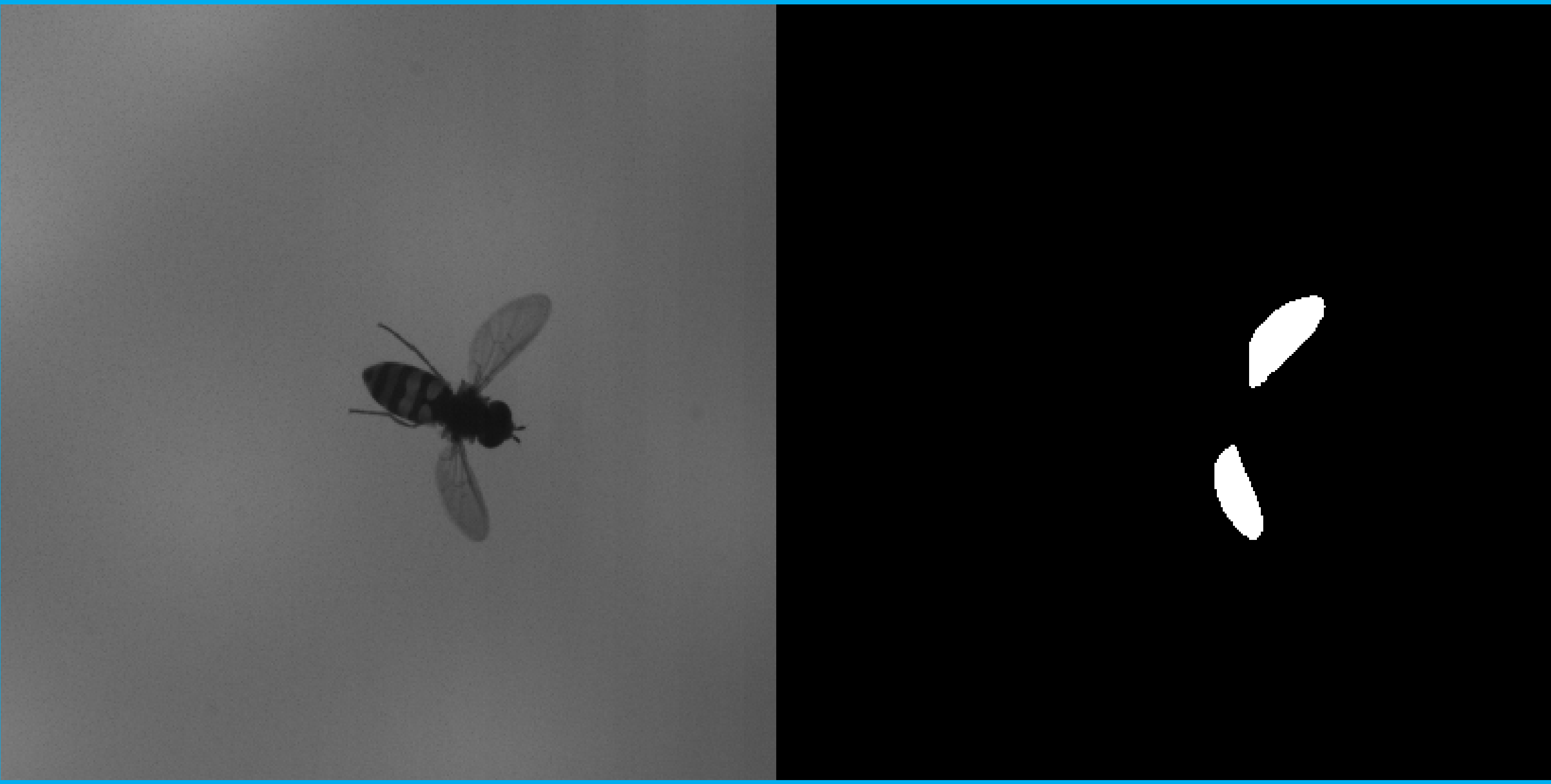
High-speed segmentation of Diptera wings during flight

Master Thesis

C. Zwart



High-speed segmentation of Diptera wings during flight

Master Thesis

by

C. Zwart

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday October 9, 2024 at 11:30 AM.

Student number:	5655447	
Project duration:	Februari 12, 2024 – October 9, 2024	
Thesis committee:	Dr. J. van Gemert,	TU Delft, Thesis Advisor
	Dr. S. Proksch,	TU Delft
	Dr. N. Tömen,	TU Delft, Daily Supervisor
	C. Feng	TU Delft, Daily Co-supervisor
	C. Wang	WUR, Daily Co-supervisor

TU Delft

Preface

This thesis forms the end of my 7 years of study. It has been an interesting journey and a pleasure to have had to opportunity to follow it. During the writing of this thesis I had the opportunity to learn, but more importantly apply my knowledge. For this I want to thank all my supervisors, but especially C. Feng, TU Delft, and C. Wang, Wageningen University of Research, with who were always available for discussions and guidance when I had questions.

Of course this work would not have been possible without friends, with whom I studied during the 9 months of research, and my family, for their support. Thank you all.

C. Zwart
Delft, October 2024

1 Introduction

Deep learning, and Computer Vision, are a rapidly growing field of research. Part of this groing research is the development of new methods and part is the application of existing methods in new fields of study. This thesis is part of the latter category, an application of known Computer Vision techniques in the field of experimental biology. Researchers at the Wageningen University of Research are interested in the flight of Diptera, two winged, insects. To accurately model their wingbeats high-speed camera's are used, generating a lot of data. The manual annotation of this data is labour intensive, which is where this thesis comes in. We developed a method to automatically and accurately identify the wings in video at high speed. This method speeds up annotation by a factor of atleast 2000 times faster.

The structure of this thesis is first a research article, followed by background information. In the research article we investigate methods to automatically identify wings in an image and study how model initialization and training method impact final accuracy. The background information is meant to further explain methods used in the paper for those interested in the topic.

Part I

Research paper

High-speed segmentation of Diptera wings during flight

Cornelis Zwart[1]*

[1]Delft University of Technology; *Corresponding to: cornelis.zwart.cz@gmail.com

Abstract

Insects such as Diptera are capable of highly complex aerial maneuvers and rapid responses to environmental stimuli, making them a subject of interest for studies in flight dynamics and motor control. To accurately quantify these movements, high-speed cameras are employed, capturing footage at 5000 frames per second. This results in a vast amount of data, necessitating the development of automated analysis models. In this work, we present a segmentation model specifically designed for the wings of Diptera species, aiming to track their wingbeats with high precision. The model's performance was evaluated on species included in the training set as well as on species only included in the testset, allowing us to assess its generalization capabilities. Our results show that the model achieves a 95th percentile error of 7.8 pixels, while operating at a speed 770 times faster than manual human analysis. This significant improvement in speed highlights the potential for our model to facilitate large-scale, high-speed analysis of insect flight, with implications for both biological research and bio-inspired engineering applications.

1. Introduction

Despite their simple brains insects are capable of complex aerial manoeuvres and and rapid responses. Researchers analyze these movements using high-speed cameras at thousands of frames per second, creating large amounts of data. The analysis of this data is done, largely, by hand, meaning it is highly labour intensive to analyze full flight sequences. In this work we present a Neural Network to automatically track the wings of Diptera species during flight to facilitate further automatic analysis. This analysis not only provides insight into the neural control mechanisms of flight but also has practical applications in fields such as pest control and bio-inspired engineering.

One key application is in pest control, where understanding insect flight behaviour can lead to more effective strategies for reducing health risks [10]. For instance, tracking mosquito flight patterns could help optimize the design of traps and repellents by revealing how they navigate environments and escape capture. By tailoring pest control devices to exploit these flight patterns, such as their response to airflow [8], or temperature and odour cues [9], we can develop better traps to reduce the spread of diseases like malaria.

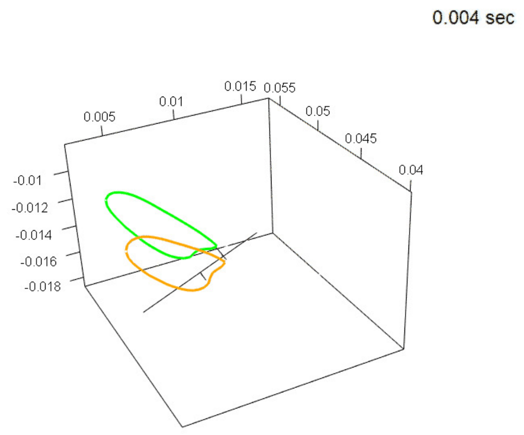


Figure 1: 3D reconstruction of Diptera during flight [22]

Another application is in the development of flapping-wing micro air vehicles(MAVs) [11, 28, 29]. These small drones aim to replicate the maneuverability, stability and energy efficiency of insects. This is particularly useful in confined space, for example in disaster zones [5] or artificial crop pollination [2]. By accurately capturing the deformations and aerodynamics of insect wings during flight, researchers can refine MAV designs to improve their performance in challenging conditions.

Currently the analysis of high-speed camera footage is done by hand, see figure 1, because automated models have not been able to track with sufficient accuracy. As these camera's can record at 5000 frames per second, the analysis becomes prohibitively expensive. Recent methods as DeepLabCut [26] have been able to track landmarks with human-level accuracy. However, during flight the wings of Diptera deform [36], meaning the entire outline has to be tracked. Creating enough landmark labels to do this accurately is still labour intensive.

In this paper we will implement a segmentation model to identify the wings of several Diptera species. Especially because of the high framerate at which data is collected this model will improve the automation of the analysis, and thus allow for greater volumes of data to be analyzed. Specifically we study the ability of the model to generalize to species not seen during the segmentation training phase. To this end we compare different weight initialization strategies and self-supervised training strategies, as well as whether masking part of the input during self-supervised training improves accuracy. In total the following are our contributions:

- Self-supervised learning does not improve the accuracy of the model.
- Masking part of the input during reconstruction training does improve the accuracy of the model.
- Propose the use of a Hausdorff distance to quantify the error in edge alignment for segmentation.
- Publish code, segmentation model and demonstration animation [43].

2. Related Works

DeepLabCut [26] is currently the state-of-the-art method for landmark prediction, achieving human-level accuracy with a RMSE of approximately 3 pixels on test sets after labeling around 800 frames per species. However, in our case, where approximately 80 landmarks would need to be tracked per frame, this approach becomes increasingly time-consuming. Labeling such a high number of landmarks for multiple frames is significantly more labor-intensive compared to labeling for segmentation tasks, where entire regions, such as wings, can be annotated more efficiently. Moreover, DeepLabCut offers limited flexibility in terms of custom training, evaluation, and dataset creation, making it difficult to adapt the tool for specific workflows or custom testing loops. Once all the necessary data is labeled, DeepLabCut does have advantages. It excels in accuracy, offers fast inference speeds, and provides straightforward calculation of 3D coordinates in multi-camera setups [27]. However, in cases where numerous landmarks must be tracked along edges, as in our application, segmentation-based methods are faster to set up and require less manual effort for labeling while still providing robust results.

Segment Anything The advantage of the methods presented in this paper relative to Segment Anything [19] are that our method, once trained, does not require human input. While SAM achieves good results, it requires human input for each frame. This has reportedly been mitigated Segment Anything 2 [31] which is context aware over consecutive frames, greatly reducing the number of frames that require human input. Due to time constraints this has not been verified for this usecase.

Transfer learning Transfer learning has become a widely used technique in computer vision, enabling models trained on one task to be applied to another [3, 18, 42], often with limited labeled data [15]. In this work, we employ transfer learning in two ways: (1) by initializing our encoder model with weights trained on ImageNet, and (2) by using a masked reconstruction training phase before fine-tuning the model for segmentation. The first step allows using features obtained from a large dataset, the second phase is closer to the final use case and thus allows to transfer more specific knowledge before the fine-tuning.

Masked autoencoders Masked autoencoders [16, 41] and inpainting [24, 30] have shown to be effective self-supervised learning techniques to learn structural features in input [4]. In these techniques a model is trained to reconstruct an image where part of the image has been occluded. The occlusion of local structure forces the model to infer local structure based on the global structure thereby improving the feature extraction. In our work we will apply this technique to learn features from unlabeled data after which we use transfer learning to apply the encoder to the segmentation task.

