_i P(o_{1:T}^i | \mathbf{S}_{1:T}) \prod_{\mathbf{T}^k \in \mathbf{S}_{1:T}} P(\mathbf{T}^k) \\
&= \operatorname{argmin}_{\mathbf{S}_{1:T}} \sum_i -\log(P(o_{1:T}^i | \mathbf{S}_{1:T})) + \sum_{\mathbf{T}^k \in \mathbf{S}_{1:T}} -\log(P(\mathbf{T}^k)) \quad (8.2.6) \\
&= \operatorname{argmin}_{\mathbf{S}_{1:T}} \sum_i c_i f_i + \sum_i c_{en,i} f_{en,i} + \sum_i c_{i,j} f_{i,j} + \sum_i c_{ex,i} f_{ex,i}
\end{aligned}$$

The edges between the source or sink and the detections allow solutions where trajectories start or end within the sequence. In [ZLN08] the probabilities for the entry and exit of targets are estimated from training data, but as the number of pigs in the videos is static per pen, the entry- and exit-probability is infinitesimal here. Therefore, the entry- and exit-costs $c_{en,i}, c_{ex,i}$ were set to a high value (e.g. 10) to penalize interruption of trajectories.

8.3 Evaluation

Since the detections and subsequent tracking are performed independently, the evaluation is also performed separately.

8.3.1 Evaluation of Detection Methods

To evaluate the performance of the proposed detection framework, the different methods described in Section 8.1.2 and Section 8.1.3 are evaluated in detail. A first visual impression of the results is given in Figure 8.9. For all evaluations an auto-encoder based on U-Net [RFB15] was used, as its modular design allows the use of different classification architectures as backbone (as described in Section 8.1.1). To investigate the influence of the different backbone all experiments were conducted with a ResNet34 [HZR+16] and an Inception-ResNet-v2 [SIV+17] backbone. They both consist of blocks of layers that combine different convolution operations with a shortcut connection. With these shortcut connections, the optimizer does not have to learn the underlying mapping of the data but simply a residual function [HZR+16] (see Appendix A). While the ResNet34

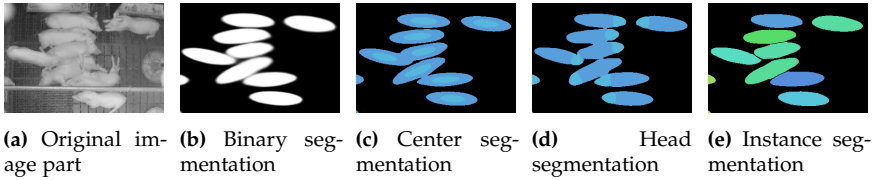


Figure 8.9. Demonstration of the results of the different detection methods on an example image (cropped) (a). Shown are the results of binary segmentation (b), center segmentation with the classes *outer edge of an animal* and *inner core of an animal* (c), body part segmentation for orientation recognition with the classes *head* and *rest of body* (d) and the combined segmentation (e). [BGT+20]

with 16 of such blocks and its 24.5M parameters serves as baseline, the more sophisticated Inception architecture (62.2M parameters) is aimed at leveraging the potential capabilities of modern classification networks. The network was implemented with the *segmentation models* library [Yak19]. For all experiments, the *Adam*-Optimizer [KB14] with an initial learning rate of $1e-4$ was used. The networks are trained with the designated training images of the respective data set and the evaluation takes place exclusively on the images of the test set (see Section 6.3.2). To speed up the calculation of the network and any subsequent clustering, the images were scaled down to a resolution of 640×512 pixels.

Categorical Segmentation

As described in Section 8.1.2, the animals can be detected by having the network predict the classes *outer edge of an animal* or *inner core of an animal* for each pixel. However, since the described framework can also generate a binary segmentation and such a segmentation can be used in other methods (e.g. in the detection free tracking described in Chapter 7), the quality of the binary segmentation is evaluated here first. In binary segmentation, the network predicts a probability that a particular pixel belongs to a target or the background. Using a threshold value of 0.5, this probability is converted into a final decision and can then be compared with the ground truth values. The accuracy is defined as the number of pixels where the prediction matches the ground truth, normalized over

8. Detection-Based Tracking

the number of all pixels in the image. The results are listed in Table 8.1.

Table 8.1. Accuracy results of the binary segmentation experiment. The experiment was additionally carried out separately on the daylight (D) and night vision (N) images of the second data set. Best results in bold.

Data Set	Backbone	Acc \uparrow	Acc (D) \uparrow	Acc (N) \uparrow
#1	ResNet34	0.9772	-	-
#1	Incep.-RN-v2	0.9760	-	-
#2	ResNet34	0.9730	0.9771	0.9692
#2	Incep.-RN-v2	0.9735	0.9774	0.9699

For center segmentation, the class *inner core of an animal* was set to 50% of the size of the ellipses (see Figure 8.9c). Extracting the ellipses with a blob search as described produces the results listed in Table 8.2. In this case, the segmentation contains several classes, but the calculation of accuracy does not change. Since the label image also consists of the same classes, the frequency with which the individual pixels match the corresponding class in the label image is counted, normalized over the number of all pixels in the image. The detection-metrics (*Precision*, *Recall*, *F1*, and *PQ*) are explained in details in Section 5.1. For all metrics the threshold for the *IoU*-Score was set to 0.5.

Table 8.2. Categorical segmentation accuracy and detection metric results of the center segmentation experiment. Best results in bold.

Data Set	Backbone	Acc \uparrow	Prec \uparrow	Recall \uparrow	F1 \uparrow	PQ \uparrow
#1	ResNet34	0.9639	0.9716	0.9439	0.9575	0.7938
#1	Incep.-RN-v2	0.9637	0.9693	0.9401	0.9545	0.7892
#2	ResNet34	0.9612	0.9586	0.9514	0.9550	0.7920
#2	Incep.-RN-v2	0.9612	0.9577	0.9505	0.9541	0.7943

In order to determine whether the massive image distortions in the night-vision images of the second data set (see Figure 6.3) has a negative impact on the network performance, Table 8.3 lists the results separately for the day and night images of the test set.

Table 8.3. Categorical segmentation accuracy and detection metric results of the center segmentation experiment carried out separately on the daylight (D) and night vision (N) images of the second data set.

Backbone	Acc (D) \uparrow	Acc (N) \uparrow	F1 (D) \uparrow	F1 (N) \uparrow	PQ (D) \uparrow	PQ (N) \uparrow
ResNet34	0.9664	0.9565	0.9619	0.9487	0.8124	0.7738
Incep.-RN-v2	0.9664	0.9564	0.9614	0.9475	0.8165	0.7742

Instance Segmentation

The last experiment uses the instance segmentation described in Section 8.1.3 (see Figure 8.9e) with an eight-dimensional pixel embedding in combination with a binary segmentation (combined segmentation). The thresholds in the discriminative loss in Equation (8.1.3) and Equation (8.1.4) were set to $\delta_v = 0.1$ and $\delta_d = 1.5$. The weights in the final loss term in Equation (8.1.6) were set to $\alpha = \beta = 1.0$ and $\gamma = 0.001$. Most values were taken from the original paper [DNV17], except for the threshold δ_v . For clustering, the HDBSCAN implementation from McInnes et al. [MHA17] was used (in contrast to a variant of the *mean-shift* algorithm in the original paper), the threshold was decreased to improve the density-based clustering of HDBSCAN. The minimal cluster size for HDBSCAN was set to 100. The influence of all these hyperparameters is additionally evaluated in the following ablation studies.

Table 8.4. Detection metric results of the instance segmentation with the combined approach. The accuracy is given for the binary segmentation, produced by the segmentation head.

Data Set	Backbone	Acc \uparrow	Prec \uparrow	Recall \uparrow	F1 \uparrow	PQ \uparrow
#1	ResNet34	0.9757	0.8433	0.8698	0.8563	0.7071
#1	Incep.-RN-v2	0.9735	0.9108	0.8513	0.8801	0.7216
#2	ResNet34	0.9722	0.9544	0.9482	0.9513	0.7966
#2	Incep.-RN-v2	0.9707	0.9495	0.9466	0.9481	0.7921

The results of the experiment are listed in Table 8.4. It is clearly apparent that despite the combined processing of pixel embedding and binary segmentation in a shared backbone, the accuracy of the binary segmentation is still good (compare with Table 8.1). Therefore, a synergetic effect of

8. Detection-Based Tracking

the two tasks can be assumed. To control the weighting of the two network heads and thus ensure the synergy effect, the multi-task-loss proposed in [KGC18] was additionally tested. However, this did not improve the results at all.

Again also the results for the day and night images of the test set are listed separately in Table 8.5 .

Table 8.5. Detection metric results of the instance segmentation with the combined approach carried out separately on the daylight (D) and night vision (N) images of the second data set. The accuracy is given for the binary segmentation, produced by the segmentation head.

Backbone	Acc (D) ↑	Acc (N) ↑	F1 (D) ↑	F1 (N) ↑	PQ (D) ↑	PQ (N) ↑
ResNet34	0.9761	0.9687	0.9588	0.9446	0.8181	0.7774
Incep.-RN-v2	0.9752	0.9666	0.9566	0.9404	0.8179	0.7689

Replacing binary segmentation with categorical head segmentation, the combined approach can additionally detect animal orientation. Details are described in Section 8.1.3. To evaluate the accuracy of this orientation recognition, the orientation of all correctly identified pigs (true positives) was assessed over the complete test set. The results are summarized in Table 8.6. In addition, the PQ was also evaluated to ensure that the segmentation of the body parts did not affect the overall accuracy of the position detection.

Table 8.6. Results of orientation recognition: The network can correctly recognize the orientation in 88% resp. 94% of the correctly found animals (true positive).

Data Set	Backbone	Orientation Acc ↑	Acc ↑	PQ ↑
#1	ResNet34	0.8818	0.9663	0.7521
#1	Incep.-RN-v2	0.8532	0.9652	0.7488
#2	ResNet34	0.9428	0.9644	0.7958
#2	Incep.-RN-v2	0.9226	0.9601	0.7898

Ablation Studies

The choice for the U-Net architecture is motivated, as already described in detail, by the ease of implementation with a wide variety of encoder net-

works as the backbone. To justify the choice, experiments with the *feature pyramid network* (FPN) (see Section 3.2.4) architectures are also shown here for comparison. In addition, with the *EfficientNet* [TL19], another more recent classification network as backbone is also evaluated. All experiments were performed only for the combined instance segmentation on the test set of data set #2 and with a reduced resolution of the input images of 320×256 pixels to ensure the evaluation of the many different versions in acceptable time. The values shown in Table 8.7 should therefore only be compared with each other and not with the official evaluations in the previous sections. The results are all very close, which indicates that the choice of backbone and architecture does not have a major impact. As stated in [BGT+20] also experiments with even more complex architectures like *DeepLabv3+* [CZP+18] did not show any improvements, as overfitting quickly occurred due to the small amount of training data.

Table 8.7. Results of the ablation study on the test data of dataset #2. Only the combined segmentation with a reduced image resolution of 320×256 pixels was evaluated. The results illustrate the marginal impact of the different architecture choices (U-Net vs. Feature Pyramid Network (FPN)) as well as the different backbones. Results from [BGT+20]

Instance segmentation (U-Net)	Acc \uparrow	Prec \uparrow	Recall \uparrow	F1 \uparrow	PQ \uparrow
ResNet34	0.9694	0.9559	0.9358	0.9457	0.7863
Incep.-RN-v2	0.9674	0.9501	0.9157	0.9326	0.7685
EfficientNet-B5	0.9692	0.9471	0.9337	0.9404	0.7768
Instance segmentation (FPN)	Acc \uparrow	Prec \uparrow	Recall \uparrow	F1 \uparrow	PQ \uparrow
ResNet34	0.9709	0.9556	0.9332	0.9442	0.7824
Incep.-RN-v2	0.9700	0.9511	0.9319	0.9414	0.7784
EfficientNet-B5	0.9709	0.9556	0.9347	0.9451	0.7861
Mask R-CNN	Acc \uparrow	Prec \uparrow	Recall \uparrow	F1 \uparrow	PQ \uparrow
ResNet101	-	0.9466	0.9363	0.9414	0.7474

Table 8.7 also lists the results of Mask R-CNN [HGD+17] (with a ResNet101 backbone). Mask R-CNN is a representative of the classic two-stage detection networks (see Section 3.2.3), which individually evaluates the bounding boxes generated by a region proposal branch. Mask R-CNN additionally learns to generate segmentation masks for the detected objects, allowing the pixel masks produced in this way to be directly compared

8. Detection-Based Tracking



(a) Grid search for loss parameters, measured in Panoptic Quality (PQ) (b) Evaluation of the minimum cluster size, measured in Panoptic Quality (PQ)

Figure 8.10. (a) Evaluation of different thresholds δ_v and δ_d which control the distances in the discriminative loss. Measured in panoptic quality (PQ). (b) Evaluation of the minimum cluster size (on the reduced 320×256 px images) showing a stable interval in the range 80 to 200. [BGT+20]

with the results of the methods proposed here. The comparison shows that the much more complex architecture of Mask R-CNN (using a backbone network that is also massively larger) has no advantage in the detection problem studied here.

