Tracking Tiny Insects in Cluttered Natural Environments using Refinable Recurrent Neural Networks

Lars Haalck*, Sebastian Thiele*, Benjamin Risse
Institute for Geoinformatics and Institute for Computer Science
University of Münster, Germany
{lars.haalck, s.thiele, b.risse}@uni-muenster.de

Abstract

Visual tracking of tiny and low-contrast objects such as insects in cluttered natural environments is a very challenging computer vision task. This is particularly true for machine learning algorithms, which usually require distinct visual foreground features to reliably identify the object of interest. Here, we propose a novel deep learning-based tracking framework capable of detecting tiny and visually camouflaged ants (covering only a few pixels) in complex and dynamic high-resolution videos. In particular, we introduce refinable recurrent Hourglass Networks, which combine color and temporal information to continuously detect insects recorded using a freely moving camera. Moreover, this architecture provides comprehensible heatmaps of positional estimations and a seamless integration of optional user-input to further refine the tracking results if necessary. We evaluated our algorithm on an extremely challenging wildlife ant dataset with a resolution of 1024×1024 *and report a mean deviation of 19 pixels from the ground truth (object* ≈ 30 *px) without any user input. By providing only 0.6% manual locations this accuracy can be improved to a mean deviation of 9 pixels. A comparison to a well known deep learning-based single frame detection algorithm (YOLOv7), two state-of-the-art tracking methods (ToMP and KeepTrack), a probabilistic tracking framework and a comprehensive ablation study reveal superior performances in all our experiments. Our tracking framework therefore provides a foundation for challenging tiny single-object tracking scenarios and a practical and interactive solution for biologists and ecologists.*

1. Introduction

Visual object tracking is an important research area in computer vision and has been strongly improved by deep learning techniques [19]. Tracking animals is a prime example for the applicability of these algorithms [7]. While these algorithms perform well for large animals, the localization and temporal association of tiny insects such as ants is however still a challenging task [4] since small appearances aggravate both, the in-frame detection [35] and the tracking [44].

In particular four difficulties can be identified for the detection of very small objects since tiny objects (1) do not provide sufficient visual features; (2) have a very limited visual context; (3) suffer from inappropriate foreground-background ratios; and (4) often lack a sufficient number of positive and diverse training examples [3].

Since most of the existing deep learning-based tracking algorithms use a tracking-by-detection paradigm [28], these algorithms also suffer from the four aforementioned challenges. Additional challenges include (5) potentially more frequent temporal ambiguities (e.g. occlusions; visual clutter) [7]; (6) aggravated temporal associations (especially in a moving camera scenario) [16]; and (7) difficulty to propagate consistent two-dimensional object representations over time [41]. Especially for very small and low-contrast object tracking in relatively large images, misdetections are inevitable so that (8) efficient correction strategies are required if a very high accuracy is needed for behavioral insect studies.

1.1. Contribution

To address these eight challenges and to provide a wildlife insect tracker, we implemented a novel deep learning-based tracking framework which combines spatial color information with temporal motion cues into refinable recurrent Hourglass Networks. Point representations are used to reflect the tiny appearance and absence of visual features or context information and continuous insect localisations are facilitated in an end-to-end joint detection and tracking paradigm. The Hourglass-based feature embeddings provide interpretable heatmaps of potential insect locations and the recurrent network architecture allows the seamless integration of optional user-input to refine the

*These authors contributed equally to this work.

tracking results.

We tested our framework using a challenging single-animal ant dataset (113,548 frames with a 1024×1024 resolution) comprising a foreground/background ratio of 0.0429%, extreme clutter, frequent occlusions, moving camera recordings and a very low foreground contrast with highly ambiguous background appearances (Fig. 1). Our results demonstrate that our framework outperforms several state-of-the-art algorithms reaching a median $L2$-deviation of 5.6 pixels from the ground truth (given an animal size of 30 pixels). In addition, a comprehensive ablation study is used to examine the importance of different input configurations and we study the impact of the processing modules of our architecture with respect to other implementations.