3. Methodology

3.1. Mask generator

U-Net is a model made for pixel-wise segmentation made for segmenting images [32]. In that setting they wanted to predict a class not for the image but for each pixel in the image. Additionally, in medical settings there is no millions of images to train on. They proposed a network, visualized in figure 2, with a contracting and expanding side. The contracting side is used to extract features from the data, while the expanding side is used to make pixel classifications on the original resolution.

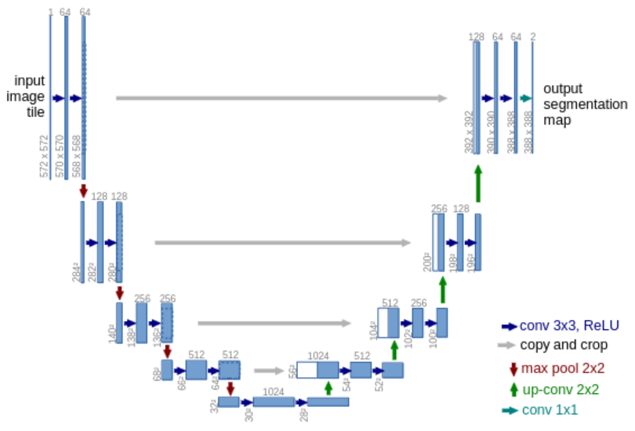


Figure 2: U-Net model architecture [32]

3.2. Auto-Encoder

Even though our amount of labelled data is limited, it has been shown that this is no issue for U-Net. However, we have more unlabelled data. Therefore a goal became to see if it is possible to learn general structures from that unlabelled data [14]. Therefore, we are interested in learning the general structure of the input data in a self-supervised manner, which can enhance the model's ability to understand and predict wing structures from limited labelled data. For this purpose, we utilize a masked auto-encoder.

A masked auto-encoder is designed to reconstruct an image from its compressed form, even when parts of the input image are masked out. This process encourages the model to understand and capture the underlying patterns and structures in the data. Specifically, the auto-encoder takes an input image, and applies a series of convolutional and pooling layers (the contracting path of the U-Net model) to compress the image into a lower resolution and higher dimensional embedding. This compressed representation is then used by the decoder part of the auto-encoder to reconstruct the original image.

To further enhance the learning process, we mask random patches of the input image before feeding it into the encoder. The model then attempts to reconstruct the missing patches during the decoding phase. This technique has been shown to work on vision transformers [16] and in general [30]. It forces the decoder to infer the missing parts based on the context provided by the visible patches. Consequently, the model learns to understand the local structures and global context of the wings, even when some parts are not directly visible.

The masked auto-encoder setup provides several advantages:

- Self-Supervised Learning: It allows the model to learn from unlabelled data by reconstructing missing parts, which is crucial when labelled data is scarce.
- Enhanced Feature Extraction: By learning to reconstruct masked images, the encoder becomes adept at extracting meaningful features that capture the essential structure and patterns of the input data.
- Robustness to Occlusions: The model becomes more robust to partial occlusions and variations in the input images, as it learns to predict and fill in missing information.

The auto-encoder is trained to minimize the reconstruction loss, which measures the difference between the original and the reconstructed image. This loss drives the learning process, guiding the model to capture the most important features of the wings.

In our combined model, the encoder part of the auto-encoder is shared with the U-Net model. This shared encoder leverages the features learned during the self-supervised pre-training phase, thus enhancing the U-Net's ability to perform precise pixel-wise segmentation of the wings. By integrating the auto-encoder with the U-Net, we combine the strengths of unsupervised feature learning with the precise segmentation capabilities of supervised learning, leading to improved performance in wing edge detection and overall segmentation tasks.

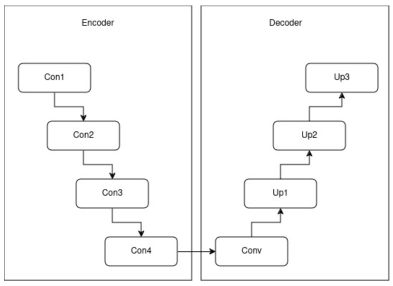
3.3. Combined model

The model we used uses the encoder from an auto-encoder to extract features from the image. For each downsampling the features are stored to serve in the expanding side of the U-Net. This allows to use of the feature extraction gained via unsupervised learning with the classification abilities of supervised learning. This architecture is visualized in figure 3a and figure 3b.

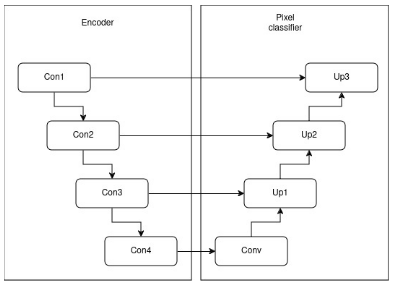
3.4. Metrics

To evaluate the accuracy of the proposed segmentation models a set of standard metrics are used. Additionally we propose the use of Hausdorff distance to quantify the error in predicted contours.

- **Intersection over Union(IoU):** This metric measures the overlap between predicted segmentation and the ground truth. A score of 0 indicates no overlap and a score of 1 indicated perfect overlap.
- **Precision and Recall:** Precision measures the proportion of correctly identified objects while recall measures the ability to find all true objects. Both are on a scale from 0, bad, to 1, perfect.
- **Mean Average Precision(mAP):** A derivation from the precision-recall curve, which quantifies the precision over varying confidence thresholds.



(a) Auto-encoder architecture with encoding and decoding sides



(b) U-Net architecture where the encoder can be shared with an auto-encoder.

- **F1**: The F1 score is a harmonic mean of the precision and recall. Specifically $F1 = 2\frac{precision*recall}{precision+recall}$.
- **Hausdorff distance:** The maximal distance from a point in set X to the closest point in set Y in pixels, and vice-versa. In this paper it is the greatest distance between the predicted wing edge and closest ground truth point, and vice versa. A lower score is better.

The Hausdorff distance is particularly useful in evaluating segmentation models because it captures the worst-case scenario of boundary mismatch between the predicted and ground truth contours. While other metrics like IoU provide an overall measure of overlap, they do not emphasize localized errors. The Hausdorff distance highlights the maximum deviation between the predicted contour and the actual boundary, making it highly sensitive to outliers or significant errors in shape prediction. This property is essential in applications like wing segmentation, where the precise delineation of fine structural details is critical. Thus, it provides a more rigorous assessment of boundary accuracy, especially in regions where minor deviations could significantly impact downstream analyses.

4. Experiments

4.1. Dataset

The dataset [23] we use contains 8 species of hoverfly with a total of 44 individuals. Included in the dataset are the images from the high-speed camera's and for each flight sequence a temporally sampled set of landmark labels. However, in the labeling process an error was introduced and the labels cannot directly be used [13], see figure 4. Since the creation of a new landmark based dataset is prohibitively expensive it was decided to use Segment Anything to generate a sizable dataset for segmentation of the wings. For each species we therefore labelled randomly selected samples, some of these were rejected because no insect was in them, or because the tool was not able to satisfactorily label it. Using this method selected 1342 images across 8 species, the breakdown of which can be seen in table 1.

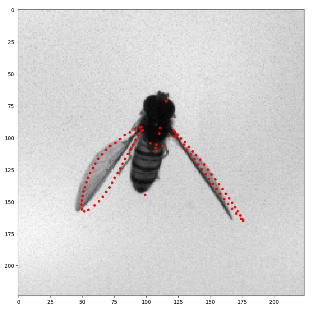


Figure 4: Points according to the dataset.

To train and evaluate our model we need ground-truth answers. That is, we need to specify for each pixel whether it is part of the wing or not. This can be done by selecting a set of points along the wing edge and inpainting this region. This however, still requires relatively much manual work, as each wing requires 20-30 points to be placed to accurately map the wing shape.

To make the labeling faster we experimented with Segment Anything [19]. This is a model developed by Facebook(META) for instance segmentation. Given an input image, the user selects points to include in the mask or exclude, from which SAM generates a mask. By placing more points the user then steers SAM to only segment the wing. An area SAM sometimes doesn't segment perfectly is the hinge.

Figure 5 shows the morphological differences between 2 of the species.

4.1.1 Data separation

To evaluate the model's ability to generalize to species not seen during training, we utilized a dataset comprising multiple flight sequences from various Diptera species. Two species, Episyrphus viridaureus and Eristalis tenax, were selected for training, with the remaining six species used exclusively for testing. This division ensures that the model is trained on a limited set while being tested on a broader range of species, thereby assessing its generalization capabilities. Because the images from a flight are not indepen-

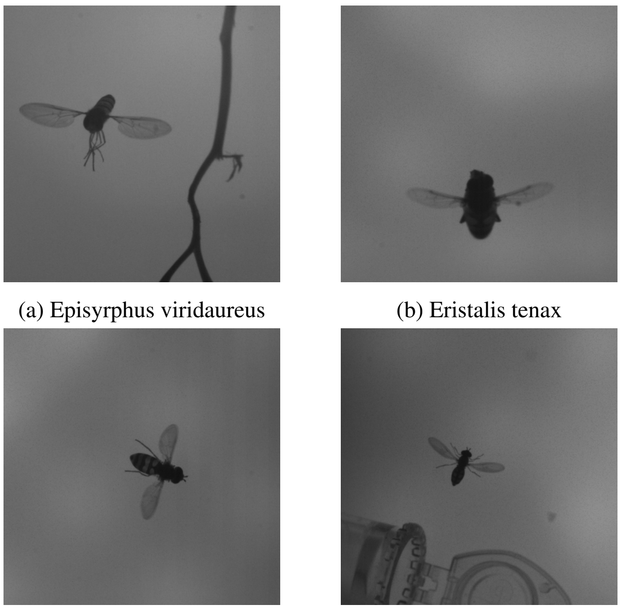

(a) Episyrphus viridaureus (b) Eristalis tenax (c) Eupeodes nielseni (d) Melanostoma mellinum

Figure 5: Difference in morphological structure for 4 of the species used to test for generalization over species. Species in figures 5a and 5b are used during segmentation training. Species in figures 5c and 5d belong to the unseen species

dent but temporally related, we allocate an entire sequence to train or test instead of individual frames.

Table 1 summarizes the distribution of flight sequences and labelled samples for each species. The training set includes four sequences (two per species) and 518 labelled samples. In contrast, the test set contains 19 sequences from six species, totalling 824 labelled samples.

4.1.2 Data augmentation

The training data was augmented with strategies as described in [34, 35] to simulate the variety of insect orientations. Specifically the following augmentations were used:

- Vertical mirroring
- Horizontal mirroring
- Rotate 90°

Each of these augmentations had a independent 50% probability of occurring, allowing multiple transformations on a single image. For computational reasons the image was cropped to 384x384px, centered on the mask. For images with no mask the createBackgroundSubtractorMOG2 class from openCV [6] was used with default parameters, the center of the cropped image was determined as the average location of foreground pixels.

4.2. Training procedure

The training of our model is divided into two distinct phases: the training of the auto-encoder (AE) and the subsequent training of the segmentation model. These phases are designed to leverage self-supervised learning for robust feature extraction and supervised learning for precise pixel-wise segmentation.

4.2.1 Auto-Encoder Training

In the first phase, the auto-encoder is trained on the dataset using a self-supervised learning approach. The primary objective is to teach the encoder to extract meaningful features by reconstructing the original images. To this end, the encoder converts the image into a lower spatial dimension after which the decoder tries to reconstruct the original image.

The auto-encoder uses a Mean Squared Error (MSE) loss function to measure the reconstruction error between the original and the reconstructed image. The model is trained using the Adam optimizer with a learning rate of 1×10^{-4}. Each 1000 frames we log several metrics, if after 10 such intervals the loss hasn't improved we stop the training prematurely.

This phase allows the model to learn a general representation of the insects from the dataset without requiring any labelled data, thus maximizing the utilization of available data and enhancing the robustness of the feature extractor.

4.2.2 Segmentation Training

After the auto-encoder training, the model is trained for the specific task of wing segmentation. In this phase, the encoder from the pre-trained auto-encoder is shared with the U-Net model, allowing it to benefit from the previously learned features. The U-Net model is then fine-tuned to predict precise pixel-wise segmentation using labelled data.