Hyperparameters In instance segmentation, the pixels are grouped into clusters in the embedding space to separate the individual animals from each other. The thresholds δ_v and δ_d for the discriminative loss that control the minimum spacing of these clusters are accordingly crucial for how well the animals can be detected in the post-processing step. As described in the evaluation of the instance segmentation, these were adjusted to work better with the clustering method used. However, the two parameters must actually only be sensibly adjusted to each other, as shown in Figure 8.4. Scaling the parameters ultimately only leads to a scaling of the embedding space and does not change the separation of the clusters. This assumption is supported by the grid search depicted in Figure 8.10a with $\delta_v \in [0.05, 0.25]$ and $\delta_d \in [1.0, 2.5]$ showing PQ values around 0.7931 with a standard deviation of $\sigma = 0.0027$.

The second hyperparameter that affects the detection of individual animals is the minimum cluster size. Thus, to prevent clusters from being erroneously separated or merged, this value must be approximately equal

to the average size of the representation of the targets in the image (measured in pixels). Thus the value depends accordingly also on the input size of the images and must be scaled appropriately when adjusting the resolution. As it is a minimum value, it must be chosen according to the size of the smallest animals in the processed data. As shown in Figure 8.10b, the value is stable in a large interval. For the data sets used here, a value of 100 has been proven to be reasonable, since the animals were observed over a longer period of time and therefore appear in different ages and sizes (see Figure 8.12).

Cross-Validation As described in section Section 6.3.2, part of the available data is initially defined as a test set. In order to show that the results of the presented procedure, not just randomly on the designated test data meet the expectations, a cross-validation was performed with the five cameras included in data set #2. This is done by declaring the images from one of the cameras to be the test set in each run, and the images from the four remaining cameras were used as the training and validation set.

The results listed in Table 8.8 show the expected small variation across the different data splits, which indicate robustness and good generalization of the network.

Table 8.8. Results for a five-fold cross-validation on data set #2. In each experiment, one of the cameras was declared as the test set while the images from the four remaining cameras were used for training and validation [BGT+20]

Test Set	Precision \uparrow	Recall \uparrow	F1 \uparrow	PQ \uparrow
Camera 1	0.9566	0.9575	0.9571	0.7849
Camera 2	0.9616	0.9588	0.9602	0.7976
Camera 3	0.9544	0.9482	0.9513	0.7966
Camera 4	0.9357	0.9331	0.9344	0.7653
Camera 5	0.9540	0.9548	0.9544	0.8002
Average	0.9525	0.9505	0.9515	0.7889
STD	0.0088	0.0094	0.0090	0.0129

Synthetic Data To test whether the generalization of the networks can be further increased by using additional (artificial) training data, these can simply be mixed in with the real training data during training. The input data is then augmented and preprocessed exactly as in the other

8. Detection-Based Tracking

Table 8.9. Evaluation of instance segmentation on the test set of data set #2. During training, synthetic data were mixed in to artificially increase the variance of the input samples.

Instance segment. (incl. synthetic)	Acc \uparrow	Prec \uparrow	Recall \uparrow	F1 \uparrow	PQ \uparrow
ResNet34	0.9705	0.9537	0.9379	0.9458	0.7848
Incep.-RN-v2	0.9693	0.9549	0.9302	0.9424	0.7824

experiments. Table 8.9 shows the results of instance segmentation on the test set of data set #2, which of course is not augmented with synthetic data. Unfortunately, similar to the other ablation studies, this evaluation does not show any positive effect on the results compared to the normal evaluation in the lower part of Table 8.4.

Visual Results

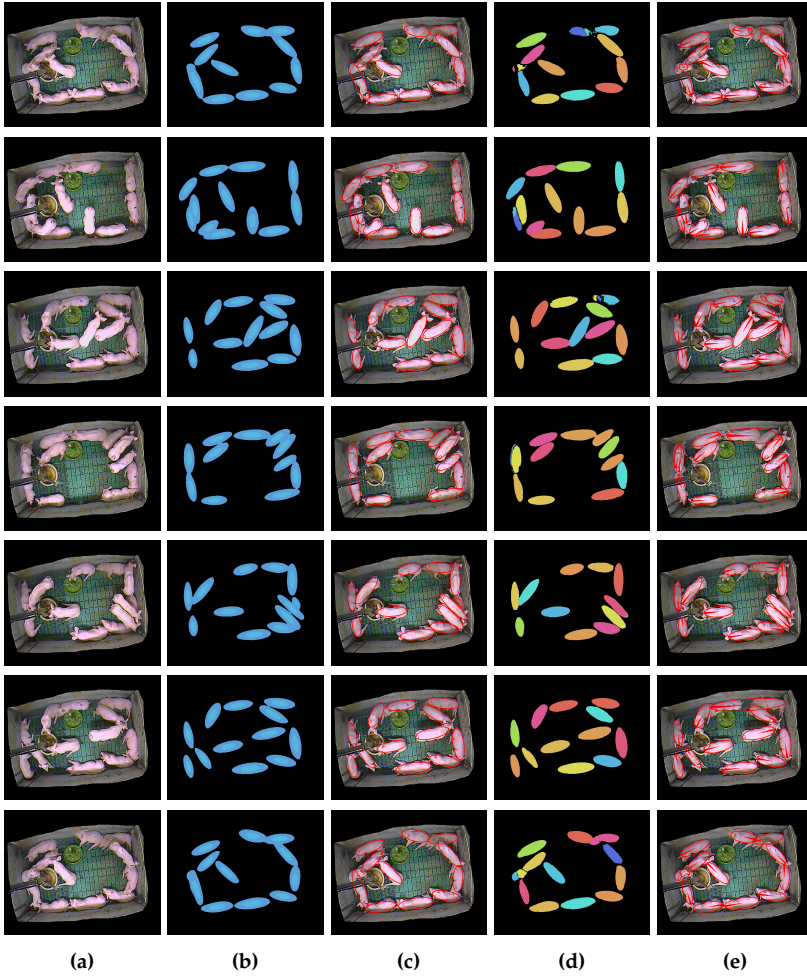


Figure 8.11. Visual results from the test set of data set #1. Each row shows the original image (a), the prediction of the center segmentation (b), the ellipses extracted from the center segmentation (c), the prediction of the combined instance segmentation (d), and the ellipses extracted from the combined instance segmentation (e) (including orientation recognition where the filled part of the ellipses shows the identified orientation of the animals.)

8. Detection-Based Tracking

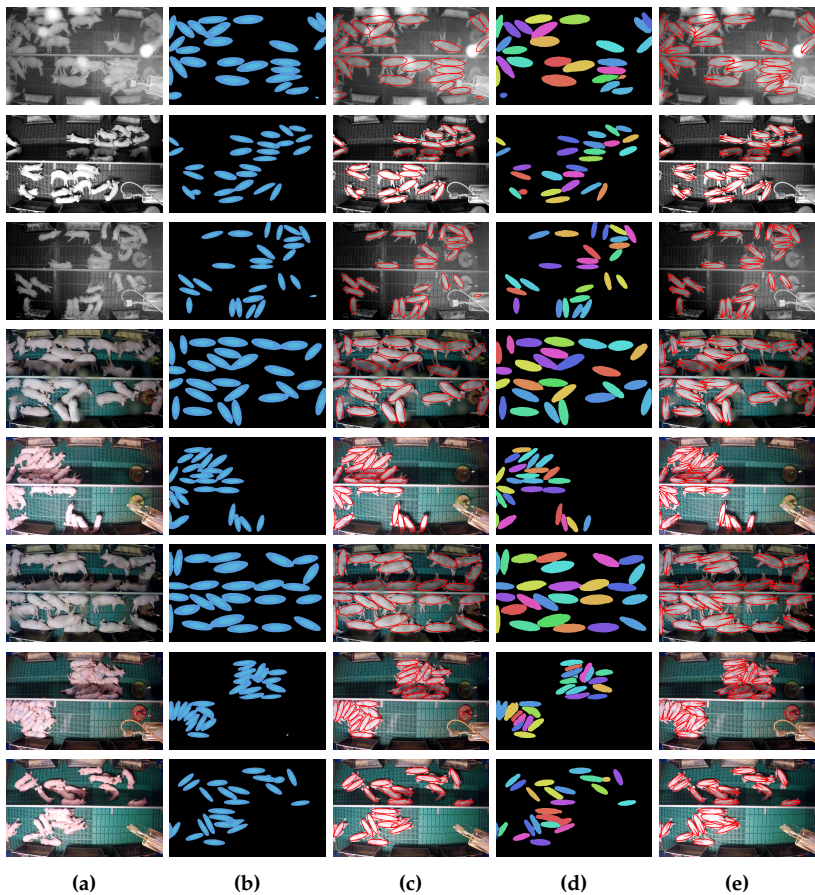


Figure 8.12. Visual results from the test set of data set #2. Each row shows the original image (a), the prediction of the center segmentation (b), the ellipses extracted from the center segmentation (c), the prediction of the combined instance segmentation (d), and the ellipses extracted from the combined instance segmentation (e) (including orientation recognition where the filled part of the ellipses shows the identified orientation of the animals.) [BGT+20]

To better visually comprehend the metrics presented in this chapter on the detection results, Figures 8.11 and 8.12 illustrate some sample images

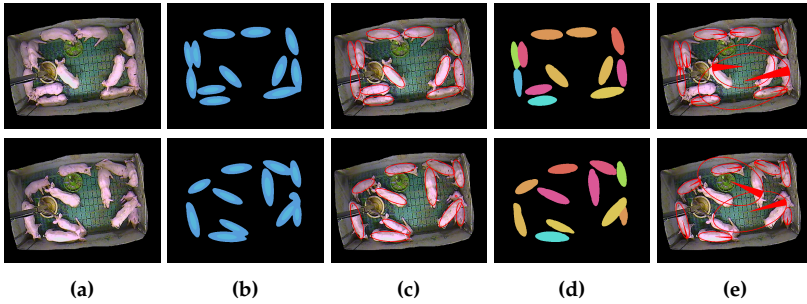


Figure 8.13. Failure cases in the assignment of the image points to the animals present in the image during instance segmentation. The membership of the image pixels to the clusters is indicated by the color (d). Since the subsequent ellipse extraction tries to combine all pixels belonging to a cluster with the ellipse, false detections occur (e).

from the test sets of the two data sets with the corresponding results. Next to the original input image are the segmentation outputs as well as the ellipses extracted from them for the center segmentation and the combined instance segmentation.