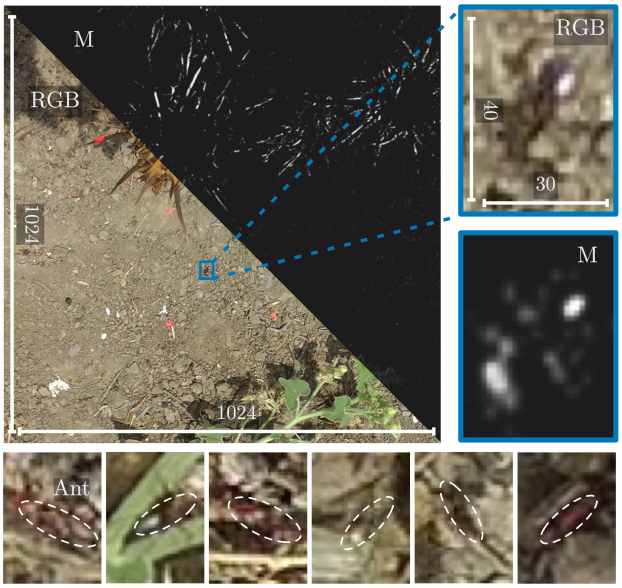

Figure 1. Tracking data overview. Full resolution 1024×1024 RGB image and corresponding motion map (M) are given. Blue 40×30 rectangle and RGB / M blow-up indicate the ant position. Additional 40×30 crops of different ants (dashed ellipses) and backgrounds are given in the bottom row. See Supp. Video S1.

2. Related Work

Due to the above mentioned challenges, the literature on deep learning-based tracking of small objects like ants in complex environments is limited. Instead, the existing work focuses on high-magnification recordings in controlled background conditions resulting in an easy-to-detect insects with a resolution of at least 64×64 pixels [4]. In the regime of tiny object tracking most work focuses on the detection challenges [35], combined work on detection and temporal association however has gained more attention recently [44].

2.1. Tiny Object Detection

The definition of tiny objects vary slightly but usually objects with a relative size smaller than 1% down to 0.1% of the original image are considered as tiny [18, 35] and potential detection strategies include super-resolution techniques [1, 22, 32], data augmentation [5, 17], context-based methods [14, 23], dedicated anchoring mechanisms [9, 39] and loss functions [25, 38] and the learning of multi-scale representations such as feature pyramid networks [10, 13]. Tiny object detection capabilities have also entered the mainstream algorithms. YOLOv7 uses an anchor-based detection strategy [36] and also CornerNet [20], CornerNet lite [21] and CenterNet [42] utilize stacked Hourglass Network backbones with additional loss adjustments. Recently, the first tiny object detection challenge evaluated different algorithms [40] and the winning architecture used a two-stage detector based on a Faster R-CNN with ResNet101 backbone and feature fusion techniques [26]. It should be noted that tiny object detection algorithms are usually developed for very specific use-cases such as face detection (e.g. [1, 14]). Another popular application is to detect small objects on remote sensing data such as aerial (e.g. drone) and satellite imagery [33, 37].

2.2. Tiny Object Tracking

Similar to tiny object detection, tracking visually small objects in relatively large images is still underrepresented in the literature [44]. Due to the small object size point representations are often used as in CenterTrack, which utilizes CenterNet conditioned with two consecutive frames and a heatmap featuring detections from more distant frames [41]. Object center points are then matched by predicting an offset vector from the previous to the current location. For non-stationary camera recordings, a single object tracking algorithm for satellite videos has been proposed [43]. This algorithm is based on a deep Siamese network to compare template regions with search regions in the next frame and incorporates an inter-frame difference centroid inertia motion model to reduce the satellite-induced model drift. In a recent tiny object tracking dataset, a baseline algorithm is introduced [44]. This algorithm is based on a knowledge distillation technique on high resolution data for a teacher network, whereas the student network receives lower resolution images. Both, teacher and student network, share the same network architecture called SuperDiMP, which itself is a combination of DiMP [2] and PrDiMP [6]. ToMP [29] and KeepTrack [30] are discriminative model predictors learning to distinguish the object of interest from the background. The former utilizes a transformer-based model predictor to capture global relations and has been shown to perform well on many benchmark datasets [29]. The latter actively keeps track of distractor objects that have a similar appearance to the object of interest [30]. It uses a learned