For the segmentation task, a weighted Dice loss function is employed to account for the class imbalance between the wing and background pixels. The Adam optimizer is used for this phase with the same learning rate of 1×10^{-4}. The training is performed with a batch size of 3.

During training, data augmentation techniques such as vertical and horizontal flipping and rotations in multiples of 90 degrees are applied as described before. To monitor the model's performance, Intersection over Union (IoU) is calculated for each sample and logged along with the training loss. The training process continues until the model shows signs of convergence, determined by the stability of the IoU scores and the training loss. The best model, based on IoU performance, is saved and used for evaluation on the test set.

Species	Train sequences	Train samples	Test sequences	Test samples
Episyrphus viridaureus	2	272	1	25
Eristalis tenax	2	246	1	130
Eupeodes nielseni	0	0	3	178
Melanostoma mellinum	0	0	3	152
Platycheirus clypeatus	0	0	2	114
Sphaerophoria scripta	0	0	3	75
Syrphus nigrilinearus	0	0	3	75
Tropidia scita	0	0	3	75
Total	4	518	19	824

Table 1: Distribution of flight sequences and labelled samples per species for training and testing.

4.3. Post-processing

The output of the trained model often contains small masked areas that are wrongly identified. To reduce the number such false positives and improve overal accuracy we implemented a series of test-time augmentations. At test time the model predicts the input images, as well as copies that are rotated 90, 180 or 270 degrees or have been mirrored along the vertical or horizontal axis. The predicted masks are transformed back to the original orientation after which a consensus-based approach is used: a pixel is classified as wing if the majority of transformed masks agree it is part of the wing. This majority voting strategy improves the robustness of the predictions by reducing the impact of random noise and eliminating many spurious objects. Furthermore, to refine the segmentation results and reduce the number of small, erroneously identified areas, we applied an area-based filtering step. Analysis of the training set revealed that 99.9% of the ground truth wing segments have an area of at least 57 pixels. Based on this observation, we set a threshold to exclude any segmented objects with an area smaller than 50 pixels. This filtering criterion effectively reduces the occurrence of false positives by removing improbably small segments that are unlikely to represent actual wings.

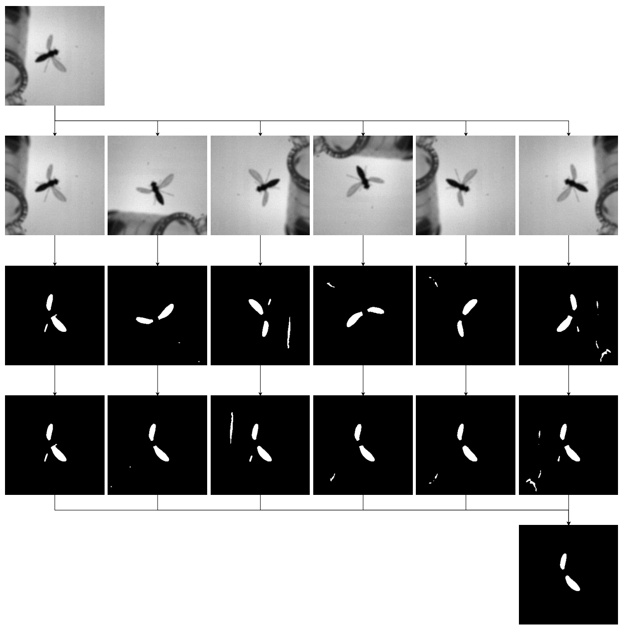


Figure 6: Visual representation of post-processing. First row: original input. Second row: Transformed input images, original, rotated, vertically and horizontally mirrored. Third row: model predictions based on transformed input. Fourth row: Prediction transformed to the original orientation. Fifth row: Majority voted pixels where each detection has a size of at least 50px.

4.4. Performance testing

We measure the performance of our model on a testset. As described before, these are images never seen during training. The evaluation is done separately for species seen during training and those that have not been seen during training. The measures we are interested in are IoU, recall, accuracy

The data was split to obtain a traingset and an unseen testset. We took the first sequence for each species for the testset and the remaining sequences for the trainingset.

As the goal of this thesis is to track the wing the most important feature is the distance between the predicted wing-edge and the true wing-edge. This can be measured with the Haufdorff distance. The Haufdorff distance measures the maximum distance you can be forced to travel to go from a point in the predicted edge to the closest point in the true edge.

Since we are not only interested in minimizing the distance to *a* point on the true edge but want the prediction to be close to *all* points on the true edge, we calculate the Haufdorff distance both from true-edge to predicted-edge and from predicted-edge to true-edge.

4.5. Effect of initialization and training procedure

To investigate the impact of different model initialization and training procedure we train and compare 4 models. The parameters under study are whether the model is initialized with weight obtained by training on ImageNet [33] and whether the model is trained using the auto-encoder training described before. The models are as follows: (1) training directly on the labeled segmentation data without ImageNet initialization or unsupervised learning (baseline), (2) utilizing unsupervised learning alone before fine-tuning on the labeled data, (3) employing ImageNet initialization followed by fine-tuning on the labeled segmentation data, and (4) combining ImageNet initialization with unsupervised learning before fine-tuning the segmentation. The encoder, share by the segmentation and reconstruction models, is based on the ResNet [17] and can therefore be initialized with weights obtained from models trained to classify images from Imagenet. In the experiments with ImageNet initialization the weights in the encoder are thus taken from a, trained by others, ResNet model. In experiments with no ImageNet initialization the encoder weights are initialized randomly, the weights for the reconstruction decoder and segmentation decoder are random in every experiment.

The auto-encoder phase is trained on a sampling from all species in the dataset, with augmentation described in section 4.1.2. During segmentation training the model is trained on 2 sequences each from Episyrphus viridaureus and Eristalis tenax, see table 1. Evaluation is done on a third sequence from these two species, the results of which are marked as ”seen species” and can be found in table 2, and the remaining 6 species which are marked ”unseen species” of which the results can be found in table 3.

These configurations were evaluated on two sets of species: those included in the training data (”seen” species) and those excluded from the training data (”unseen” species).

Tables 2 and table 3 display the results for evaluation on species seen during training and not seen during training respectively. As expected, the models initialized with ImageNet weights perform better on most metrics than the models with random initialization, with higher IoU’s, precision and 50th and 95th hausdorff percentiles. Surprisingly, including unsupervised learning in the training-pipeline improves the hausdorff distances for seen species but not for unseen species, even though it learns from unseen species during the unsupervised training. What causes this lack of improvement is unknown as of yet.

4.6. Effect of partial masking

As unsupervised training based on reconstructing the input did not improve performance on species not seen during segmentation training, we decided to experiment with another form of unsupervised learning based on the process used for Masked Auto Encoders [16]. During the unsupervised training the model reconstructs the original input image, while the provided input image is partially obscured. The goal is that the model is forced to learn global context, since local context may miss. The initialization and training process for these experiments is similar to experiment (4) from section 4.5, where the encoder is initialized with weights obtained from a model trained to classify images from the ImageNet dataset, again the reconstruction decoder and segmentation decoder are initialized randomly. The data used in the training of the masked auto-encoder is again a sampling from images from all 8 species. However, in these experiments patches of 8px by 8px are randomly masked, totalling a set percentage that changes per experiment. Experiments were run for 0%(baseline), 20%, 40%, 60% and 80%. In MAE’s performance is good even when masking 70% of the input, which is why such high percentages are tested.

The segmentation training is identical to that in section 4.5, where the model is trained on Episyrphus viridaureus and Eristalis tenax. The results of segmentation is again split between the testset of species seen during segmentation training, table 4, and species not seen during segmentation training, table 5.

For all expiriments we observe almost no impact on IoU, mAP, precision, recall and F1. On species seen during segmentation training we see that higher masking fractions lead to higher 75th and 95th percentile hausdorff distances, except for the 95th percentile at 80% masking.

For species not seen during segmentation training we observe that the 95th percentile hausdorff distance improves when 60% or 80% of the image is masked during unsupervised training. In the case 60% is masked the 95th percentile hausdorff distance improves 1.4 pixels, indicating there are fewer outliers in the predictions.

5. Discussion

As shown in the previous section, the best model is obtained when using transfer learning with ImageNet weights. We then train it in a self-supervised manner to reconstruct input with 60% of the pixels masked and when loss converges the model is trained to segment wings. On species not seen during training this results in 75% of images having a Hausdorff distance of less than 3 which is considered human-level accuracy [26], and 95% of images having a max Hausdorff distance of 7.8px. Precision and recall are 0.97 and 0.96 respectively, indicating that almost all detections indeed are part of the wing, and that most wings are indeed identified. Therefore the model can be used as a reliable basis for fast, 13 frames per second, segmentation of the wings. Currently the model is evaluated on cropped images, with a resolution approximately 37% of the full image. Using the model to segment the full image is possible,

ImageNet	Unsupervised	IoU	mAP	Precision	Recall	F1	Hausdorff 50%	Hausdorff 75%	Hausdorff 95%
No	No	0.87	0.68	0.95	1.00	0.97	3.6	7.2	199.3
No	Yes	0.88	0.68	0.96	1.00	0.98	3.6	7.0	15.5
Yes	No	0.93	0.67	1.00	1.00	1.00	3.6	6.0	9.7
Yes	Yes	0.92	0.69	0.99	1.00	1.00	3.0	5.3	9.0

Table 2: Performance on species **seen** during training, different training configurations (Imagenet pretraining and Unsupervised learning). Hausdorff distances are in pixels.

ImageNet	Unsupervised	IoU	mAP	Precision	Recall	F1	Hausdorff 50%	Hausdorff 75%	Hausdorff 95%
No	No	0.85	0.66	0.93	0.97	0.95	2.0	3.6	213.0
No	Yes	0.84	0.67	0.91	0.97	0.94	2.0	3.6	229.2
Yes	No	0.89	0.66	0.96	0.96	0.96	2.0	3.0	9.1
Yes	Yes	0.89	0.66	0.96	0.96	0.96	2.0	3.0	9.2

Table 3: Performance on species **not seen** during training, different training configurations (Imagenet pretraining and Unsupervised learning). Hausdorff distances are in pixels.

but is both slower and worse accuracy because of more superfluous detections. To mitigate these factors a bounding box object detector should be trained, as it will minimize the computational requirements per frame and lowers the area for superfluous detections.

Of note in our results is that Hausdorff distances are, mostly, lower for species not seen in the segmentation training than they are for species used in the segmentation training, while the IoU, mAP, Precision, Recall and F1 are better for the species seen during training. This is true for both the experiments concerning initialization and the experiments with masked self-supervised learning. The loss does not indicate overfitting, as the train- and testloss don't diverge. For all metrics we expected the performance to be better in species seen during segmentation training, we therefore recommend further investigation into this result in the future.

6. Future work

To research Diptera flight it's movement need to be followed in 3d, therefore there is need for further research in this topic. Given that DLT constants are known the normal procedure would be to match points in different camera views and calculate the corresponding point in 3d space. In our scenario this is not possible, as it is unknown which points along the wing in cameraview 1 correspond to which points along the wing in cameraview 2. For our purposes a method of 3d point reconstruction needs to be developed that does not require matching points. Two methods we suggest for further investigation are:

- populate 3d space with points, calculate their corresponding 2d points, discard points far away from the segmentation, iteratively generate 3d points closer together around non-discarded points.
- in segmentation1 generate points evenly spaced around each object, calculate epipolar lines in segmentation2. The matching point lie on a intersection of the epipolar line and an object segmentation in segmentation2. However, a heuristic is needed to determine which intersection is the corresponding point.

Another avenue of improvement is the use of the temporal relation between frames. Since the framerate of the camera's used is very high the movement of wings between two frames should be relatively minor. We see the following as possible improvements:

- Each frame is segmented individually, like is now the case. However, in post-processing all objects that do not overlap with an object in the previous frame is rejected. This should reduce the number of superfluous objects, improving precision and the Hausdorff 95th percentile. More on this method can be found in the appendix.
- Since the encoder is based on ResNet it has 3 input channels, of which only 1 is used because the data is grayscale. The other two channels might be populated with the previous 2 input frames or the segmentation masks for the previous 2 frames, or the image and segmentation from the previous frame.