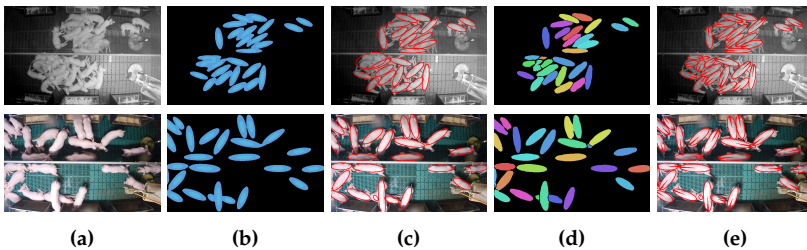


Figure 8.14. Failure cases where center segmentation fails. If the animals cross or are too close to each other, the pixels can merge with the "inner core" class of different animals or the pixel of a single animal can be separated into two parts (b). This disturbs the extraction of the ellipses (c). [BGT+20]

The values in Tables 8.4 and 8.6 have already indicated that the instance segmentation on the images of data set #1 does not give quite as good results. Figure 8.13 shows two such failure cases. Due to faulty assignment in

8. Detection-Based Tracking

the clustering of the pixels (note the colorizations in Figure 8.13d), several animals are combined to one detection, which leads to large deviations of the detected ellipses. For subsequent process steps, such false detections should be filtered out based on their strong deviations from the expected model parameters. In the same way, errors occur in center segmentation when animals are too close together, causing the inner core to merge (see Figure 8.14b).

8.3.2 Evaluation of Tracking Methods

The evaluation of the different tracking methods is done in exactly the same way as the evaluation of the detection-free tracking method in Section 7.2.2, so the results are directly comparable. While the evaluation of the different detection networks in the previous chapter Section 8.3.1 was applied on the original data split of the datasets, the evaluation of the tracking is again done on the three tracking sequences #1-#3 (see Section 6.5.1).

As described in the introduction of this section (see Chapter 8), the focus here is not on improving the tracking methods, but rather on whether the detection methods presented, provide reliable tracking of the animals. Detections were generated for this evaluation using the combined instance segmentation approach described in Section 8.1.3 (marked as ISN). Thereby, the networks were trained on the images of the corresponding dataset (dataset #1 for the first two tracking sequences and dataset #2 for the third sequence). The methods are again also applied to the ground truth data (marked as GT), which sets the upper bound of accuracy, evaluated according to the metrics presented in Section 5.2.

Multiple Object Tracking Precision (MOTP), by definition, is more of an evaluation of the quality of the detections incorporated into the tracking. In a detection-free based approach, where the positions of the targets are determined in the same step as the tracking, this evaluation has some relevance. Here in the detection-based approach, however, the detections are predetermined, so that the MOTP values are the same for all three methods examined. Nevertheless, they are included to allow comparison with the detection-free based method. On the other hand, the listing of *track quality measures* is omitted, since all three methods have such a high accuracy that all trajectories found fall into the category *mostly tracked*

(MT).

Noticeable is the 100% correct assignment of the tracking in the tracking sequence #3 on the ground truth detections for all three methods. The sequence does not seem to contain any situations that could lead to ambiguous assignments.

In all evaluations, the transition costs were generated by the same classifier (see Section 8.2.1). The separate evaluation in Table 8.13 clearly shows the suitability of this method by means of very high values in the correct assignment. For the pruning of the cost-matrix the threshold was set to 5 in all experiments.

Bipartite Matching

Bipartite matching is most comparable to the detection-free tracking approach presented in Chapter 7, as it also works purely locally, resulting in gaps in the trajectories in the case of a lack of matching detections. In direct comparison, it can be seen in Table 8.10 that even the simple bipartite matching could improve the metrics MOTA as well as *identity switches* (IDSW) and *fragmentation* (Frag). The MOTP values are not improved, but the metric, as described, refers more to the detection quality of the successful detections anyway.

As with the detection-free approach, the differences between the results using a network and those based on ground-truth data are marginal. This demonstrates the quality of the network output as input to the tracking procedures.

Table 8.10. Tracking results of the bipartite matching tracking method. The evaluation is performed separately on detections from an instance segmentation network (ISN) and on the ground truth positions (GT) as a baseline.

Sequence	Detection	MOTA \uparrow	IDSW \downarrow	Frag \downarrow	MOTP \uparrow
#1	ISN	0.9892	36	6	0.8694
#1	GT	0.9945	29	1	0.9868
#2	ISN	0.9940	12	5	0.8724
#2	GT	0.9963	10	4	0.9858
#3	ISN	0.9857	37	19	0.8267
#3	GT	1.0000	0	0	0.9942

8. Detection-Based Tracking

Global Bipartite Matching

With the second stage, which merges the potentially fragmented trajectories from the bipartite matching in a subsequent step, the results in all experiments improve as expected. In Table 8.11, the reduction of *identity switches* (IDSW) can be seen, which correspondingly also improves the MOTA value. The number of *fragmentations* (Frag), on the other hand, is not improved because, as described in Section 8.2.3, no artificial positions are inserted through interpolation.

Table 8.11. Tracking results of the global bipartite matching tracking method. The evaluation is performed separately on detections from an instance segmentation network (ISN) and on the ground truth positions (GT) as a baseline.

Sequence	Detection	MOTA \uparrow	IDSW \downarrow	Frag \downarrow	MOTP \uparrow
#1	ISN	0.9927	15	6	0.8694
#1	GT	0.9978	9	1	0.9868
#2	ISN	0.9943	10	5	0.8724
#2	GT	0.9967	8	4	0.9858
#3	ISN	0.9857	37	19	0.8267
#3	GT	1.0000	0	0	0.9942

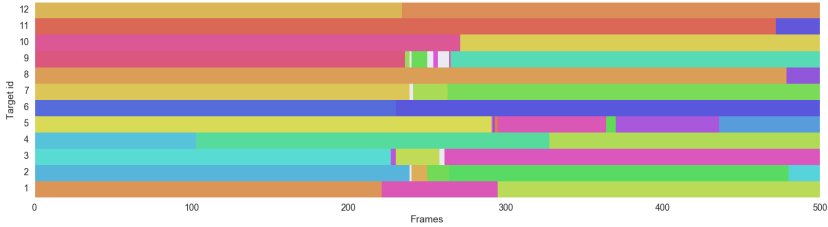
Minimum-Cost Flow

Global optimization of all possible trajectories using the minimum-cost flow method achieves similar good results as global bipartite matching. Table 8.12 lists the corresponding results.

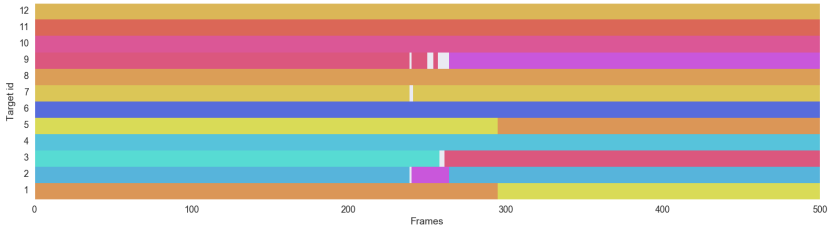
Table 8.12. Tracking results of the minimum-cost flow tracking method. The evaluation is performed separately on detections from an instance segmentation network (ISN) and on the ground truth positions (GT) as a baseline.

Sequence	Detection	MOTA \uparrow	IDSW \downarrow	Frag \downarrow	MOTP \uparrow
#1	ISN	0.9915	22	10	0.8694
#1	GT	0.9981	7	3	0.9868
#2	ISN	0.9937	14	7	0.8724
#2	GT	0.9960	12	6	0.9858
#3	ISN	0.9859	35	18	0.8267
#3	GT	1.0000	0	0	0.9942

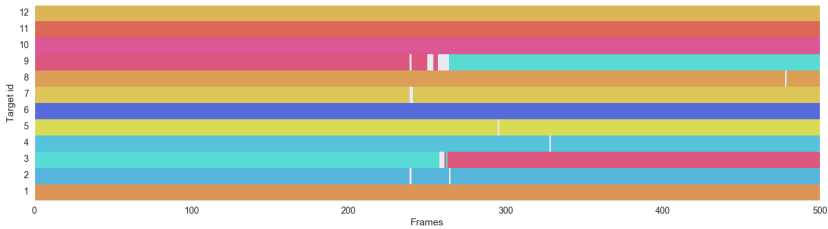
Visual Comparison



(a) Bipartite matching on network detections (ISN) for tracking sequence #1.



(b) Global bipartite matching on network detections (ISN) for tracking sequence #1.



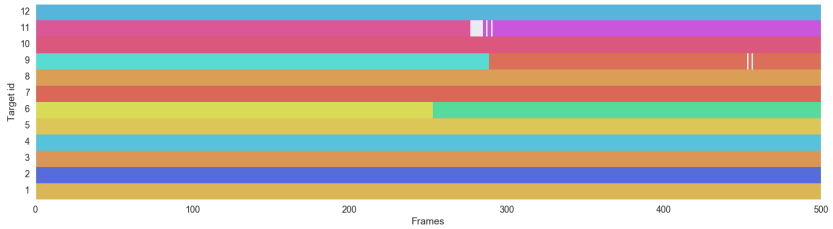
(c) Minimum-cost flow on network detections (ISN) for tracking sequence #1.

Figure 8.15. Visual representation of tracking results on network detections (ISN) for tracking sequence #1. Assigned target IDs are represented by colors to show continuity or interruptions of each trajectory.

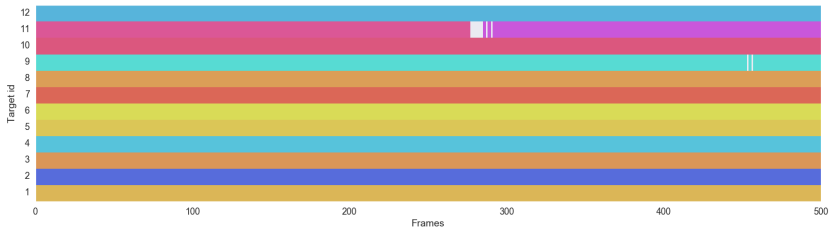
To make the tracking results more visually understandable, the target IDs assigned to each target can be plotted as a defined color on a timeline.

8. Detection-Based Tracking

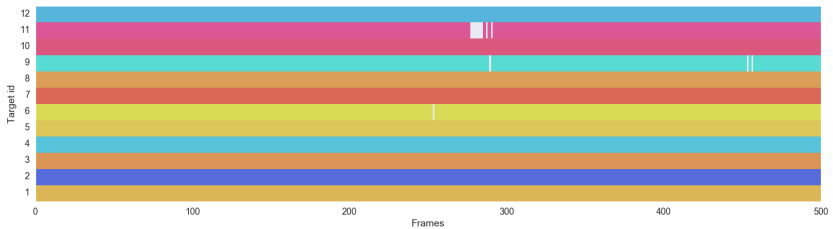
This makes it clear, for example, that all three tracking methods fail in exactly the same situations. Figures 8.15 to 8.17 show such plots for the three tracking sequences.



(a) Bipartite matching on network detections (ISN) for tracking sequence #2.



(b) Global bipartite matching on network detections (ISN) for tracking sequence #2.

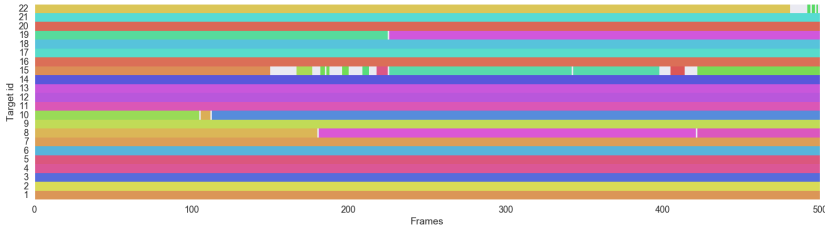


(c) Minimum-cost flow on network detections (ISN) for tracking sequence #2.

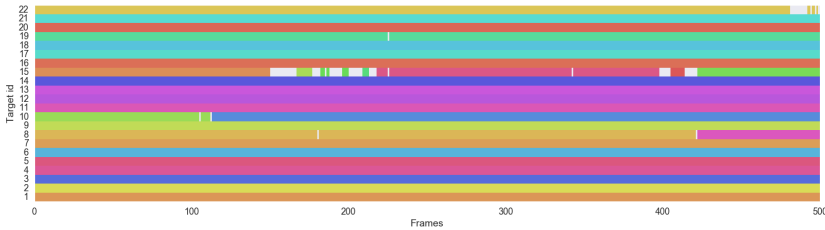
Figure 8.16. Visual representation of tracking results on network detections (ISN) for tracking sequence #2. Assigned target IDs are represented by colors to show continuity or interruptions of each trajectory.