association network to match targets and distractors for each frame of a video and was integrated into SuperDiMP. ToMP has been shown to generally outperform KeepTrack, except for extremely challenging tracking scenarios with small, fast moving objects and distractors present. In particular, occlusions in combination with distractors are a common failure case for ToMP [29]. The aim and scope of SuperDiMP, ToMP and KeepTrack and their associated benchmark datasets is to solve generic single object tracking. We, however, are not trying to solve generic tracking, but tracking of task specific tiny objects in extremely challenging data, where generic trackers fail. Therefore the benchmark datasets of the those algorithms are unsuited for our proposed algorithm.

Even though not tuned for tiny object tracking, the ROLO algorithm shares some similarity with our proposed method [31]. ROLO combines object representations based on YOLO with a recurrent neural network (LSTM) to process spatial and temporal information jointly. Apart from the previously discussed deep learning based strategies, there are also attempts to track small objects using probabilistic inference strategies such as factor graphs [34]. Since such algorithms require complete start-to-end image stacks they suffer from slow computational times and are also purely based on motion cues, which aggravates object re-identifications after ambiguous situations such as occlusions. It should be noted that most tracking algorithms use consecutive image pairs and IoU / Hungarian matching methods so that prolonged occlusions, which frequently occur in behavioral insect recordings, cannot be resolved efficiently. Moreover, none of the above mentioned deep learning algorithms enable an efficient integration of additional user input to refine the tracking results in high-uncertainty situations: Only a few manually added locations suffice and smoothly steers the trajectory towards these locations rather than causing abrupt changes along the trajectory.

3. Method

In the following, we first describe our input modalities in Sec. 3.1. Afterwards, we present our tracking methodology which is organized in a modular fashion. These modules are called single frame, recurrent and refinement module and are outlined in Secs. 3.2 to 3.4 respectively. Each module is built on top of the previous module and incrementally contributes to minimizing a shared loss function, which is described in Sec. 3.5. [1]

3.1. Input Configurations

To track tiny ants from continuous potentially high-frame-rate top-down video recordings, both spatial and temporal information can be used. Spatial information is provided from the RGB channels directly (1024×1024 for each channel) and we provide additional temporal cues by calculating pixel-wise differences of consecutive camera motion compensated frames, which we call motion maps. These image differences are similar to the unaries described in [34] and are defined by subtracting successive video frames after rectifying camera motion through the use of pairwise homographies. In contrast to [34], we do not weigh the resulting motion cues by a centered Gaussian. Since these motion maps share the pixel dimensions of the images we can add them as an additional layer to our input resulting in a $1024 \times 1024 \times 4$-dimensional input layer as shown in Fig. 2. The impact of the additional motion maps is studied by providing three different input configurations for our network: RGB (I, 3 channels), motion maps (M, 1 channel) or RGB concatenated with motion maps along the channel axis (I+M, 4 channels).

3.2. Single Frame Module

The single frame module is the most basic module of our proposed tracking framework and results in a single frame object detection algorithm when only RGB information is provided as input. The only way this module has access to temporal information is through motion maps. This module heavily relies on the detection capabilities of its feature extraction backbone. Therefore, we studied two mutually exclusive backbone architectures with different levels of computational complexity to identify the best overall trade-off between time and accuracy.

Hourglass Networks In order to learn the representation of tiny object variations, Feature Pyramid and stacked Hourglass Networks have become very popular since these models can effectively fuse local and global information [40]. In fact, it has been shown that additional local information is particularly useful for object detection given very small appearances [35]. Therefore, we test two different Hourglass backbones, namely CornerNet [20] (HG) and CornerNet-Squeeze [21] (HGS) and evaluate these models with respect to their prediction accuracy and model size.