7. Conclusions

In this thesis we presented a segmentation model designed to track the wings of Diptera species during flight. By integrating transfer learning with self-supervised masked auto-encoders we demonstrated how pre-training on unlabeled data improves the generalization to species not seen during segmentation training. Our model showed im-

Mask fraction	IoU	mAP	Precision	Recall	F1	Hausdorff 50%	Hausdorff 75%	Hausdorff 95%
0.0	0.92	0.69	0.99	1.00	1.00	3.0	5.4	9.2
0.2	0.92	0.68	0.99	0.99	0.99	3.0	5.4	10.4
0.4	0.93	0.68	1.00	1.00	1.00	3.0	5.8	11.4
0.6	0.93	0.68	1.00	1.00	1.00	3.0	5.9	11.4
0.8	0.92	0.68	1.00	1.00	1.00	3.0	6.0	10.1

Table 4: Performance on species **seen** during training, different fractions of masked input during pre-training. Hausdorff distances are in pixels.

Mask fraction	IoU	mAP	Precision	Recall	F1	Hausdorff 50%	Hausdorff 75%	Hausdorff 95%
0.0	0.89	0.66	0.96	0.96	0.96	2.0	3.0	9.2
0.2	0.89	0.66	0.95	0.96	0.95	2.0	3.0	11.0
0.4	0.88	0.65	0.94	0.95	0.95	2.0	3.0	21.6
0.6	0.90	0.65	0.97	0.96	0.96	2.0	3.0	7.8
0.8	0.88	0.64	0.97	0.95	0.96	2.0	3.0	9.0

Table 5: Performance on species **not seen** during training, different fractions of masked input during pre-training. Hausdorff distances are in pixels.

provements in accuracy when trained initialized with ImageNet weights and when a masked auto-encoder was used with 60% of the input masked. The model achieved human-level results in 75% of the cases, and remained useful in another 20% with errors of at most 7.8px. This work contributes to the intersection of experimental biology and computer vision by evaluating and publishing a model to automate the analysis of Diptera species in flight on high-speed camera's. On low-cost consumer hardware speeds of 13 frames segmented per second were achieved, indicating at least a factor 770 speed-up compared to common human labeling strategies.

References

[1] . Most used activation functions in neural networks. https://ai-artificial-intelligence.webyes.com.br/most-used-activation-functions-in-neural-networks/, 2024. [Online; accessed 8-October-2024]. 15

[2] Shaik Abdullah, Priyasha Appari, Srihari Rao Patri, and Srinivas Katkoori. Smart agriculture using flapping-wing micro aerial vehicles (fwmavs). In *IFIP International Internet of Things Conference*, pages 32–47. Springer, 2021. 2, 14

[3] Abdelilah Adiba, Hicham Hajji, and Mustapha Maatouk. Transfer learning and u-net for buildings segmentation. In *Proceedings of the New Challenges in Data Sciences: Acts of the Second Conference of the Moroccan Classification Society*, pages 1–6, 2019. 3

[4] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 3

[5] Yingcai Bi, Menglu Lan, Jiaxin Li, Shupeng Lai, and Ben M Chen. A lightweight autonomous mav for indoor search and rescue. *Asian Journal of Control*, 21(4):1732–1744, 2019. 2, 14

[6] G. Bradski. The OpenCV Library, 2000. 6

[7] Centre for Disease Control. Malaria's impact worldwide, 2024. [Online; accessed 8-October-2024]. 13

[8] Antoine Cribellier, Leonardo Honfi Camilo, Pulkit Goyal, and Florian T. Muijres. Mosquitoes escape looming threats by actively flying with the bow wave induced by the attacker. *Current Biology*, 34(6):1194–1205.e7, 2024. 2, 13

[9] Antoine Cribellier, Jeroen Spitzen, Henry Fairbairn, Cedric Van De Geer, Johan L Van Leeuwen, and Florian T Muijres. Lure, retain, and catch malaria mosquitoes. how heat and humidity improve odour-baited trap performance. *Malaria Journal*, 19:1–16, 2020. 2

[10] Antoine Cribellier, Jens A van Erp, Alexandra Hiscox, Martin J Lankheet, Johan L van Leeuwen, Jeroen Spitzen, and Florian T Muijres. Flight behaviour of malaria mosquitoes around odour-baited traps: capture and escape dynamics. *Royal Society open science*, 5(8):180246, 2018. 2, 13

[11] GCHE De Croon, M Perçin, B Remes, R Ruijsink, and C De Wagter. The delfly. *Dordrecht: Springer Netherlands. doi*, 10:978–94, 2016. 2

[12] DELFLY. Delfly. https://www.delfly.nl/, 2024. [Online; accessed 8-October-2024]. 14

[13] Ebraheem I. Fontaine, Francisco Zabala, Michael H. Dickinson, and Joel W. Burdick. Wing and body motion during flight initiation in Drosophilarevealed by automated visual tracking. *Journal of Experimental Biology*, 212(9):1307–1323, 05 2009. 5, 20, 23

[14] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations, 2018. 4

[15] 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 (CVPR)*, June 2014. 3

[16] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021. 3, 4, 8

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 8, 21

[18] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016. 3

[19] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023. 3, 5

[20] Siddharth Krishnan. A comprehensive guide to convolutional neural networks — the eli5 way. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a5, 2018. Accessed: 2024-09-30. 16

[21] V7 labs. Mean average precision (map) explained: Everything you need to know. https://www.v7labs.com/blog/mean-average-precisionmean-average-precision-for-object-detection. 20

[22] Camille Le Roy. personal communication. 2

[23] Camille Le Roy, Nina Tervelde, Thomas Engels, and Florian T. Muijres. Changes in wing morphology rather than wingbeat kinematics enabled evolutionary miniaturization of hoverflies. June 2024. 5, 20

[24] Guilin Liu, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 3

[25] Ning Lv, Zenghui Zhang, Cong Li, Jiaxuan Deng, Tao Su, Chen Chen, and Yang Zhou. A hybrid-attention semantic segmentation network for remote sensing interpretation in land-use surveillance. *International Journal of Machine Learning and Cybernetics*, 14(2):395–406, 2023. 21

[26] Mamidanna P. Cury K.M. et al Mathis, A. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, page 1281–1289, 2018. 2, 3, 8, 20

[27] Tanmay Nath, Alexander Mathis, An Chi Chen, Amir Patel, Matthias Bethge, and Mackenzie Weygandt Mathis. Using deeplabcut for 3d markerless pose estimation across species and behaviors. *Nature protocols*, 14(7):2152–2176, 2019. 3

[28] Quoc-Viet Nguyen, Woei Leong Chan, and Marco Debiasi. An insect-inspired flapping wing micro air vehicle with double wing clap-fling effects and capability of sustained hovering. In *Bioinspiration, Biomimetics, and Bioreplication 2015*, volume 9429, pages 136–146. SPIE, 2015. 2

[29] Quoc-Viet Nguyen, Woei Leong Chan, and Marco Debiasi. Hybrid design and performance tests of a hovering insect-inspired flapping-wing micro aerial vehicle. *Journal of Bionic Engineering*, 13(2):235–248, 2016. 2

[30] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 3, 4

[31] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos, 2024. 3

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. 2015. 3, 4

[33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 8

[34] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019. 6

[35] Luke Taylor and Geoff Nitschke. Improving deep learning using generic data augmentation, 2017. 6

[36] Simon M Walker, Adrian LR Thomas, and Graham K Taylor. Deformable wing kinematics in free-flying hoverflies. *Journal of the Royal Society Interface*, 7(42):131–142, 2010. 2

[37] Wikipedia contributors. Hausdorff distance — Wikipedia, the free encyclopedia, 2024. [Online; accessed 15-October-2024]. 20

[38] Wikipedia contributors. Jaccard index — Wikipedia, the free encyclopedia, 2024. [Online; accessed 15-October-2024]. 19

[39] Wikipedia contributors. Neural network (machine learning) — Wikipedia, the free encyclopedia, 2024. [Online; accessed 16-October-2024]. 14

[40] Wikipedia contributors. Precision and recall — Wikipedia, the free encyclopedia, 2024. [Online; accessed 15-October-2024]. 19, 20

[41] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9653–9663, June 2022. 3

[42] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014. 3

[43] C. Zwart. Dipterasegmentation. `https://github.com/cornelis1998/DipteraSegmentation`, 2024. 3

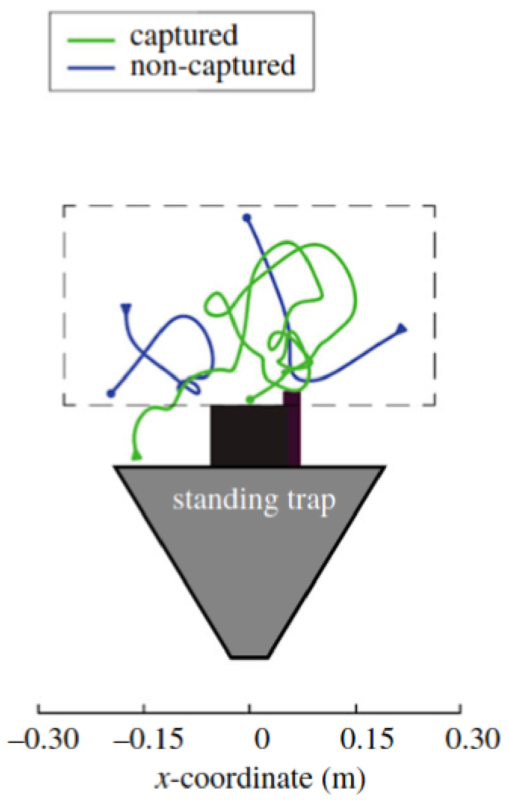
Part II
Background material

The following part is to give more information about the thesis. In general the setting will be more informal, with a focus on intuitive explanations rather than formal ones. The sections will vary from a more detailed motivation for the research, a quick primer on neural networks and their components, an explanation of the model we used, how data was labeled, how we measured the success of experiments, and finally some experiments that were interesting but not ultimately necessary for our research questions.

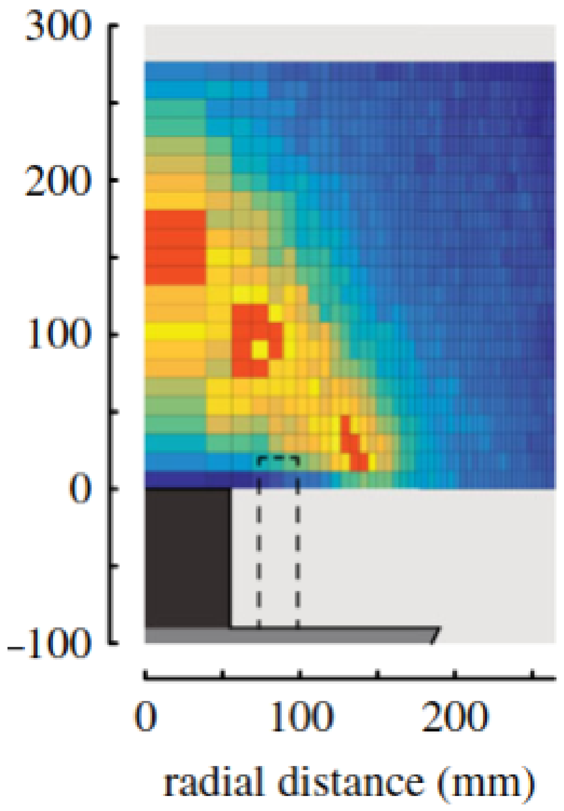
1. Motivation

Most people probably won't think much about insects, however they can be very interesting. The Experimental Zoology group from the Wageningen University, among many researchers, studies them for several reasons. Two of the reasons are that pest control and to create drones with flapping wings, so called flapping wing micro aerial vehicles(FWMAV).

For example, in 2022 Malaria, as discease spread by mosquitos, killed 608.000 people [7]. This makes the creation of better traps for mosquitos a lifesaving topic. An example of tracking mosquitos helped improve traps can be found in figure 7. Here they tracked the flight of mosquito's to map how they behaved around a trap. In a later study experiments were performed to quantify the importance of several factors to the likelyhood of captering mosquitos. For example mosquitos were much more likely to approach the trap if it was heated and water is inside. Quantification of such differences allows for a systematic design of traps to protect people by captering mosquito's that might otherwise infect them with malaria.