8.3. Evaluation

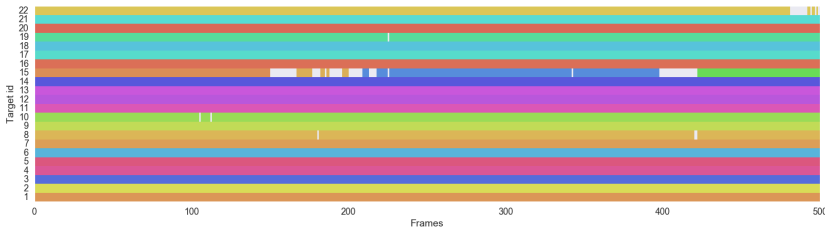
For example, the linking of shorter tracklets from the bipartite matching to longer tracklets by the additional second step in the global bipartite matching described in Section 8.2.3 can be seen clearly (compare Figures 8.15a and 8.15b, Figures 8.16a and 8.16b and Figures 8.17a and 8.17b).



(a) Bipartite matching on network detections (ISN) for tracking sequence #3.



(b) Global bipartite matching on network detections (ISN) for tracking sequence #3.



(c) Minimum-cost flow on network detections (ISN) for tracking sequence #3.

Figure 8.17. Visual representation of tracking results on network detections (ISN) for tracking sequence #3. Assigned target IDs are represented by colors to show continuity or interruptions of each trajectory.

8. Detection-Based Tracking

Transition Costs

In Section 8.2.1, it was described that the transition costs for all detections were generated using a classification method (Gradient Boosted Decision Tree). To justify the choice of this method, Table 8.13 lists the accuracy of the assignments on the ground truth data of the three tracking sequences. As described, the classification is based solely on the ellipse parameters and derived geometric and temporal differences and does not include any information about the appearance of the targets. The classifier was trained on 40% of the available data and with a maximum time-gap of 5:

Table 8.13. Evaluation of correct target assignments with the used transition cost generation. The accuracy is calculated on the ground-truth data for different time-gaps on all three tracking sequences.

Sequence / time-gap [frames]	1	2	3	4	5
#1	0.9963	0.9794	0.9609	0.9422	0.9224
#2	0.9997	0.9972	0.9920	0.9842	0.9796
#3	1.0000	0.9999	0.9995	0.9967	0.9920

8.3.3 Runtime

The detection and tracking methods presented here take different approaches and therefore vary in complexity. The comparison of the runtime is therefore only for completeness because the focus of the elaboration is on the quality of the detections and found trajectories (see Sections 8.3.1 and 8.3.2).

Runtime Detection

The runtime of the network execution is already highly optimized by using appropriate hardware (e.g. GPUs). In addition, only the last network layer is changed in each of the different experiments due to the modular structure of the presented framework, so that only post-processing remains as the one variable part of the runtime. Depending on the experiment, the post-processing consists of the extraction of the ellipses and, if necessary, the clustering. In order to be able to assess the share of the respective parts in the total runtime, Table 8.14 lists the average runtimes per image for the

entire test set of data set #2. The evaluation was performed on a desktop PC with an Intel i7-6700K CPU @ 4.00GHz CPU and a NVIDIA GeForce GTX TITAN X GPU.

Table 8.14. Runtime evaluation of the proposed experiments for categorical segmentation and combined instance segmentation: All values are given as mean runtime and standard deviation over the 226 images of the test set of data set #2.

Center Seg.	Network [ms]	Post [ms]	
ResNet34	32.47 ± 0.31	2.47 ± 0.22	
Incep.-RN-v2	88.91 ± 0.23		
Instance Seg.	Network [ms]	Post (incl. Clustering) [ms]	Clustering [ms]
ResNet34	41.98 ± 1.12	2115.73 ± 448.26	2068.53 ± 443.72
Incep.-RN-v2	101.39 ± 2.41		

The center segmentation based on the slimmer ResNet34 backbone is with approx. 35 ms able to run in real-time (24 fps). With the much more complex Inception-ResNet-v2 backbone, still more than 10 frames per second can be processed, which is one of the common recording rates for surveillance video. The instance segmentation requires a few milliseconds more for the network evaluation due to the two network heads (pixel embedding and binary segmentation), but the subsequent clustering is so complex that this method with approx. 2 seconds per frame cannot be used for an evaluation of classic surveillance videos in real time.

Runtime Tracking

As noted in the previous chapter, not all of the detection techniques presented are capable of producing output in real time. Since the minimum-cost flow tracking method is an offline method anyway, which requires the detections for all frames of the sequence in advance for the global processing of the tracking problem, the runtimes of the different tracking methods are evaluated regardless of their real-time capability. The following evaluation only serves as a rough estimation of runtimes, since the implementations were not specially optimized with respect to runtime. For a fair comparison, only the processing of the methods is measured and listed in Table 8.15 and not the read and write operations for storing

8. Detection-Based Tracking

the results in the database.

Table 8.15. Comparison of the runtime of the three methods shown on the three tracking sequences. For sequence #3 with twice as many animals in the image, the runtime of all methods increases accordingly.

Sequence	Method	Per frame [s]	Post-processing [s]	Total [s]
#1	Bipartite matching	0.0507 ± 0.0029	-	25.3133
#1	Global bipartite match.	0.0535 ± 0.0026	0.0482	26.7507
#1	Minimum-cost flow	0.1245 ± 0.0143	0.8864	63.1861
#2	Bipartite matching	0.0519 ± 0.0018	-	25.9023
#2	Global bipartite match.	0.0523 ± 0.0018	0.0025	26.1218
#2	Minimum-cost flow	0.1165 ± 0.0094	0.7839	59.0485
#3	Bipartite matching	0.0702 ± 0.0035	-	35.0266
#3	Global bipartite match.	0.0707 ± 0.0034	0.0481	35.3238
#3	Minimum-cost flow	0.2214 ± 0.0175	1.2073	111.8948

Conclusion

The goal of the approaches presented here is to support research in the field of agricultural sciences or veterinary medicine and to further automate the processes involved in behavioral studies of farm animals. As stated in the introduction (see Chapter 1), conventional livestock farming, like agriculture in general, will continue to digitalize, increasing the need for automated processes. The installation of cameras and the use of computer vision to build on this is certainly promising. The applications presented in the second part of this thesis (see Chapter 6) show the different approaches where images from surveillance cameras make simple and non-invasive monitoring possible. However, they also introduce the problems that arise in productive operations, such as the complex lighting situation and contamination and occlusion of the recording systems.

An important decision in defining the task is the model used to represent the data. The ellipse model (see Section 6.4) used in this work to represent the pigs has proven to be a reasonable choice, since it on the one hand reproduces the bodies of the pigs very well (in contrast to the otherwise common bounding boxes), and on the other hand allows the annotation of training data with reasonable effort. Even though the ellipses come close to pixel-precise marking of the animals, they do introduce a systematic error, since the static ellipses naturally do not represent possible bending of the animals. Thus, erroneous information occurs in the process of annotation, where ellipses set by hand will inevitably not only cover the actual animal, but also parts of the background. Thus, these pixels get the label *pig* and provide ambiguous information to the neural networks. Figure 8.9 shows well that the neural networks are hardly affected by such noise in the data and do not necessarily produce purely elliptical segmentations if the processed image implies other information. It remains

9. Conclusion

to be considered, however, that the evaluation is performed using this not entirely correct ground truth and the networks would have to specifically reproduce the wrong data to achieve 100% accuracy.

Based on the theoretical principles in the first part of this thesis, different methods for detecting and tracking the pigs were presented in the second part. The different approaches all have their advantages and disadvantages. *Detection-free tracking* (Chapter 7) would theoretically allow an online processing of the data and could even do without any training data if the segmentation is based on the color values. Since the animal model is not based on any color information and has geometric constraints alone, the method quickly reaches its limits if the segmentation is inaccurate. However, if the segmentation reaches a high accuracy (see Table 7.1), the results come close to the upper bound determined with the ground truth segmentation. Due to the fixed number of targets to be tracked (each animal is assigned a separate tracker), the detection rate is also very good, since even the slightest hints of a hidden animal are used. In *detection-based tracking* (Chapter 8), detections are generated separately and subsequently combined into the global best-fit trajectories. Since the detections are generated independently of the tracking, no global prior knowledge is incorporated in this process and individual animals are not detected in every case, as indicated by the slightly lower detection accuracies (compare Tables 7.1, 8.2 and 8.4). However, the subsequent tracking can then produce a much better result by the simultaneous assignment of all targets between two images or even the global optimization over the whole sequence, and thus compensate for the slightly lower detection accuracy. With a MOTA value of consistently around 0.99, the tracking results on the sequences used for evaluation are almost perfect. The lower MOTP values indicate problems in the accuracy of the detections, but as shown in the various experiments in the ablation studies (Section 8.3.1), the detection accuracy could not be further increased even by more complex network architectures. Also, the approach using synthetic images to augment the training data was not successful, which is somewhat disappointing.

In any case, the main finding throughout this work is the superiority of learning-based methods for detection or segmentation. As described in Chapter 3, the hand-selected features of classical detection methods are not able to approach the accuracy of neural networks. Especially in

the case of high variance in the representation of the searched objects, e.g. due to dirt and noise in the images, it would be difficult to define suitable features that are insensitive to these factors. Since the main goal of the work was robust detection, the methods based on neural networks are definitely the right choice. In order to improve the results even further, it would therefore make sense to also process the *data association* of the tracking with the help of learned methods. The neural network approaches presented in this thesis still follow the classical approach in which the detection and tracking processes run separately (detection-based tracking). But just like the combined approach presented in Section 2.2 (detection-free tracking), current research is again moving in the direction of solving the issue in a more unified way by performing the two processes together and simultaneously. In the so-called tracking by *re-detection* or *re-identification*, the outputs of the detector are used to simultaneously describe the individual targets and, if necessary, to recognize them again after a short-term occlusion [BML19; WZL+20; ZWW+20; LXY+20]. It is undisputed that both processes could benefit from the knowledge of the other, because based on the previous trajectory of an object, its position can be estimated very accurately and thus the detection can be improved. In turn, tracking also benefits from better detections. Just as human perception can target attention to specific areas of the scene when tracking objects, neural networks are increasingly equipped with attention modules [SJZ+20; WZW+21; CYZ+21]. Thus, during detection, the neural network can access images in the past (and in the case of offline tracking, also in the future) and extract information.

And just as in the other areas, in the future hopefully also in this research area, larger curated data collection will be created. Because only with enough data to allow researchers to test new approaches and compare their results new breakthroughs can be achieved.

In summary, however, the research question posed at the beginning can be answered with confidence. Already the approaches to robust tracking presented here provide very good results. And with the current development of approaches based on neural networks, the methods will become more and more accurate and (just like in other computer vision tasks) eventually outperform humans.

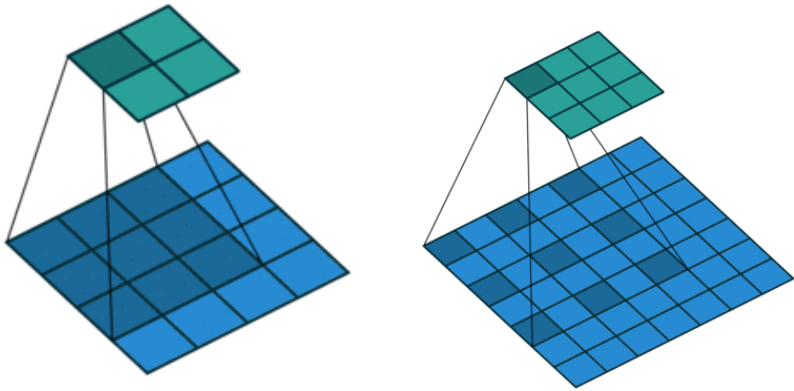
Deep Neural Networks

There are already some excellent books on the functioning and structure of deep neuronal networks such as [Nie15; GBC16; SAV20]. Therefore, only a rough overview is given here to make the methods described in Sections 3.2 and 8.1 comprehensible.