The lower part of Fig. 2 shows a schematic overview our Hourglass approach. Regardless of the version, the Hourglass Network first computes a representation feature map of size $h_{hgn} \times w_{hgn} \times d_{hgn}$, which corresponds to the output of the last layer with the highest spatial resolution before the Corner Pooling is applied. Using Convolutions and Transposed Convolutions, this feature map is then scaled up and reduced in depth to an easily interpretable single channel heatmap $hm \in \mathbb{R}^{h \times w}$ with the same size as the input frame (see Fig. 5). To determine the position of the object, we calculate the arg max along the height and width of hm.

ResNet50 Since all Feature Pyramid-like networks that first down- and then upsample the spatial resolution come at a

[1] The code is available at https://github.com/LarsHaalck/refinable-rnn.

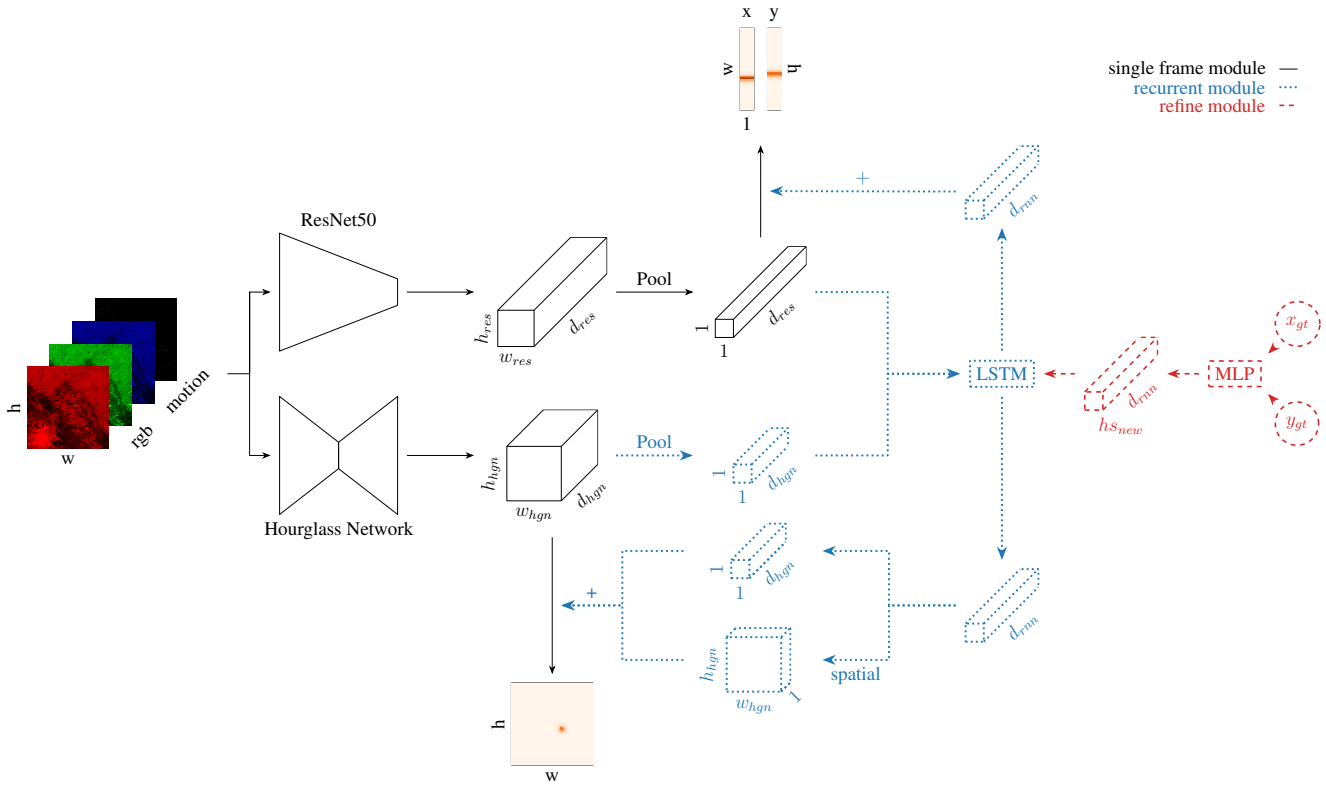

Figure 2. Our method can be separated into three different modules: single frame module, recurrent module and refine module. For single frame detection, we take the (pooled) feature map output of the backbones and compute the position of the object directly. The recurrent module adds temporal memory by appending an LSTM to the spatially pooled feature representations. This LSTM receives the current pooled backbone representation as input and computes a new spatio-temporal representation as its output. After a transformation depending on the model, the representation is added to the (pooled) backbone output and the prediction is created analogously to the single frame module. To refine the tracking predictions, we can incorporate user input (e.g. a starting point or corrections) by replacing the LSTM's hidden state with a new version based on an MLP's representation of a user-provided location (x_{gt}, y_{gt}) for the respective frame.

cost of more computational overhead (resulting in slower tracking speeds), we further reduce this overhead by exploring the performance of a standard ResNet50 as the backbone for our detection and tracking task. As seen in the top part of Fig. 2, a given input $f \in \mathbb{R}^{h \times w}$ is first reduced to a feature map of size $h_{res} \times w_{res} \times d_{res}$ by the ResNet50, that is subsequently reduced by Global Average Pooling to a representation vector of dimension d_{res} which has been shown to be able to preserve localization information [15]. We explore two different ResNet50-based architectures, a regression (RR) and a classification (RC) model. For regression, we directly predict a coordinate (x, y) using a single MLP with the representation vector as input. For classification, we deploy two MLPs that predict $x \in \mathbb{R}^w$ and $y \in \mathbb{R}^h$ respectively. The elements of each vector represent a position along the width and height of the image. An estimation of the object's position is then made by calculating the arg max along x and y independently. A heatmap can be constructed by computing the outer product of the predicted vectors x and y after applying Softmax.