(a) Diagram of a standing mosquito trap.



(b) Heatmap showing where mosquito's where most likely to fly. Random flight would result in a uniform distribution

Figure 7: Diagram showing the flight of mosquito's around a trap. [10]

This research does not require the tracking of individual parts, like wings, of the mosquito. A simple rectangle around their body is enough to track their location through 3d space. However another paper looked at how mosquito's are able to avoid fly swatters. Here they tracked many points along the wings and body of an mosquito, while a mechanical swatter tried to hit them. Because the location and the way it moves is known through time, it was possible to calculate the the forces exuded by the wings and thus how it should accelerate. When the predicted acceleration and actual acceleration didn't match it was clear that mosquito's "surf" on the high air pressure produces by the swatter [8]. Because mosquito's beat their wings rapidly the cameras used record at several thousand frames per second. The analysis of that much data is only feasible with automatic analysis methods, they developed specifically for mosquito's. However, since this model is not tested on other species it is not known how accurate the results would be.

The other application mentioned is the creation of drones that fly like insects. One example of this the DelFly [?] drone,

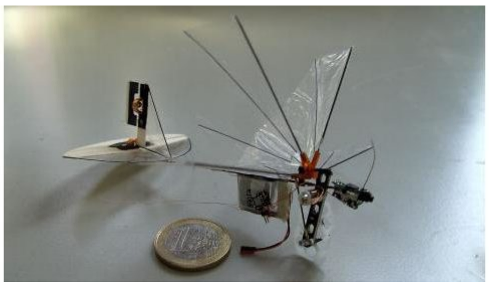


Figure 8: DelFly drone [12]

developed by TUDelft students. It is a drone not much larger than a coin, capable of flying any direction by beating it's wings. In this way it roughly mimics the way insects fly through nature. Of course before insect flight can be mimicked we need to be able to track and model their flight. For now these tiny robots aren't used much in practice, but they hold potential for tasks like artificial pollination [2] or for reconnaissance in disasters [5].

In this paper we therefore discuss the development and evaluation of a computer vision model that can be used to track many Diptera, insect, species during flight. The hope is that other research can use this to analyze vast amounts of footage to develop a better understanding of their flight, instead of being limited by the amount that can be manually processed.

2. Neural networks

Over the last decade neural networks(NN's) have become increasingly popular, to a point where most people use them unknowingly. In principle it is a program that takes some input, for example an image, and produces an output, like whether there is a dog in the image. What makes this type of program increadibly usefull is that humans don't need to explain what a dog looks like, but that the program can learn this itself. After all, it's increadibly hard to explain in exact rules what a dog looks like based on individual pixels in an image. We give a neural network a set of images for which we know whether a dog is present and let it learn from this data. After a while it will understand what a dog is and will be able to say whether one is in a new image that is presented. Below we will give a more detailed explanation of ”regular” artificial neural networks and then progress to image based networks.

2.1. Base architecture

Neural networks are inspired by the neurons in our brain. A neuron can store some piece of, numerical, information, manipulate it, and pass it on to another neuron. Each neuron has a base value, called bias, and each connection has a weight. The value of a neuron is then determined by the weighted sum of the input, plus its bias. Lets say we want to predict what the temperature tomorrow will be, as input we use the temperature from two days ago, yesterday and today.

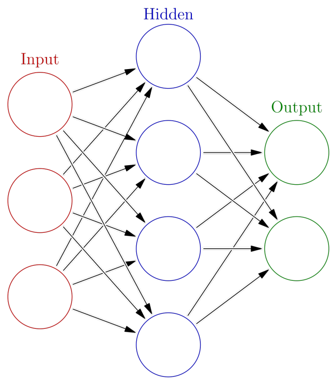


Figure 9: Visual representation of an neural network. The left nodes are given the input to the model. Hidden layers learn increasingly abstract representations of the data. The output layer gives the desired conclusion based on the data, e.g. the temperature tomorrow. Arrows indicate the flow of information from one neuron to another. [39]

The value of each hidden or output neuron is:

$$z = b + \sum x_i * w_i$$

The value from each neuron is then used to compute the values in the next layer. So far each neuron is a linear combination of the input, limiting the behaviour that can be expressed by the model. To allow for more complex behaviour neural networks introduce non-linear functions after calculating z.

$$a = \sigma(z) = \sigma(b + \sum x_i * w_i)$$

where σ is the activation function. An overview of the most used activation functions is shown in figure 10. These additions of the activation function allows us to model non-linear relations, it further allows succesive layers of the neural network to represent increasingly complex relationships. The ability for deeper networks to represent more complex data has resulted in increasingly deep networks, called Deep Neural Networks and Deep learning.

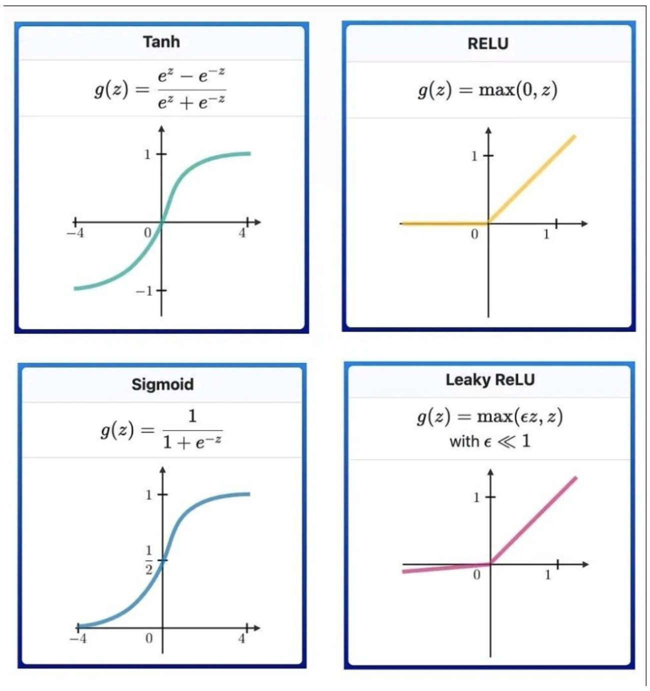


Figure 10: Most used activation functions. The x-axis is the value ”z” from the computed neuron, the y-value gives the output and thus value of ”a”. [1]

2.2. Learning

So far we have looked at a situation where the weights are already known. However in reality we don’t know what the weights are, so we need a way to find them. Initially we start with random weights, meaning that our prediction will be random and thus terrible. In order to train we have to quantify how bad the prediction is, which is done by a loss function. The most commonly used function is the Mean Squared Error(MSE). The first step is to make a set of predictions on samples for which we know the correct answers, from this we calculate the difference with the prediction, square it and finally take the mean of all samples. Here lower loss is better, as the mean square error is lower. The next step is to learn with a method called back propagation. Since the model is a combination of linear dependencies and activation functions with derivatives, it is possible to calculate the derivative of the weights w.r.t. the input. Therefore we know how each weight should be changed to reduce the loss, thus reducing the error. We change the weights a little bit in that direction, and repeat the process. Over several iterations the loss will lower until it plateaus and the best error for this model has been achieved.

3. Convolutional Neural Networks (CNNs)

While traditional neural networks are powerful, they are not always the most efficient choice for processing specific types of data, such as images. In images the pixels next to eachother are related, together they form edges, lines, and combinations of those. CNNs are a specialized kind of neural network designed to recognize patterns and structures in visual data, making them exceptionally well-suited for tasks like image classification, object detection and more segmentation, such as in this paper.

Images are composed of pixels arranged in a grid, and nearby pixels are often related to each other, forming edges, textures, and shapes. Traditional neural networks treat each pixel as an independent input, which ignores these spatial relationships and can lead to a large number of parameters, making the model inefficient. CNN's improve upon this by treating patches of the image instead of pixels.

Imagine we have the image of a car, as in figure 11. In the first layer of the CNN we use a convolution filter. This is a small window that moves over the image, checking whether a certain feature, like an edge or corner, is present around each pixel. Because it uses the same parameters everywhere in the image it can find the feature everywhere equally well. This reduces the number of parameters that have to be learned, which makes training the network easier. The result of applying these filters is a set of feature maps, which show where certain features are found in the image. As we go deeper in the network, these feature maps get combined, allowing the CNN to recognize more complex structures, like wheels or windows in the case of a car. What makes CNN's powerful is that it can detect features no matter where in the image they are located. Early layers focus on edges and corners, while features in later layers indicate the presence of wheels, doors or the car.

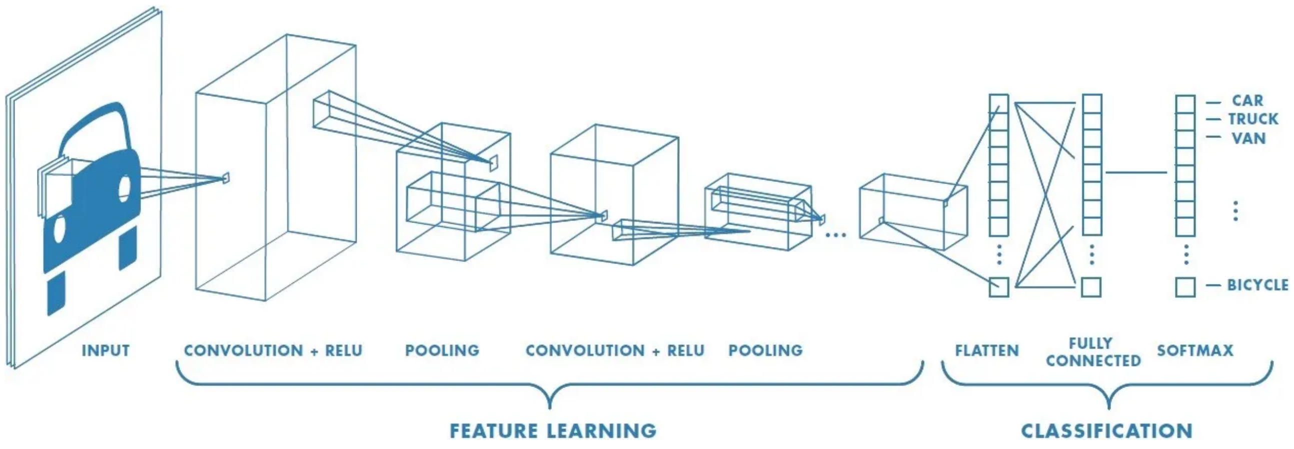


Figure 11: Illustration of a convolutional layer applying multiple filters to an input image, resulting in several feature maps. [20]

3.0.1 Activation Functions

After each convolution operation an activation function, like in regular NN's is applied to introduce non-linearity into the model. This allows the network to learn more complex patterns. This is done for each pixel in the feature map individually.

3.0.2 Pooling Layers

Each featuremap produced in a CNN is, depending on some settings, the same size as it's input. In some cases, however, we want to reduce this size. For example if we want to detect whether there is a car in the image a CNN is useful to detect the wheels, doors window etc, but in the final layer we don't care where in the image the car is. In that case we use a max pooling layer, just like a convolution it takes a window from the feature map but returns the maximal value in that window. The key difference is that instead of the window moving 1 pixel to the right before it repeats there is no overlap in the windows. This reduces the resolution of the feature map, giving a courser but more global knowledge.

In summary, Convolutional Neural Networks enhance the capabilities of traditional neural networks by efficiently capturing and learning from the spatial hierarchies present in image data. Their specialized architecture makes them indispensable tools in modern computer vision applications.

4. Network training

The model used in this paper consists of 3 parts that together are used to segment the wings of insects, visualized in figure 12. In our case all these parts are variations on CNN's, but with specific tasks.