As defined in [GBC16], the goal of a neural network is to approximate an arbitrary function f^* . The shape of the function is learned based on the available training data, which contains the input data x and the corresponding desired output data (or label) y as *ground truth*. The searched function would assign the correct label to each input example $y = f^*(x)$, and neural networks define a mapping $y = f(x, \theta)$ accordingly, such that the contained parameters θ can be learned in such a way that the approximated function f represents the real function f^* as closely as possible. This is accomplished by combining individual linear modules (or layers) with a nonlinear activation function, and from these modules f^n the function f can then be formed as a chain of functions $f(x) = f^n(f^{n-1}(f^{n-2}(\dots f^1(x))))$ so that the output of the previous module is used as the input of the next module.

To adjust the initial randomly initialized parameters θ of the network, the input data x is fed through the network and the output \hat{y} obtained at the output layer is compared to the existing label y (ground truth). Using an error (often also called *loss*) function chosen on a per-problem basis, the error is calculated with respect to each parameter and the parameters are then adjusted accordingly to reduce the error and thus match the output of the network to the ground truth data. On the basis of the gradient (i.e. the degree of change of the error with a change of the parameters by one unit at a time), the entire set of parameters can be adjusted in each such training step [SAV20]. However, in order to adapt the network to all data

A. Deep Neural Networks



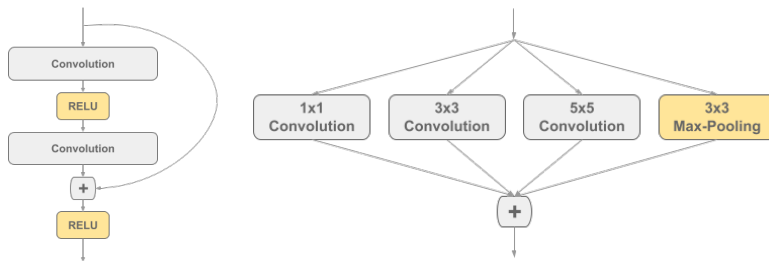
(a) Classical convolution with kernel-size 3×3 , (no stride)

(b) Dilated convolution with kernel-size 3×3 , (no stride)

Figure A.1. Visualization of the classical and the dilated convolution with highlighted input pixels (blue) which were taken into account for the calculation of the one output pixel (cyan). [DV16]

equally, the gradient updates are calculated and averaged for all input data. In order to optimally detect the direction of the global minimum in this gradient descent procedure, the update is often constrained by an additional parameter called *learning rate*.

The calculation of the gradient update does not necessarily have to be based on all input data, but the data can also be divided into randomly arranged batches, so that the update can be performed after each such batch (called *stochastic gradient descent*). In addition, the update rule can also be more granularly related to the individual parameters, so that separate learning rates or additional parameters such as momentum are included. Thus, in addition to the classic *stochastic gradient descent*, there are several established optimization strategies [GBC16]. In Section 3.2.1, *convolutional neural networks* (CNN) have already been introduced, which are especially suited for processing images as input data. With their two-dimensional weight matrices (*kernels*), local features of the input data are captured and processed in subsequent layers (see Figure A.1). Parameters such



(a) ResNet-Block with two convolutional layers and the residual connection. (b) Inception module with different parallel convolutions.

Figure A.2. Visualization of high-level network modules, to easily train very deep architectures or use different receptive fields. Depictions adopted from [HZR+16; SLJ+15]

as the size of the kernel matrix, the sampling step size (*stride*), or the sparse distribution of the calculation (*dilated convolutions*) result in many hyperparameters for adapting the network to the task [DV16]. The sheer amount of hyperparameters, would allow almost infinite architectures. So instead of composing network architectures from individual layers, building blocks have gradually been developed that combine multiple layers and can then also be combined in a modular fashion to form new architectures. Examples are the *residual blocks* in the ResNet [HZR+16] architecture or the *inception modules* that apply differently parameterized convolutions in parallel [SLJ+15] (see Figure A.2). Similarly, it is possible to think of the architecture as part of the optimization problem and optimize it simultaneously [ZVS+18].

Covariance Matrix Adaptation - Evolution Strategy

To find the global maximum in the optimization problem for the detection free tracking (Chapter 7) the Covariance Matrix Adaptation - Evolution Strategy (CMA-ES) [HO01] was used, as it has shown great performance¹ in the domain of randomized black-box search techniques. Randomized black-box optimizer use an internally maintained probability distribution to randomly choose points where the given fitness-function is to be evaluated. The gathered fitness-function values are then used to update all the distribution-parameters trying to maximize the probability of finding the best solution (see Algorithm 1). In the case of CMA-ES a multivariate normal distribution $\mathcal{N}(m, C)$ is used, where $m \in \mathbb{R}^l$ is the mean of the parameter space and $C \in \mathbb{R}^{l \times l}$ the covariance matrix of the parameter space respectively. C is usually initialized as a diagonal matrix filled with the variances of the individual parameters.

Algorithm 1: Randomized black-box search

```

1 Initialize distribution;
2 for generation  $0, 1, 2, \dots$  do
3   | Sample  $\lambda$  independent vectors of parameters  $\vec{\theta}$  from
   |   distribution;
4   | Evaluate the samples on fitness-function  $f$ ;
5   | Update distribution parameters;
6   | if termination criterion met then
7   |   | break;

```

¹see 2009 Black-Box Optimization Benchmarking Competition (BBOB)

B. Covariance Matrix Adaptation - Evolution Strategy

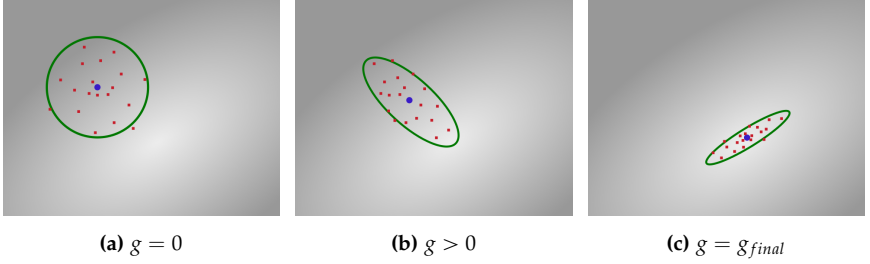


Figure B.1. Schematic visualization of the CMA-ES optimization process evolving over generation g with population-size $\lambda = 19$ and $l = 2$ parameters. The background shows the fitness function value (brighter = higher). Also shown are the covariance matrix C (green ellipse), the mean m (blue dot) and the sample points (population) according to C and m (red points). [BTK15]

Since CMA-ES follows an evolutionary strategy the $\lambda \in \mathbb{N}$ sampled points $\vec{\theta}_{i=0, \dots, \lambda-1}$ can be interpreted as a population. Starting with the initial distribution $\mathcal{N}(m^{(0)}, C^{(0)})$ in each iteration CMA-ES regenerates its population and forms a new generation $\vec{\theta}_{i=0, \dots, \lambda-1}^{(g+1)}$ by sampling from an updated distribution $\mathcal{N}(m^{(g+1)}, C^{(g+1)})$, with $g \in \mathbb{N}$ depicting the generation number. Thus the evolution between g and $g + 1$ is done by updating the mean and the covariance-matrix. To move the mean the best $\mu \in \mathbb{N}_{\leq \lambda}$ samples of the current population are selected. Assuming the samples are sorted according to their fitness $f(\vec{\theta}_0) \geq f(\vec{\theta}_1) \geq \dots \geq f(\vec{\theta}_{\lambda-1})$ the new mean is calculated by applying a weighted average with weights $w_{i=0, \dots, \lambda-1} \in [0, 1]$ and $\sum_{i=0}^{\lambda-1} w_i = 1$. To enforce the selection the last $\lambda - \mu$ weights are set to 0.

$$m^{(g+1)} = \sum_{i=0}^{\mu-1} w_i \vec{\theta}_i^{(g)} \quad (\text{B.0.1})$$

To maximize the probability of finding the best parameter-vector the covariance-matrix is updated:

$$C^{(g+1)} = \frac{1}{\mu} \sum_{i=0}^{\mu-1} (\vec{\theta}_i^{(g)} - m^{(g)}) (\vec{\theta}_i^{(g)} - m^{(g)})^T. \quad (\text{B.0.2})$$

This effects the probability density function of the distribution and there-

fore the spread of the samples in the search-space (see Figure B.1). For further optimized methods of the covariance-update we refer to the written tutorial of Nicolaus Hansen². As long as no termination-criterion is met, the fitness-function is evaluated in each iteration for each individual of the population. CMA-ES defines different criteria for checking for convergence or runtime-requirements. The two key termination-criteria are a stagnation in the history of the last fitness-function values and the maximum number of iterations (or generations of the population). With a reasonable value for stagnation-limitation set, the maximum number of iterations is rarely reached and can therefore be kept untouched. The remaining control-parameter is the population-size λ . The significant influence of the population-size on the global search performance is evaluated and confirmed by [HK04]. If the size of the population is too low, more iterations are needed to achieve a significant improvement of the distribution parameters. If it is too high, the count of fitness-function evaluations increases and therefore the runtime. [Han09] proposed the optimal population-size λ for l parameters as $\lambda = 4 + \lfloor 3 \ln l \rfloor$.

²<https://www.lri.fr/~hansen/cmatutorial.pdf>

Bibliography

- [ADF12] B. Alexe, T. Deselaers, and V. Ferrari. “Measuring the Objectness of Image Windows”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (Nov. 2012), pp. 2189–2202. ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2012.28.
- [AGK11] Peter Ahrendt, Torben Gregersen, and Henrik Karstoft. “Development of a real-time computer vision system for tracking loose-housed pigs”. In: *Computers and Electronics in Agriculture* 76.2 (2011), pp. 169–174.
- [ANB+04] IL Andersen, E Nævdal, M Bakken, and KE Bøe. “Aggression and group size in domesticated pigs, *sus scrofa*: ‘when the winner takes it all and the loser is standing small’”. In: *Animal behaviour* 68.4 (2004), pp. 965–975.
- [BB05] Kristin Branson and Serge Belongie. “Tracking multiple mouse contours (without too many samples)”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 1. IEEE. 2005, pp. 1039–1046.
- [BC08] François Bardet and Thierry Chateau. “Mcmc particle filter for real-time visual tracking of vehicles”. In: *2008 11th International IEEE Conference on Intelligent Transportation Systems*. IEEE. 2008, pp. 539–544.
- [BDK+19] J Brünger, S Dippel, Reinhard Koch, and C Veit. “Tailception: using neural networks for assessing tail lesions on pictures of pig carcasses”. In: *animal* 13.5 (2019), pp. 1030–1036.
- [Ben12] Yoshua Bengio. “Deep learning of representations for unsupervised and transfer learning”. In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. 2012, pp. 17–36.

Bibliography

- [BES06] Keni Bernardin, Alexander Elbs, and Rainer Stiefelhagen. "Multiple object tracking performance metrics and evaluation in a smart room environment". In: *Sixth IEEE International Workshop on Visual Surveillance, in conjunction with ECCV*. Vol. 90. 91. Citeseer. 2006.
- [BFT+11] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. "Multiple object tracking using k-shortest paths optimization". In: *IEEE transactions on pattern analysis and machine intelligence* 33.9 (2011), pp. 1806–1819.
- [BGT+20] Johannes Brünger, Maria Gentz, Imke Traulsen, and Reinhard Koch. "Panoptic segmentation of individual pigs for posture recognition". In: *Sensors* 20.13 (2020), p. 3710.
- [Bha43] Anil Bhattacharyya. "On a measure of divergence between two statistical populations defined by their probability distributions". In: *Bull. Calcutta Math. Soc.* 35 (1943), pp. 99–109.
- [Bis06] Christopher M. Bishop. *Pattern recognition and machine learning*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN: 0387310738.
- [BK69] Brent Berlin and Paul Kay. *Basic color terms: their universality and evolution*. California UP, 1969.
- [BKC17] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "Segnet: a deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495.
- [BL02] Matthew Brown and David G Lowe. "Invariant features from interest point groups." In: *BMVC*. Vol. 4. 2002.
- [BL71] Francois Bourgeois and Jean-Claude Lassalle. "An extension of the munkres algorithm for the assignment problem to rectangular matrices". In: *Communications of the ACM* 14.12 (1971), pp. 802–804.