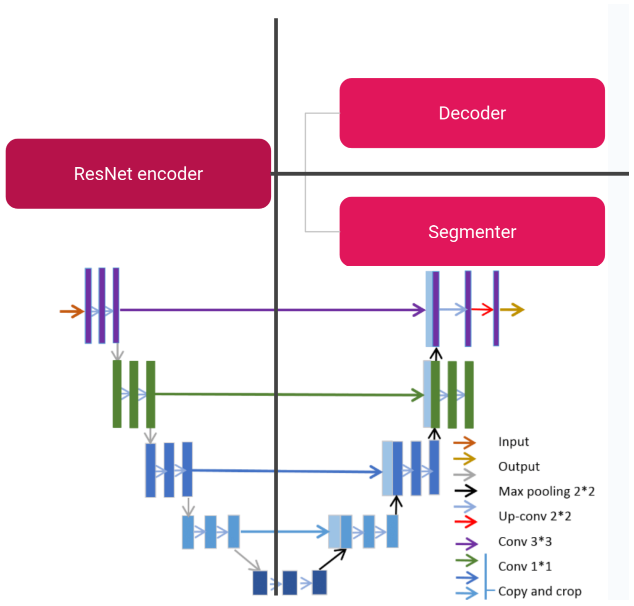


Figure 12: Model architecture used in this paper.

4.1. Feature extractor

The first stage of the network is called a feature extractor or encoder. It's goal is to extract general features, such as the lines, corners, and body parts. The layers extract increasingly complex patterns while reducing the size of the image, making the knowledge more global. Because this process is general in nature it doesn't change much between our use case and that of others. It is therefore common to use a general model, trained on large datasets, to perform this step. This means that we start, initialize, our model with weights of another trained network. In our first research question, we quantified the effect of initializing the model with general weights before training on our use case. Here we concluded that initializing with weights from ImageNet increased the accuracy greatly.

4.2. Reconstruction

A large part of our research was to quantify how well the model generalizes to species it has not seen during training. The reason for this is that we labelled data for 8 species, while in Wageningen the total dataset encompasses 37 species. If the model only performs well on species for which we put in significant effort to label, then the amount of labour saved is minimal. It would be nice if we could learn not only from the "limited" dataset that was labelled but also from images that are not labelled. A common technique to do this is to reconstruct the input. Given an image, we extract the features with the feature extractor before we try to recreate the original image. The model that reconstructs the input from the extracted features, also called a decoder, is the second part of the 3 models we used. In this case, the loss function is again the MSE function, the average squared difference between the reconstructed and original input image. During training the loss decreases, meaning it can reconstruct the image better. This means that the feature extractor learns to extract information specific to insects instead of the dataset it was originally trained on.

4.3. Segmentation

The final module of our network is the segmentation module, that will indicate for each pixel whether it is part of the wing or not. Our segmentation module employs a U-Net architecture, known for it's ability to learn from limited data and the incorporation of skip connections. These skip connections play a crucial role in preserving fine-grained spatial information by directly transferring feature maps from the encoder to the corresponding decoder layers. This design ensures that the network retains detailed contextual information, enabling it to produce more accurate and detailed segmentation masks. To reconstruct the spatial dimensions lost during the encoding phase, the segmentation module utilizes upsampling layers that progressively increase the size of the feature maps. Each upsampling step is combined with partially encoded features of the same resolution followed by convolutional layers that refine the feature representations, enhancing the network's ability to capture complex patterns and local structures within the image. The output layer uses the sigmoid activation function, which scales each pixel to a value between 0 and 1, where 1 indicates the probability of a pixel being part of the wing. The last step, only needed during evaluation, is to go from probabilty to either wing or not. This is simply done by defining all probabilities equal to or greater than 50% as wing and all others as not wing.

4.4. Masked input

The second research question we tested was to see if accuracy improves if we mask part of the input image during the training of the reconstruction module. Literature suggests that up to 75% of the image can be masked, made black, while reconstruction remains possible. The reasoning is that the area around the masked patch gives information about what was hidden. In figure 13 the input, reconstruction and original are shown. Even though the reconstruction may not look great, the body is located correctly and a vague outline can be seen at the base of the wings. More importantly, after training with this fraction of the image masked, the model has learned the features better then with no masking.

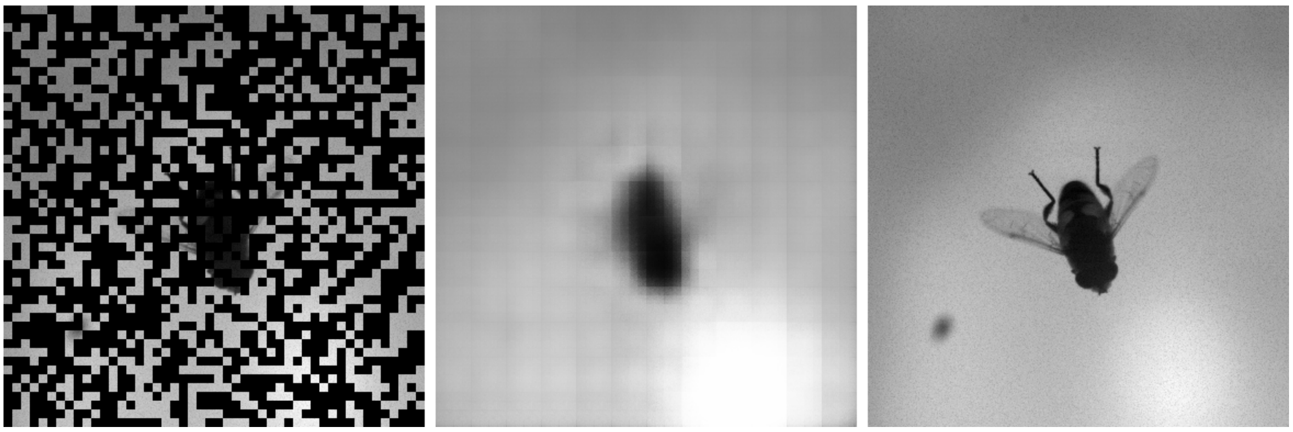


Figure 13: Left the input image, masked 60%. Middle is the reconstructed image. Right is the original image.

5. Data labeling

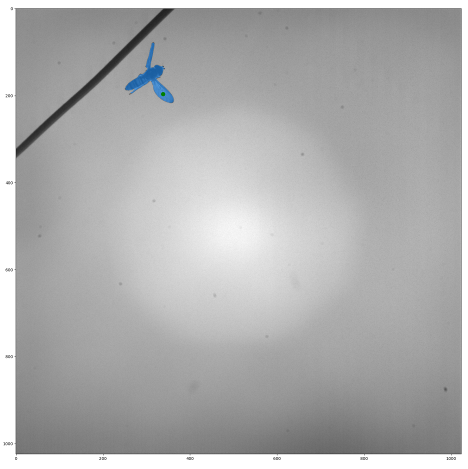
To train the neural network which pixels are part of the wing we first need to know it before the network can learn it. For this I used a homemade setup that uses Segment Anything, a model developed by Meta. You give it an input image, in our case of the insect and a point in the region of interest. The model then masks the region it thinks you are interested in. This can be further refined by giving other points that should be included, or giving points that should be excluded.

Using this method it was possible to label approximately 90 images in an hour, instead of the 10 if this process was done fully manually. At this speed, it is still highly impractical to label all 68.000 frames, so after selecting a species to label the frames were sampled randomly over that species.

Using this method 1342 frames were labelled, which was enough to train the network as well as have a testset that is not used during training to evaluate the accuracy of the model.

5.1. Continuous improvement

The model has trouble labeling some frames, where it thinks there are more than 2 wings in the image. Due to the experimental setup we are sure that in each frame there are at most 2 wings, sometimes it is only 1 if the wing is behind the body. To continually improve the model a workflow would be for the model to predict where wings would be in an image, if there are more than 2 wings it should request a human to label the image. After labeling some images the model can be re-trained after which this process is repeated. This minimizes the number of frames labeled.


(a) One green point on the bottom wing indicates it should be included in the mask. The highlighted blue section is what the labeling model expects the clicked object is.

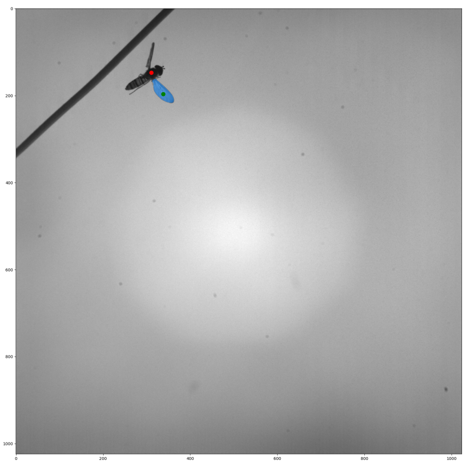

(b) A red, exclusion, point is added relative to figure a. Now the labeling model understands we are only interested in the wing.

Figure 14: The tool used for labeling. Points can be added, for each specified whether they should be included in the mask or excluded from it.

6. Metrics

In the paper we already gave a brief introduction to the metrics used, here we will give a more detailed explanation, including figures and equations.

IoU [38] The Intersection over Union measures the similarity between two sets. Graphically this can be seen in figure 15, with the equation $IoU = \frac{|A \cap B|}{|A \cup B|}$. A perfect overlap has a score of 1 and no overlap has a score of 0.

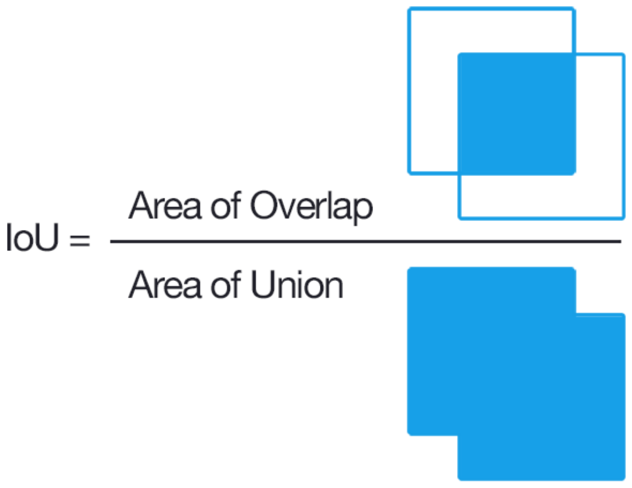


Figure 15: Intersection over Union [38]

Precision and Recall [40] Precision measures the fraction of predicted object belong to the ground truth. Recall measures the fraction of objects in the ground truth that are detected by the model. In formula's we can say that $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$ where TP is the number of correctly predicted object, FN is the number of missed objects and FP is the FP is the number of predicted objects where none should be according to the ground truth. To determine if an object overlaps it is common to say it is overlaps if it scores an IoU of at least 0.5.

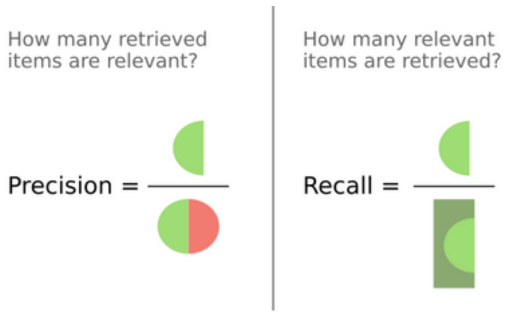

How many retrieved items are relevant?
Precision =
How many relevant items are retrieved?
Recall =

Figure 16: On the left a visual explanation of precision. On the right a visual representation of recall [40]

F1 score The F1 score is the harmonic mean of precision and recall, and it is used to balance the trade-off between the two metrics, especially when one is more important than the other in a specific task. The formula for the F1 score is $F1 = \frac{2*Precision*Recall}{Precision+Recall}$, a perfect score is 1 and the worst is a 0. The benefit of the F1 score is that it provides a single, comprehensive metric that captures the overall performance, even when precision and recall values differ significantly.

mAP [21] The mean Average Precision gives a view of how well the model performs at different IoU thresholds. For a given set of IoU values, in our case 0.5, 0.55 etc up to 0.95, the precision and recall is calculated. These pairs of precision-recall pairs are sorted from lowest recal to highest. The mAP is then calculated as the weighted average precision, where the weight is the increase in recall over the past IoU theshold.

Hausdorff distance [37] The final metric we used is the Hausdorff distance. It is the greatest minimal distance between two sets, in our case between the groundtruth wing outline and the predicted wing outline. Visually this can be seen in figure 17, assume the blue line is the groundtruth outline of the wing and the red line is the predicted outline of the wing. The Hausdorff distance is the largest distance you can be forced to move to go from one line to the other. For our use case this is a great metric, as we are most interested in how well the outline of our predictions match the true shape and location of the wing. One disadvantage it has is that it is very sensitive to small outliers. If in an image we have a perfect overlap between a prediction and the true wing but a tiny extra object is detected on the other side of the image the Hausdorff distance becomes enormous even though it is clear to humans that this is a small error. This is why we displayed percentiles of the Hausdorff distances, the few outliers won't affect the 50th and 75th percentile, and often not even the 95th percentile. Because in our images multiple wings are often visible we consider each wing independently, calculating the distance between a predicted object and the groundtruth object that has the largest IoU. When a prediction has no overlap the distance is calculated to the closest groundtruth object.

7. Experiments

During this thesis several experiments have been performed, only some of which made it into the final paper. Below I will highlight some of the experiments performed, their motivation and results.

7.1. Landmark prediction

Originally, the goal of this thesis was to develop a landmark based model, see figure 20. For this, three datasets were provided used by people from the Wageningen Experimental Zoology group, a dataset containing Bumblebees, one containing Mosquito's, and finally the dataset later used in this thesis which contains a total of 37 species, 8 of which have been shared [23]. Of these the Bumblebee and Mosquito datasets were labeled using DeepLabCut [26], and the last dataset was labeled with an extension on a Matlab tool "Kine" [13].

As stated the data provided was labelled in DeepLabCut(DLC), and thus stored in its preferred data format. In this dataset the insects were extracted and the frames from each of the 3 cameras were cropped and connected side-by-side, creating a single image three times as wide as it is high, see figure 18.

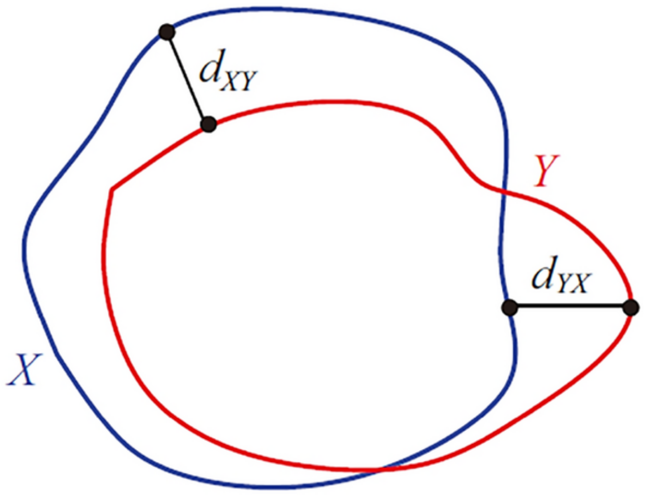


Figure 17: Visual representation of the Hausdorff distance. The blue line is the groundtruth wing outline. The red line is the predicted wing outline. [25]

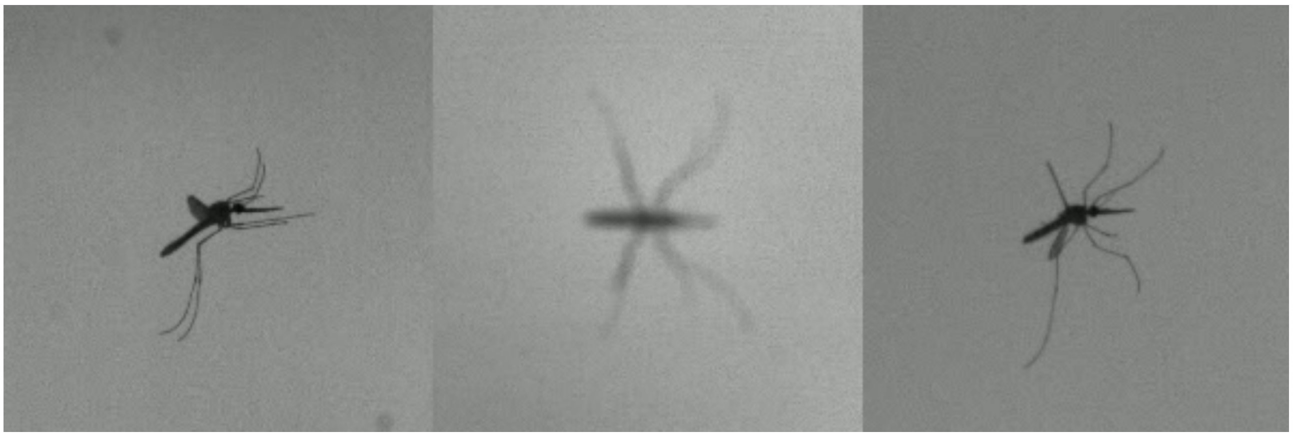
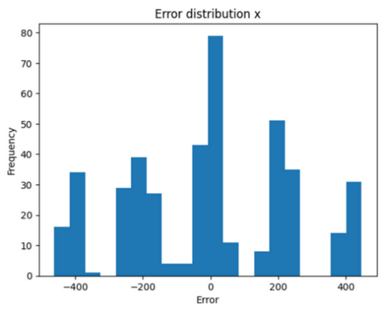


Figure 18: Three camera perspectives at the same moment in time stitched together. This was the input to the model used here.

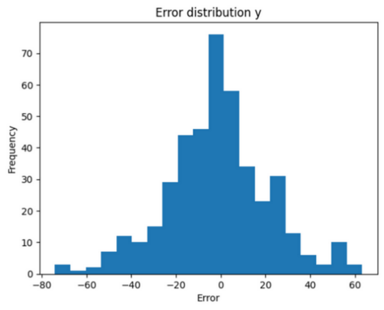
The first experiment performed focused on the current model used to track mosquitoes, with 600x200px of resolution while ResNet works on a resolution of 224x224. Our assumption was therefore that, because the image had to be resized to 224x224px, this impacted the accuracy of the model. Our expectation was that because the image was wider than tall the X-error would be worse than the Y-error. A histogram of the X-error19a, Y-error19b and total-error19c can be seen in figure 19. Here it is clear that the Y-error is roughly normally distributed, which is to be expected. However, the error in the X-direction has 5 peaks, at -400px, -200px, 0px, 200px and 400px. This seems to indicate that the model confuses whether a landmark is in the left, middle or right patch of the image causing it to be offset in increments of those frame sizes.

Because the images used are gray-scale, and ResNet uses Red, Green and Blue colour images, it is possible to not put the different camera views side-by-side but each in one of the red, green or blue channels of the input. While experimenting with this new setup we concluded that working with DLC works well until custom code is needed. Therefore, we looked at implementing a landmark-based model using the same architecture, a ResNet backbone connected with a feed-forward neural network to do the final prediction of confidence, X- and Y-coordinates for each landmark. This gave us the additional freedom to choose which of the ResNet model sizes, 18, 34, 50, 101 or 152 layers, layers to use. The expected result is that, given enough data, the deeper models will perform better, at the cost of speed [17].

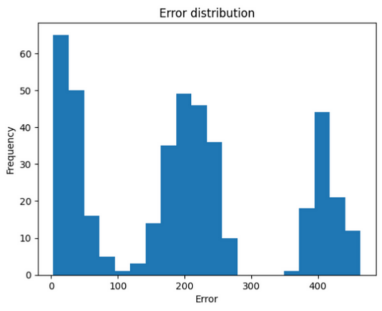
In addition to to these results a histogram showed that the x-error now only has 1 peak. The biggest problem with the



(a) DeepLabCut X-error.



(b) DeepLabCut Y-error.



(c) DeepLabCut total-error.

Figure 19: Automatically generated pointclouds of a single wing based on 2 cameras.

ResNet size	Total-error	X-error	Y-error
18	14.4	9.8	8.4
34	20.2	14.4	11.1
50	14.4	9.2	9.2
101	22.9	16.6	12.4
152	18.9	13.1	10.9

Table 6: Accuracy of landmark detection for different ResNet model sizes.

Resnet layers	Hausdorff 50%(px)	Hausdorff 75%(px)	Hausdorff 95%(px)	Processing speed(fps)
18	1.4	2.8	9	18.6
34	2	3	10.8	16.9
50	2	3	14.7	15.2
101	2	3.6	220	12.6
152	2	3.1	212	9.5

Table 7: Comparison of different ResNet backbones on segmentation accuracy and speed

dataset was that it only contained a single species since the bumblebee and mosquito datasets had different landmarks tracked. Therefore the decision was made to use the dataset with 8 published species.

7.2. Impact of model size on segmentation

Since the goal of this paper was to quantify the impact of training techniques on the accuracy not much time was spend on researching the impact of different ResNet backbones. However, like in the previous section, where we explored the impact on landmark prediction, a simple hyper-parameter sweep was performed. In these experiments the model was initialized with weights obtained from training on ImageNet and no reconstruction training was performed. The goal of this experiment is to determine which ResNet-backbone to use in our research. This decision will be made on a combination of accuracy and speed. A faster processing speed allows for more experiments, lowering the effort needed to try many settings. Therefore, as long as faster models don't perform extremely poorly preference will be given to fast models. The results can be found in table 7, please note that this experiment was not identical to the one in the paper and should thus not be directly compared.

With this data several interesting things become clear:

1. The large models often perform worse than the small models. This is likely due to the small amount of labelled data available, where the model overfits on the training. Because this was a short trail only to decide which ResNet backbone to use no further investigation was performed.

2. Larger models are slower, with the ResNet-152 performing at approximately half the speed of ResNet-18.

Since the accuracy is actually best for the small models, where training speed is also highest, we decided to use the ResNet-18 backbone for all experiments moving forward.

Mask fraction	Hausdorff 50%	Hausdorff 75%	Hausdorff 90%
0.2	7.456	39.909	84.684
0.4	10.48	38.947	78.684
0.6	9.05	40.157	81.752
0.8	7.26	43.19	82.646

Table 8: Results of segmentation, on species not seen during segmentation training, without test-time augmentation.

7.3. Test-Time augmentation

As described in the paper, at the moment of evaluation the model segments for each each of a number of transformations on the input image. The segmented images are then transformed back, and a pixel is considered part of the wing if a majority of masks think it is part of the wing. An additional restriction is placed on the output, that at most 4 objects can exist and wings should have a minimum size. Though in each frame at most 1 insect, and thus 2 wings, are present it may occur that the test-tube is wrongly considered a wing, it is not in the training data and thus unfamiliar to the model. To give some leeway in this regard we decided to only include the 4 biggest detection's in the final output, in practice this worked well. Additionally from our trainingset we know the distribution of wing sizes, in number of pixels surface. Any objects smaller than 50px in surface are discarded, as this would be less than 0.1% of cases in the trainingset. The removed detection's would therefore likely be random noise instead of legitimate wings. A table with results without test-time augmentation can be seen in table 8, again please note that the results were not produced in identical circumstances as the results in the paper. However, it is clear that these results are a factor 3 to 10 higher and thus worse.

8. 3D reconstruction

The ultimate goal of this research is to have a 3D digital model of the wing movements at high speed. With this information, simulations can be run to calculate the forces on the wings, for example to see how insects evade traps or to build mechanical insects. We therefore want the outline of the wing in 3d coordinates, so we can assume that the surface it encloses is the wing.

Before the filming of flight sequences the camera's are calibrated. This results in a set of 12 parameters that can be used to map points between 2d and 3d using a process called Direct Linear Transformation(DLT). In our set there are 3 camera's that capture a 3d space from different angles. Using the DLT parameters and a point in 3d space we can calculate for each camera in which pixel the point would be. However, from 2d to 3d this process becomes more involved as a 2d point corresponds with a line in 3d space.

8.1. Landmark based

If we have a point p_0 in camera 0 and p_1 in camera 1 which we know belong to the same physical point, e.g. the wing tip it is easy to calculate the corresponding point in 3d. After all, a 2d point projects a line in 3d, the lines corresponding to p_0 and p_1 intersect in exactly 1 place. Of course due to some measurement errors it should be expected that these lines don't exactly intersect. However they should be close enough to reliably determine the 3d coordinates of the landmark.