- [BML19] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. “Tracking without bells and whistles”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 941–951.
- [Bra00] G. Bradski. “The OpenCV Library”. In: *Dr. Dobb’s Journal of Software Tools* (2000).
- [BRL+09] Michael D Breitenstein, Fabian Reichlin, Bastian Leibe, Esther Koller-Meier, and Luc Van Gool. “Robust tracking-by-detection using a detector confidence particle filter”. In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 1515–1522.
- [BTK15] Johannes Brünger, Imke Traulsen, and Reinhard Koch. “Randomized global optimization for robust pose estimation of multiple targets in image sequences”. In: *Math. Model. Comput. Methods 2* (2015), pp. 45–53.
- [BTK18] Johannes Brünger, Imke Traulsen, and Reinhard Koch. “Model-based detection of pigs in images under sub-optimal conditions”. In: *Computers and electronics in agriculture* 152 (2018), pp. 59–63.
- [BTV06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “Surf: speeded up robust features”. In: *European conference on computer vision*. Springer, 2006, pp. 404–417.
- [BVK20] Lara Blömke, Nina Volkmann, and Nicole Kemper. “Evaluation of an automated assessment system for ear and tail lesions as animal welfare indicators in pigs at slaughter”. In: *Meat science* 159 (2020), p. 107934.
- [CFL06] Yizheng Cai, Nando de Freitas, and James J Little. “Robust visual tracking for multiple targets”. In: *European conference on computer vision*. Springer, 2006, pp. 107–118.
- [Cho15] Wongun Choi. “Near-online multi-target tracking with aggregated local flow descriptor”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 3029–3037.

Bibliography

- [CLG+08] Ö Cangar, Toon Leroy, Marcella Guarino, Erik Vranken, Richard Fallon, J Lenehan, John Mee, and Daniel Berckmans. “Automatic real-time monitoring of locomotion and posture behaviour of pregnant cows prior to calving using online image analysis”. In: *Computers and electronics in agriculture* 64.1 (2008), pp. 53–60.
- [CMS13] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. “Density-based clustering based on hierarchical density estimates”. In: *Pacific-Asia conference on knowledge discovery and data mining*. Springer. 2013, pp. 160–172.
- [CPK+14] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. “Semantic image segmentation with deep convolutional nets and fully connected crfs”. In: *arXiv preprint arXiv:1412.7062* (2014).
- [CPK+18] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. “Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2018), pp. 834–848.
- [CPS+17] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. “Rethinking atrous convolution for semantic image segmentation”. In: *arXiv preprint arXiv:1706.05587* (2017).
- [CYZ+21] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. “Transformer tracking”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8126–8135.
- [CZP+18] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. “Encoder-decoder with atrous separable convolution for semantic image segmentation”. In: *ECCV*. 2018.

- [DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: a large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.
- [DFB+99] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. "Monte carlo localization for mobile robots". In: *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*. Vol. 2. IEEE. 1999, pp. 1322–1328.
- [DHL+16] Jifeng Dai, Kaiming He, Yi Li, Shaoqing Ren, and Jian Sun. "Instance-sensitive fully convolutional networks". In: *European Conference on Computer Vision*. Springer. 2016, pp. 534–549.
- [DLH+16] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. "R-fcn: object detection via region-based fully convolutional networks". In: *Advances in neural information processing systems*. 2016, pp. 379–387.
- [DNV17] Bert De Brabandere, Davy Neven, and Luc Van Gool. "Semantic instance segmentation with a discriminative loss function". In: *arXiv preprint arXiv:1708.02551* (2017).
- [DSF+14] Martin Danelljan, Fahad Shahbaz Khan, Michael Felsberg, and Joost Van de Weijer. "Adaptive color attributes for real-time visual tracking". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1090–1097.
- [DT05] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 886–893.
- [DV16] Vincent Dumoulin and Francesco Visin. "A guide to convolution arithmetic for deep learning". In: *ArXiv e-prints* (Mar. 2016). eprint: 1603.07285.

Bibliography

- [EEV+15] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. “The pascal visual object classes challenge: a retrospective”. In: *International journal of computer vision* 111.1 (2015), pp. 98–136.
- [EST+14] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. “Scalable object detection using deep neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 2147–2154.
- [FDG+14] Luca Fiaschi, Ferran Diego, Konstantin Gregor, Martin Schiegg, Ullrich Koethe, Marta Zlatic, and Fred A Hamprecht. “Tracking indistinguishable translucent objects over time using weakly supervised structured learning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 2736–2743.
- [FF+96] Andrew W Fitzgibbon, Robert B Fisher, et al. *A buyer’s guide to conic fitting*. University of Edinburgh, Department of Artificial Intelligence, 1996.
- [FGM+09] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. “Object detection with discriminatively trained part-based models”. In: *IEEE transactions on pattern analysis and machine intelligence* 32.9 (2009), pp. 1627–1645.
- [FH04] Pedro F Felzenszwalb and Daniel P Huttenlocher. “Efficient graph-based image segmentation”. In: *International journal of computer vision* 59.2 (2004), pp. 167–181.
- [FH05] Pedro F Felzenszwalb and Daniel P Huttenlocher. “Pictorial structures for object recognition”. In: *International journal of computer vision* 61.1 (2005), pp. 55–79.
- [FLB11] Rana Farah, JM Pierre Langlois, and Guillaume-Alexandre Bilodeau. “Rat: robust animal tracking”. In: *Robotic and Sensors Environments (ROSE), 2011 IEEE International Symposium on*. IEEE. 2011, pp. 65–70.

- [FPZ03] Robert Fergus, Pietro Perona, and Andrew Zisserman. "Object class recognition by unsupervised scale-invariant learning". In: *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings*. Vol. 2. IEEE. 2003, pp. II–II.
- [Fri01] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001), pp. 1189–1232.
- [FSB+97] AR Frost, CP Schofield, SA Beaulah, TT Mottram, JA Lines, and CM Wathes. "A review of livestock monitoring and the need for integrated systems". In: *Computers and electronics in agriculture* 17.2 (1997), pp. 139–159.
- [FYT+09] Toshiyuki Fujii, Hiroshi Yokoi, Tatsuya Tada, Kotaro Suzuki, and Kenji Tsukamoto. "Poultry tracking system with camera using particle filters". In: *2008 IEEE International Conference on Robotics and Biomimetics*. IEEE. 2009, pp. 1888–1893.
- [GAN+18] Oleksiy Guzhva, Håkan Ardö, Mikael Nilsson, Anders Herlin, and Linda Tufvesson. "Now you see me: convolutional neural network based tracker for dairy cows". In: *Frontiers in Robotics and AI* 5 (2018), p. 107.
- [GB10] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256.
- [GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [GDD+14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

Bibliography

- [GDF+19] Brian Q Geuther, Sean P Deats, Kai J Fox, Steve A Murray, Robert E Braun, Jacqueline K White, Elissa J Chesler, Cathleen M Lutz, and Vivek Kumar. “Robust mouse tracking in complex environments using neural networks”. In: *Communications biology* 2.1 (2019), pp. 1–11.
- [Gir15] Ross Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.
- [GLA+09] Chunhui Gu, Joseph J Lim, Pablo Arbeláez, and Jitendra Malik. “Recognition using regions”. In: *2009 IEEE Conference on computer vision and pattern recognition*. IEEE. 2009, pp. 1030–1037.
- [GMZ+20] M Gentz, C Meckbach, S Zeidler, V Loges, J Brünger, R Koch, and I Traulsen. “Activity behaviour of pigs: comparison of manual and automated video analyses”. In: *Pig tail biting in different farrowing and rearing systems with a focus on tail lesions, tail losses and activity monitoring* (2020), p. 76.
- [Gre95] Peter J Green. “Reversible jump markov chain monte carlo computation and bayesian model determination”. In: *Biometrika* 82.4 (1995), pp. 711–732.
- [GSH+13] Luca Giancardo, Diego Sona, Huiping Huang, Sara Sannino, Francesca Managò, Diego Scheggia, Francesco Papaleo, and Vittorio Murino. “Automatic visual tracking and social behaviour analysis with multiple mice”. In: *PLoS one* 8.9 (2013), e74557.
- [Han09] Nikolaus Hansen. “Benchmarking a bi-population cma-es on the bbob-2009 function testbed”. In: *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. GECCO '09. Montreal, Quebec, Canada: ACM, 2009, pp. 2389–2396. ISBN: 978-1-60558-505-5. DOI: 10.1145/1570256.1570333.
- [Has70] W Keith Hastings. “Monte carlo sampling methods using markov chains and their applications”. In: (1970).

- [HGD+17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.
- [HHR13] Martin Hofmann, Michael Haag, and Gerhard Rigoll. “Unified hierarchical multi-object tracking using global data association”. In: *2013 IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*. IEEE. 2013, pp. 22–28.
- [HK04] Nikolaus Hansen and Stefan Kern. “Evaluating the cma evolution strategy on multimodal test functions”. English. In: *Parallel Problem Solving from Nature - PPSN VIII*. Vol. 3242. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, pp. 282–291. ISBN: 978-3-540-23092-2. DOI: 10.1007/978-3-540-30217-9_29.
- [HK94] Glenn E Healey and Raghava Kondepudy. “Radiometric ccd camera calibration and noise estimation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16.3 (1994), pp. 267–276.
- [HLV+17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. “Densely connected convolutional networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.
- [HO01] Nikolaus Hansen and Andreas Ostermeier. “Completely derandomized self-adaptation in evolution strategies”. In: *Evol. Comput.* 9.2 (June 2001), pp. 159–195. ISSN: 1063-6560. DOI: 10.1162/106365601750190398.
- [HS+88] Christopher G Harris, Mike Stephens, et al. “A combined corner and edge detector.” In: *Alvey vision conference*. Vol. 15. 50. Citeseer. 1988, pp. 10–5244.
- [HWN08] Chang Huang, Bo Wu, and Ramakant Nevatia. “Robust object tracking by hierarchical association of detection responses”. In: *European Conference on Computer Vision*. Springer. 2008, pp. 788–801.

Bibliography

- [HZC04] Xuming He, Richard S Zemel, and Miguel A Carreira-Perpinán. “Multiscale conditional random fields for image labeling”. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004*. Vol. 2. IEEE. 2004, pp. II–II.
- [HZR+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [IB98] Michael Isard and Andrew Blake. “Condensation - conditional density propagation for visual tracking”. In: *International journal of computer vision* 29.1 (1998), pp. 5–28.
- [JCS+18] Miso Ju, Yunchang Choi, Jihyun Seo, Jaewon Sa, Sungju Lee, Yongwha Chung, and Daihee Park. “A kinect-based segmentation of touching-pigs for real-time monitoring”. In: *Sensors* 18.6 (2018), p. 1746.
- [JWC+20] Alexander B. Jung et al. *Imgaug*. 2020. URL: <https://github.com/aleju/imgaug>.
- [KB01] Timor Kadir and Michael Brady. “Saliency, scale and image description”. In: *International Journal of Computer Vision* 45.2 (2001), pp. 83–105.
- [KB14] Diederik P Kingma and Jimmy Ba. “Adam: a method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [KBD04] Zia Khan, Tucker Balch, and Frank Dellaert. “An mcmc-based particle filter for tracking multiple interacting targets”. In: *European Conference on Computer Vision*. Springer. 2004, pp. 279–290.
- [KBD05] Zia Khan, Tucker Balch, and Frank Dellaert. “Mcmc-based particle filtering for tracking a variable number of interacting targets”. In: *IEEE transactions on pattern analysis and machine intelligence* 27.11 (2005), pp. 1805–1819.

- [KBH+13] Mohammadamin Kashiha, Claudia Bahr, Sara Amirpour Haredasht, Sanne Ott, Christel PH Moons, Theo A Niewold, Frank O Ödberg, and Daniel Berckmans. "The automatic monitoring of pigs water use by cameras". In: *Computers and Electronics in Agriculture* 90 (2013), pp. 164–169.
- [KBO+13] Mohammadamin Kashiha, Claudia Bahr, Sanne Ott, Christel PH Moons, Theo A Niewold, Frank O Ödberg, and Daniel Berckmans. "Automatic identification of marked pigs in a pen using image pattern recognition". In: *Computers and electronics in agriculture* 93 (2013), pp. 111–120.
- [KBO+14] Mohammad Amin Kashiha, Claudia Bahr, Sanne Ott, Christel PH Moons, Theo A Niewold, Frank Tuytens, and Daniel Berckmans. "Automatic monitoring of pig locomotion using image analysis". In: *Livestock Science* 159 (2014), pp. 141–148.
- [KCL17] Wonjun Kim, Yong Beom Cho, and Sangrak Lee. "Thermal sensor-based multiple object tracking for intelligent livestock breeding". In: *IEEE Access* 5 (2017), pp. 27453–27463.
- [KGC18] Alex Kendall, Yarin Gal, and Roberto Cipolla. "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7482–7491.
- [KHG+19] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. "Panoptic segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9404–9413.
- [KKA+20] Steffen Küster, Matthias Kardel, Stefanie Ammer, Johannes Brünger, Reinhard Koch, and Imke Traulsen. "Usage of computer vision analysis for automatic detection of activity changes in sows during final gestation". In: *Computers and Electronics in Agriculture* 169 (2020), p. 105177.
- [KMM12] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. "Tracking-learning-detection". In: *IEEE transactions on pattern analysis and machine intelligence* 34.7 (2012), pp. 1409–1422.

Bibliography

- [KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.
- [Kuh55] Harold W Kuhn. “The hungarian method for the assignment problem”. In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97.
- [LAE+16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. “Ssd: single shot multibox detector”. In: *European conference on computer vision*. Springer. 2016, pp. 21–37.
- [LBB+98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
- [LBH08] Christoph H Lampert, Matthew B Blaschko, and Thomas Hofmann. “Beyond sliding windows: object localization by efficient subwindow search”. In: *2008 IEEE conference on computer vision and pattern recognition*. IEEE. 2008, pp. 1–8.
- [LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.
- [LBO+12] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. “Efficient backprop”. In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.
- [LBZ+17] Tuan Anh Le, Atilim Güneş Baydin, Robert Zinkov, and Frank Wood. “Using synthetic data to train neural networks is model-based reasoning”. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2017, pp. 3514–3521.
- [LDG+17] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Bharath Hariharan, and Serge Belongie. “Feature pyramid networks for object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

- [LHB+99] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. "Object recognition with gradient-based learning". In: *Shape, contour and grouping in computer vision*. Springer, 1999, pp. 319–345.
- [LJP+16] Jonguk Lee, Long Jin, Daihee Park, and Yongwha Chung. "Automatic Recognition of Aggressive Behavior in Pigs Using a Kinect Depth Sensor". en. In: *Sensors* 16.5 (May 2016), p. 631. DOI: 10.3390/s16050631.
- [LLW+17] Xiaodan Liang, Liang Lin, Yunchao Wei, Xiaohui Shen, Jianchao Yang, and Shuicheng Yan. "Proposal-free network for instance-level object segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 40.12 (2017), pp. 2978–2991.
- [LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. "Microsoft coco: common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.
- [Low+99] David G Lowe et al. "Object recognition from local scale-invariant features." In: *iccv*. Vol. 99. 2. 1999, pp. 1150–1157.
- [LQD+17] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. "Fully convolutional instance-aware semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2359–2367.
- [LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.
- [LXM+14] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, Xiaowei Zhao, and Tae-Kyun Kim. "Multiple object tracking: a literature review". In: *arXiv preprint arXiv:1409.7618* (2014).

Bibliography

- [LXY+20] Wei Li, Yuanjun Xiong, Shuo Yang, Siqi Deng, and Wei Xia. “Smot: single-shot multi object tracking”. In: *arXiv preprint arXiv:2010.16031* (2020).
- [MBJ17] Tim Michels, Dimitri Berh, and Xiaoyi Jiang. “An rjmc-based method for tracking and resolving collisions of drosophila larvae”. In: *IEEE/ACM transactions on computational biology and bioinformatics* 16.2 (2017), pp. 465–474.
- [MCP+05] Erikson F Morais, Mario Fernando Montenegro Campos, Flavio LC Padua, and Rodrigo L Carceroni. “Particle filter-based predictive tracking for robust fish counting”. In: *XVIII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAP'05)*. IEEE. 2005, pp. 367–374.
- [MHA17] Leland McInnes, John Healy, and Steve Astels. “Hdbscan: hierarchical density based clustering”. In: *The Journal of Open Source Software* 2.11 (Mar. 2017). DOI: 10.21105/joss.00205. URL: <https://doi.org/10.21105%2Fjoss.00205>.
- [MIF+18] Nikolaus Mayer, Eddy Ilg, Philipp Fischer, Caner Hazirbas, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. “What makes good synthetic training data for learning disparity and optical flow estimation?”. In: *International Journal of Computer Vision* 126.9 (2018), pp. 942–960.
- [MLR+16] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. “Mot16: a benchmark for multi-object tracking”. In: *arXiv preprint arXiv:1603.00831* (2016).
- [MMC+16] Stephen G. Matthews, Amy L. Miller, James Clapp, Thomas Plötz, and Ilias Kyriazakis. “Early detection of health and welfare compromises through automated detection of behavioural changes in pigs”. en. In: *The Veterinary Journal* 217 (Nov. 2016), pp. 43–51. ISSN: 10900233. DOI: 10.1016/j.tvjl.2016.09.005.
- [MMP+17] Stephen G Matthews, Amy L Miller, Thomas Plötz, and Ilias Kyriazakis. “Automated tracking to measure behavioural changes in pigs for health and welfare monitoring”. In: *Scientific reports* 7.1 (2017), p. 17582.

- [MPC+18] Mateusz Mittek, Eric T. Psota, Jay D. Carlson, Lance C. Pérez, Ty Schmidt, and Benny Mote. “Tracking of group-housed pigs using multi-ellipsoid expectation maximisation”. In: *IET Computer Vision* 12.2 (2018), pp. 121–128. ISSN: 1751-9640. DOI: 10.1049/iet-cvi.2017.0085.
- [MPP+16] Mateusz Mittek, Eric T Psota, Lance C Pérez, Ty Schmidt, and Benny Mote. “Health monitoring of group-housed pigs using depth-enabled multi-object tracking”. In: *Proceedings of Int Conf Pattern Recognit, Workshop on Visual observation and analysis of Vertebrate And Insect Behavior*. 2016.
- [MRS13] Anton Milan, Stefan Roth, and Konrad Schindler. “Continuous energy minimization for multitarget tracking”. In: *IEEE transactions on pattern analysis and machine intelligence* 36.1 (2013), pp. 58–72.
- [MS95] Nigel JB McFarlane and C Paddy Schofield. “Segmentation and tracking of piglets in images”. In: *Machine vision and applications* 8.3 (1995), pp. 187–193.
- [NAÅ+14] Mikael Nilsson, Håkan Ardö, Kalle Åström, Anders Herlin, Christer Bergsten, and Oleksiy Guzhva. “Learning based image segmentation of pigs in a pen”. In: *Visual observation and analysis of vertebrate and insect behavior—Workshop at the 22nd International Conference on Pattern Recognition (ICPR 2014), Stockholm, Sweden, August*. 2014, pp. 24–28.
- [NEM+17] Abozar Nasirahmadi, Sandra A. Edwards, Stephanie M. Matheson, and Barbara Sturm. “Using automated image analysis in pig behavioural research: Assessment of the influence of enrichment substrate provision on lying behaviour”. en. In: *Applied Animal Behaviour Science* 196 (Nov. 2017), pp. 30–35. ISSN: 01681591. DOI: 10.1016/j.applanim.2017.06.015.
- [NES17] Abozar Nasirahmadi, Sandra A Edwards, and Barbara Sturm. “Implementation of machine vision for detecting behaviour of cattle and pigs”. In: *Livestock Science* 202 (2017), pp. 25–38.

Bibliography

- [NHE+16a] Abozar Nasirahmadi, Oliver Hensel, Sandra A. Edwards, and Barbara Sturm. "Automatic detection of mounting behaviours among pigs using image analysis". en. In: *Computers and Electronics in Agriculture* 124 (June 2016), pp. 295–302. ISSN: 01681699. DOI: 10.1016/j.compag.2016.04.022.
- [NHE+16b] Abozar Nasirahmadi, Oliver Hensel, Sandra A Edwards, and Barbara Sturm. "Automatic detection of mounting behaviours among pigs using image analysis". In: *Computers and Electronics in Agriculture* 124 (2016), pp. 295–302.
- [Nie15] Michael A Nielsen. *Neural networks and deep learning*. <http://neuralnetworksanddeeplearning.com>. Determination Press, 2015.
- [NRH+15a] Abozar Nasirahmadi, Uwe Richter, Oliver Hensel, Sandra Edwards, and Barbara Sturm. "Using machine vision for investigation of changes in pig group lying patterns". en. In: *Computers and Electronics in Agriculture* 119 (Nov. 2015), pp. 184–190. ISSN: 01681699. DOI: 10.1016/j.compag.2015.10.023.
- [NRH+15b] Abozar Nasirahmadi, Uwe Richter, Oliver Hensel, Sandra Edwards, and Barbara Sturm. "Using machine vision for investigation of changes in pig group lying patterns". In: *Computers and Electronics in Agriculture* 119 (2015), pp. 184–190.
- [NSE+19] Abozar Nasirahmadi, Barbara Sturm, Sandra Edwards, Knut Håkan Jeppsson, Anne-Charlotte Olsson, Simone Müller, and Oliver Hensel. "Deep learning and machine vision approaches for posture detection of individual pigs". In: *Sensors* 19.17 (2019), p. 3738.
- [OFL07] Mustafa Ozuysal, Pascal Fua, and Vincent Lepetit. "Fast keypoint recognition in ten lines of code". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. Ieee. 2007, pp. 1–8.
- [OMK+14] S. Ott, C.P.H. Moons, M.A. Kashiha, C. Bahr, F.A.M. Tuytens, D. Berckmans, and T.A. Niewold. "Automated video analysis of pig activity at pen level highly correlates to human observations of behavioural activities". en. In: *Livestock*

Science 160 (Feb. 2014), pp. 132–137. ISSN: 18711413. DOI: 10.1016/j.livsci.2013.12.011.

- [OPH94] Timo Ojala, Matti Pietikainen, and David Harwood. “Performance evaluation of texture measures with classification based on kullback discrimination of distributions”. In: *Proceedings of 12th International Conference on Pattern Recognition*. Vol. 1. IEEE. 1994, pp. 582–585.
- [Orl97] James B Orlin. “A polynomial time primal network simplex algorithm for minimum cost flows”. In: *Mathematical Programming* 78.2 (1997), pp. 109–129.
- [OTD+04] Kenji Okuma, Ali Taleghani, Nando De Freitas, James J Little, and David G Lowe. “A boosted particle filter: multitarget detection and tracking”. In: *European conference on computer vision*. Springer. 2004, pp. 28–39.
- [Ots79] Nobuyuki Otsu. “A threshold selection method from gray-level histograms”. In: *IEEE transactions on systems, man, and cybernetics* 9.1 (1979), pp. 62–66.
- [PC14] Fabio Poiesi and Andrea Cavallaro. “Tracking multiple high-density homogeneous targets”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 25.4 (2014), pp. 623–637.
- [PCD15] Pedro O Pinheiro, Ronan Collobert, and Piotr Dollár. “Learning to segment object candidates”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 1990–1998.
- [PES+09] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. “You’ll never walk alone: modeling social behavior for multi-target tracking”. In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE. 2009, pp. 261–268.
- [PLC+16] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. “Learning to refine object segments”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 75–91.