When the dataset was created for each frame 97 points were tracked in 3d. However, these points were not saved as the 2d coordinates for each camera but as 3d coordinates [13]. Using the DLT's it is possible to calculate the 2d coordinates, which was our original approach. We then found that these 2d points are not accurate enough to train an NN. In figure 20 is an example of the 3d points converted to 2d, the points clearly have the shape of an wing but are offset in some areas. The cause of this was the assumption that a wing is flat, which lead to erroneous 3d coordinates, and thus erroneous 2d coordinates. For the original intent of the dataset this error was not important, however in our case the model needs to learn where on the wing a 'wingtip' is. If the 'wingtip' label is placed in a gray region it is impossible to distinguish from another gray patch and thus the model cannot accurately learn to place it.

8.2. Segmentation based

Because the lack of labels for landmark based tracking we switched to a segmentation models, i.e. for each pixel we say whether it is part of the wing or not. As discussed in the paper our main focus was to correctly place the contour of the wing. We have the locations of the wings in 2d, now what? This already allows for some analysis of the shape in 2d, but again we would prefer to see the wing move in 3d so forces on the wings can be calculated.

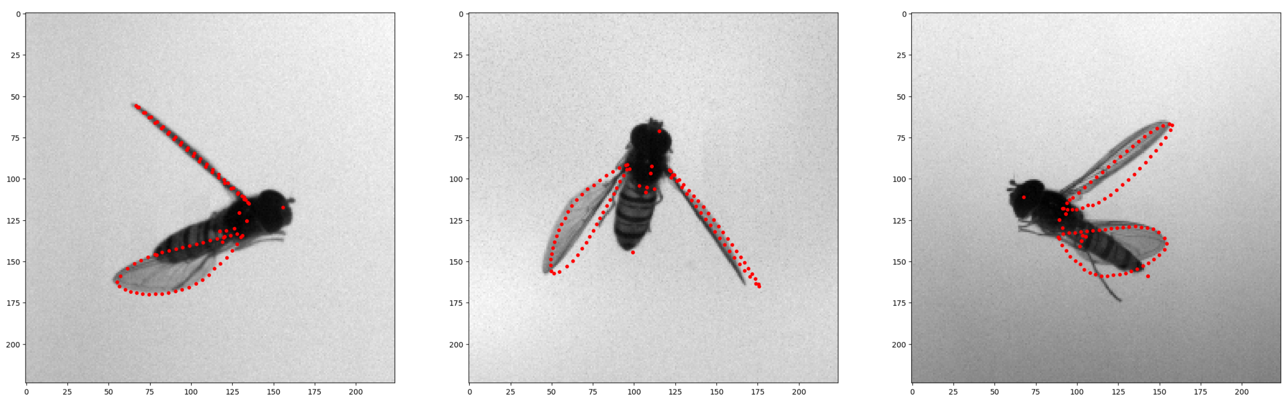


Figure 20: 3 camera views made at the same moment. Red dots indicate different 3d points converted to their 2d counterparts. Here it is clear the points don't always lie on the edge of the wing. (NOTE: 8 points on belong on the body of the insect)

The main issue with the segmentation is that we don't have points in 2 camera's of which we know they belong to the same physical point, e.g. the wing tip. So we can't go from 2 2d points to a 3d point, however we can still go from a 3d point to 2 2d points. So the question becomes: how do we go from segmentation masks and DLT parameters to a set of 3d points that outline the wing.

The naive solution is to generate 1 million points in 3d, for each calculate the 2d points, and keep the 3d points whose projection lay on the edge of the segmentation. Computationally this is not feasible, therefore we need an approximation. I used the following process

Algorithm 1 Iterative 3D Point Refinement Based on 2D Mask Distances

```
Generate 250 random points in 3D space.
each 3D point Project the 3D point onto the 2D plane (using camera or DLT parameters).
each projected 2D point Calculate the distance from the point to the nearest edge of the mask.
Sum the distances for the projected points corresponding to the 3D point.
Sort the 3D points based on their cumulative 2d distance (to the mask).
Remove the 50% of 3D points with the worst (highest) cumulative distances.
each remaining 3D point Generate new points at a distance of $\epsilon$ in each of the six cardinal directions (±x, ±y, ±z).
Halve the value of $\epsilon$ for the next iteration.
Repeat the process until convergence or the desired number of iterations is reached.
```

Using this process we generated the 3d pointcloud displayed in figure 21a. When looking perpendicular to the plane of the wing this results in a nice wing shape.

However, when looking from the edge of the wing the result becomes less perfect, as shown in figure 21b. Here we can see that the reconstruction makes 2 wings, one behind the other at an angle. The image this reconstruction is based on contains a single wing, so the error is due to an inaccuracy of matching points between camera 0 and camera 1. I suspect the error originates from the point in camera 0 being on the leading edge of the wing while in camera 1 it is projected to the trailing edge. The current metric does not differentiate between this, thus introducing an error of several 2.4mm at the wingtips.

I believe that with further tweaking of the code this can be fixed. For example logic can be applied that the direction to the inside of the wing should be the same,

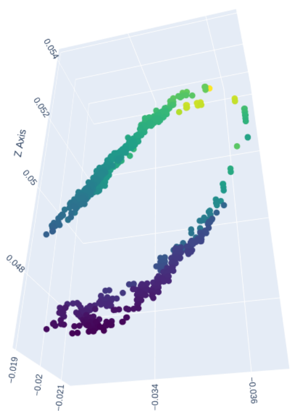
References

[1] . Most used activation functions in neural networks. https://ai-artificial-intelligence.webyes.com.br/most-used-activation-functions-in-neural-networks/, 2024. [Online; accessed 8-October-2024]. 15

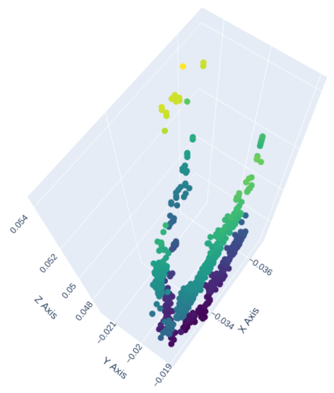
[2] Shaik Abdullah, Priyasha Appari, Srihari Rao Patri, and Srinivas Katkoori. Smart agriculture using flapping-wing micro aerial vehicles (fwmavs). In *IFIP International Internet of Things Conference*, pages 32–47. Springer, 2021. 2, 14

[3] Abdelilah Adiba, Hicham Hajji, and Mustapha Maatouk. Transfer learning and u-net for buildings segmentation. In *Proceedings of the New Challenges in Data Sciences: Acts of the Second Conference of the Moroccan Classification Society*, pages 1–6, 2019. 3

[4] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 3



(a) Automatically generated point cloud shown from the front.



(b) Automatically generated point cloud shown from the side.

Figure 21: Automatically generated pointclouds of a single wing based on 2 camera's.

[5] Yingcai Bi, Menglu Lan, Jiaxin Li, Shupeng Lai, and Ben M Chen. A lightweight autonomous mav for indoor search and rescue. *Asian Journal of Control*, 21(4):1732–1744, 2019. 2, 14

[6] G. Bradski. The OpenCV Library, 2000. 6

[7] Centre for Disease Control. Malaria's impact worldwide, 2024. [Online; accessed 8-October-2024]. 13

[8] Antoine Cribellier, Leonardo Honfi Camilo, Pulkit Goyal, and Florian T. Muijres. Mosquitoes escape looming threats by actively flying with the bow wave induced by the attacker. *Current Biology*, 34(6):1194–1205.e7, 2024. 2, 13

[9] Antoine Cribellier, Jeroen Spitzen, Henry Fairbairn, Cedric Van De Geer, Johan L Van Leeuwen, and Florian T Muijres. Lure, retain, and catch malaria mosquitoes. how heat and humidity improve odour-baited trap performance. *Malaria Journal*, 19:1–16, 2020. 2

[10] Antoine Cribellier, Jens A van Erp, Alexandra Hiscox, Martin J Lankheet, Johan L van Leeuwen, Jeroen Spitzen, and Florian T Muijres. Flight behaviour of malaria mosquitoes around odour-baited traps: capture and escape dynamics. *Royal Society open science*, 5(8):180246, 2018. 2, 13

[11] GCHE De Croon, M Perçin, B Remes, R Ruijsink, and C De Wagter. The delfly. *Dordrecht: Springer Netherlands. doi*, 10:978–94, 2016. 2

[12] DELFLY. Delfly. https://www.delfly.nl/, 2024. [Online; accessed 8-October-2024]. 14

[13] Ebraheem I. Fontaine, Francisco Zabala, Michael H. Dickinson, and Joel W. Burdick. Wing and body motion during flight initiation in Drosophilarevealed by automated visual tracking. *Journal of Experimental Biology*, 212(9):1307–1323, 05 2009. 5, 20, 23

[14] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations, 2018. 4

[15] 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 (CVPR)*, June 2014. 3

[16] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021. 3, 4, 8

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 8, 21

[18] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016. 3

[19] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023. 3, 5

[20] Siddharth Krishnan. A comprehensive guide to convolutional neural networks — the eli5 way. https://towardsdatascience.com/a-comprehensive-guide-to- convolutional-neural-networks-the-eli5-way-3bd2b1164a5, 2018. Accessed: 2024-09-30. 16

[21] V7 labs. Mean average precision (map) explained: Everything you need to know. https://www.v7labs.com/blog/mean-average-precisionmean-average-precision-for-object-detection. 20

[22] Camille Le Roy. personal communication. 2

[23] Camille Le Roy, Nina Tervelde, Thomas Engels, and Florian T. Muijres. Changes in wing morphology rather than wingbeat kinematics enabled evolutionary miniaturization of hoverflies. June 2024. 5, 20

[24] Guilin Liu, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 3

[25] Ning Lv, Zenghui Zhang, Cong Li, Jiaxuan Deng, Tao Su, Chen Chen, and Yang Zhou. A hybrid-attention semantic segmentation network for remote sensing interpretation in land-use surveillance. *International Journal of Machine Learning and Cybernetics*, 14(2):395–406, 2023. 21

[26] Mamidanna P. Cury K.M. et al Mathis, A. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, page 1281–1289, 2018. 2, 3, 8, 20

[27] Tanmay Nath, Alexander Mathis, An Chi Chen, Amir Patel, Matthias Bethge, and Mackenzie Weygandt Mathis. Using deeplabcut for 3d markerless pose estimation across species and behaviors. *Nature protocols*, 14(7):2152–2176, 2019. 3

[28] Quoc-Viet Nguyen, Woei Leong Chan, and Marco Debiasi. An insect-inspired flapping wing micro air vehicle with double wing clap-fling effects and capability of sustained hovering. In *Bioinspiration, Biomimetics, and Bioreplication 2015*, volume 9429, pages 136–146. SPIE, 2015. 2

[29] Quoc-Viet Nguyen, Woei Leong Chan, and Marco Debiasi. Hybrid design and performance tests of a hovering insect-inspired flapping-wing micro aerial vehicle. *Journal of Bionic Engineering*, 13(2):235–248, 2016. 2

[30] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 3, 4

[31] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos, 2024. 3

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. 2015. 3, 4

[33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 8

[34] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019. 6

[35] Luke Taylor and Geoff Nitschke. Improving deep learning using generic data augmentation, 2017. 6

[36] Simon M Walker, Adrian LR Thomas, and Graham K Taylor. Deformable wing kinematics in free-flying hoverflies. *Journal of the Royal Society Interface*, 7(42):131–142, 2010. 2

[37] Wikipedia contributors. Hausdorff distance — Wikipedia, the free encyclopedia, 2024. [Online; accessed 15-October-2024]. 20

[38] Wikipedia contributors. Jaccard index — Wikipedia, the free encyclopedia, 2024. [Online; accessed 15-October-2024]. 19

[39] Wikipedia contributors. Neural network (machine learning) — Wikipedia, the free encyclopedia, 2024. [Online; accessed 16-October-2024]. 14

[40] Wikipedia contributors. Precision and recall — Wikipedia, the free encyclopedia, 2024. [Online; accessed 15-October-2024]. 19, 20

[41] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9653–9663, June 2022. 3

[42] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014. 3

[43] C. Zwart. Dipterasegmentation. `https://github.com/cornelis1998/DipteraSegmentation`, 2024. 3