Bibliography

- [PMP+19] Eric Psota, Mateusz Mittek, Lance Pérez, Ty Schmidt, and Benny Mote. “Multi-Pig Part Detection and Association with a Fully-Convolutional Network”. en. In: *Sensors* 19.4 (Feb. 2019), p. 852. ISSN: 1424-8220. DOI: 10.3390/s19040852.
- [POM+10] Hemerson Pistori, Valguima Victoria Viana Aguiar Odakura, João Bosco Oliveira Monteiro, Wesley Nunes Gonçalves, Antonia Railda Roel, Jonathan de Andrade Silva, and Bruno Brandoli Machado. “Mice and larvae tracking using a particle filter with an auto-adjustable observation model”. In: *Pattern Recognition Letters* 31.4 (2010), pp. 337–346.
- [Pow07] David MW Powers. “Evaluation: from precision, recall and f-factor to roc, informedness, markedness & correlation”. In: (2007).
- [RBO+17] Benjamin Risse, Dimitri Berh, Nils Otto, Christian Klämbt, and Xiaoyi Jiang. “Fimtrack: an open source tracking and locomotion analysis software for small animals”. In: *PLoS computational biology* 13.5 (2017), e1005530.
- [RDG+16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. “You only look once: unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.
- [RDS+15] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.
- [RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.
- [RHG+15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. “Faster r-cnn: towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems*. 2015, pp. 91–99.

- [RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.
- [RMN09] Rajat Raina, Anand Madhavan, and Andrew Y Ng. "Large-scale deep unsupervised learning using graphics processors". In: *Proceedings of the 26th annual international conference on machine learning*. 2009, pp. 873–880.
- [RT16] Bernardino Romera-Paredes and Philip Hilaire Sean Torr. "Recurrent instance segmentation". In: *European conference on computer vision*. Springer. 2016, pp. 312–329.
- [Rud16] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).
- [RZ17] Mengye Ren and Richard S Zemel. "End-to-end instance segmentation with recurrent attention". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6656–6664.
- [SAT+17] Pudith Sirigrivatanawong, Shogo Arai, Vladimiro Thoma, and Koichi Hashimoto. "Multiple drosophila tracking system with heading direction". In: *Sensors* 17.1 (2017), p. 96.
- [SAV+12] Hans AM Spoolder, André AJ Aarnink, Herman M Vermeer, Johan van Riel, and Sandra A Edwards. "Effect of increasing temperature on space requirements of group housed finishing pigs". In: *Applied Animal Behaviour Science* 138.3-4 (2012), pp. 229–239.
- [SAV20] Eli Stevens, Luca Antiga, and Thomas Viehmann. *Deep learning with pytorch*. Manning Publications Company, 2020.
- [SBF98] D Sergeant, R Boyle, and M Forbes. "Computer visual tracking of poultry". In: *Computers and Electronics in Agriculture* 21.1 (1998), pp. 1–18.
- [Sch15] Jürgen Schmidhuber. "Deep learning in neural networks: an overview". In: *Neural networks* 61 (2015), pp. 85–117.

Bibliography

- [SCL+19] Jaewon Sa, Yunchang Choi, Hanhaesol Lee, Yongwha Chung, Daihee Park, and Jinho Cho. “Fast pig detection with a top-view camera under various illumination conditions”. In: *Symmetry* 11.2 (2019), p. 266.
- [SCZ+08] Xuan Song, Jinshi Cui, Hongbin Zha, and Huijing Zhao. “Vision-based multiple interacting targets tracking via on-line supervised learning”. In: *European Conference on Computer Vision*. Springer. 2008, pp. 642–655.
- [SEZ+13] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. “Overfeat: integrated recognition, localization and detection using convolutional networks”. In: *arXiv preprint arXiv:1312.6229* (2013).
- [SG94] Joseph M Stookey and Harold W Gonyou. “The effects of regrouping on behavioral and production parameters in finishing swine”. In: *Journal of animal science* 72.11 (1994), pp. 2804–2811.
- [SGO05] Kevin Smith, Daniel Gatica-Perez, and J-M Odobez. “Using particles to track varying numbers of interacting people”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 1. IEEE. 2005, pp. 962–969.
- [Shi+94] Jianbo Shi et al. “Good features to track”. In: *1994 Proceedings of IEEE conference on computer vision and pattern recognition*. IEEE. 1994, pp. 593–600.
- [SIV+17] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. “Inception-v4, inception-resnet and the impact of residual connections on learning”. In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.
- [SJZ+20] Peize Sun, Yi Jiang, Rufeng Zhang, Enze Xie, Jinkun Cao, Xinting Hu, Tao Kong, Zehuan Yuan, Changhu Wang, and Ping Luo. “Transtrack: multiple-object tracking with transformer”. In: *arXiv preprint arXiv:2012.15460* (2020).

- [SK87] Lawrence Sirovich and Michael Kirby. “Low-dimensional procedure for the characterization of human faces”. In: *Josa a* 4.3 (1987), pp. 519–524.
- [SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.
- [SM00] Jianbo Shi and Jitendra Malik. “Normalized cuts and image segmentation”. In: *IEEE Transactions on pattern analysis and machine intelligence* 22.8 (2000), pp. 888–905.
- [Smi07] Kevin C Smith. “Bayesian methods for visual multi-object tracking with applications to human activity recognition”. In: (2007).
- [SRE+14] Christian Szegedy, Scott Reed, Dumitru Erhan, Dragomir Anguelov, and Sergey Ioffe. “Scalable, high-quality object detection”. In: *arXiv preprint arXiv:1412.1441* (2014).
- [SSS+17] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. “Revisiting unreasonable effectiveness of data in deep learning era”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 843–852.
- [Sta18] Statistisches Bundesamt. *Tierhaltung Weltweit*. Bestellnummer: 0030007-18900-1. 2018.
- [Sta20] Statistisches Bundesamt. *Viehbestand und tierische Erzeugung*. Fachserie 3 Reihe 4.1. 2020.
- [STE13] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. “Deep neural networks for object detection”. In: *Advances in neural information processing systems*. 2013, pp. 2553–2561.
- [SVI+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

Bibliography

- [SX08] Bin Shao and Hongwei Xin. “A real-time computer vision assessment and control of thermal comfort for group-housed pigs”. en. In: *Computers and Electronics in Agriculture* 62.1 (June 2008), pp. 15–21. ISSN: 01681699. DOI: 10.1016/j.compag.2007.09.006.
- [SZ14] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).
- [Tar97] Robert E Tarjan. “Dynamic trees as search trees via euler tours, applied to the network simplex algorithm”. In: *Mathematical Programming* 78.2 (1997), pp. 169–177.
- [TK91] Carlo Tomasi and Takeo Kanade. “Detection and tracking of point features”. In: (1991).
- [TL19] Mingxing Tan and Quoc Le. “Efficientnet: rethinking model scaling for convolutional neural networks”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6105–6114.
- [TOM97] R D Tillet, C M Onyango, and JA Marchant. “Using model-based image processing to track animal movements”. In: *Computers and electronics in agriculture* 17.2 (1997), pp. 249–261.
- [TP91] Matthew A Turk and Alex P Pentland. “Face recognition using eigenfaces”. In: *Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition*. IEEE Computer Society. 1991, pp. 586–587.
- [UCF+16] Jonas Uhrig, Marius Cordts, Uwe Franke, and Thomas Brox. “Pixel-level encoding and depth layering for instance-level semantic labeling”. In: *German Conference on Pattern Recognition*. Springer. 2016, pp. 14–25.
- [UVG+13] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. “Selective search for object recognition”. In: *International journal of computer vision* 104.2 (2013), pp. 154–171.

- [UVK+14] Winanda W Ursinus, Cornelis G Van Reenen, Bas Kemp, and J Elizabeth Bolhuis. "Tail biting behaviour and tail damage in pigs and the relationship with general behaviour: predicting the inevitable?" In: *Applied Animal Behaviour Science* 156 (2014), pp. 22–36.
- [VDP+03] Jaco Vermaak, Arnaud Doucet, Perez Patrick, et al. "Maintaining multi-modality through mixture tracking". In: *Computer Vision, IEEE International Conference on*. Vol. 3. IEEE Computer Society. 2003, pp. 1110–1110.
- [VIO+14] S. Viazzi, G. Ismayilova, M. Oczak, L.T. Sonoda, M. Fels, M. Guarino, E. Vranken, J. Hartung, C. Bahr, and D. Berckmans. "Image feature extraction for classification of aggressive interactions among pigs". en. In: *Computers and Electronics in Agriculture* 104 (June 2014), pp. 57–62. ISSN: 01681699. DOI: 10.1016/j.compag.2014.03.010.
- [VJ01] Paul Viola and Michael Jones. "Rapid object detection using a boosted cascade of simple features". In: *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*. Vol. 1. IEEE. 2001, pp. I–I.
- [VSV+09] Joost Van De Weijer, Cordelia Schmid, Jakob Verbeek, and Diane Larlus. "Learning color names for real-world applications". In: *IEEE Transactions on Image Processing* 18.7 (2009), pp. 1512–1523.
- [VTH+16] Christina Veit, Imke Traulsen, Mario Hasler, Karl-Heinz Tölle, Onno Burfeind, Elisabeth grosse Beilage, and Joachim Krieter. "Influence of raw material on the occurrence of tail-biting in undocked pigs". en. In: *Livestock Science* 191 (Sept. 2016), pp. 125–131. ISSN: 18711413. DOI: 10.1016/j.livsci.2016.07.009.
- [VUG+11] Koen EA Van de Sande, Jasper RR Uijlings, Theo Gevers, and Arnold WM Smeulders. "Segmentation as selective search for object recognition". In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 1879–1886.

Bibliography

- [WD09] Heleen A. van de Weerd and Jon E.L. Day. “A review of environmental enrichment for pigs housed in intensive housing systems”. en. In: *Applied Animal Behaviour Science* 116.1 (Jan. 2009), pp. 1–20. ISSN: 01681591. DOI: 10.1016/j.applanim.2008.08.001.
- [WS09] Kilian Q Weinberger and Lawrence K Saul. “Distance metric learning for large margin nearest neighbor classification.” In: *Journal of machine learning research* 10.2 (2009).
- [WZL+20] Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. “Towards real-time multi-object tracking”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI* 16. Springer. 2020, pp. 107–122.
- [WZW+21] Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. “Transformer meets tracker: exploiting temporal context for robust visual tracking”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 1571–1580.
- [XGD+17] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. “Aggregated residual transformations for deep neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1492–1500.
- [Yak19] Pavel Yakubovskiy. *Segmentation Models*. Publication Title: GitHub repository. GitHub, 2019. URL: https://github.com/qubvel/segmentation_models.
- [YCB+14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. “How transferable are features in deep neural networks?” In: *Advances in neural information processing systems*. 2014, pp. 3320–3328.
- [YK16] Fisher Yu and Vladlen Koltun. “Multi-scale context aggregation by dilated convolutions”. In: *International Conference on Learning Representations (ICLR)*. May 2016.

- [YN12] Bo Yang and Ram Nevatia. “Multi-target tracking by online learning of non-linear motion patterns and robust appearance models”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 1918–1925.
- [Zel14] Claudius Zelenka. “Gas bubble shape measurement and analysis”. In: *German Conference on Pattern Recognition*. Springer. 2014, pp. 743–749.
- [ZGY+18] Lei Zhang, Helen Gray, Xujiong Ye, Lisa Collins, and Nigel Allinson. “Automatic individual pig detection and tracking in surveillance videos”. In: *arXiv:1812.04901 [cs]* (Dec. 2018). arXiv: 1812.04901.
- [ZJR+15] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. “Conditional random fields as recurrent neural networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1529–1537.
- [ZLN08] Li Zhang, Yuan Li, and Ramakant Nevatia. “Global data association for multi-object tracking using network flows”. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, pp. 1–8.
- [ZSF+15] Ziyu Zhang, Alexander G Schwing, Sanja Fidler, and Raquel Urtasun. “Monocular object instance segmentation and depth ordering with cnns”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2614–2622.
- [ZSG+19] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. “Object detection in 20 years: a survey”. In: *arXiv preprint arXiv:1905.05055* (2019).
- [ZSQ+17] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. “Pyramid scene parsing network”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2881–2890.

Bibliography

- [ZVS+18] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. “Learning transferable architectures for scalable image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8697–8710.
- [ZWW+20] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. “Fairmot: on the fairness of detection and re-identification in multiple object tracking”. In: *arXiv preprint arXiv:2004.01888* (2020).