3.3. Recurrent Module

Our recurrent module extends the single frame module in a straight-forward way and enables the incorporation of continuous temporal embeddings. As the dotted blue arrows in Fig. 2 show, the recurrent module is built on top of the features extracted from the respective backbone architecture by appending an LSTM [12] to the output of the pooled feature maps, that are additionally resized with fully connected layers to match the LSTM's input dimensionality d_{rnn} The output of the LSTM is subsequently resized by fully connected layers to match the previous depth of the feature maps produced by the backbone resulting in d_{hgn} for the Hourglass Network and d_{res} for the ResNet50 architecture. In the case of the Hourglass backbone, we add this vector to its output feature map by broadcasting it to each spatial position of the feature map. Alternatively, we explore the possibility of transforming the output of the LSTM into a single channel feature map of size $w_{hgn} \times h_{hgn}$ with an MLP, which is then added to the output of the Hour-

glass backbone. We call this model, depending on the backbone, Hourglass Spatial (HG-S) or Hourglass Squeeze Spatial (HGS-S). If the ResNet50 is used, the vector can be directly added to the pooled feature representation. The final positional prediction for the frame is computed as described in the single frame module (Sec. 3.2).

3.4. Refine Module

Up to this point, our algorithm does not need to rely on any user-specified positional information during inference as long as there is only one dominantly visible object of interest as it is frequently the case in moving-camera recordings. Given the often unavoidable ambiguities in tracking extremely small objects such as ants in their natural habitat, explicit user input might however be required (see Sec. 1) to resolve these ambiguities. For this reason, we introduce the refine module (dashed red arrows in Fig. 2) which receives a supplied coordinate $(x_{gt}, y_{gt}) \in \mathbb{R}^2$ for a given frame as an additional input. This coordinate is transformed by a small MLP into a new hidden state vector hs_{new}, which replaces the current hidden state of the LSTM and therefore incorporates localization information into our algorithm without the need to recompute an entire forward pass of the backbone architecture with modified input. In other words, this module enables to interact with the recurrent neural network by resetting and correcting a (potentially corrupted) state to generate refined trajectories in an interactive and efficient way.

The impact of potential user input is shown in Fig. 3. Let f_0 to f_5 be consecutive frames of a video. Then we can provide the LSTM with an optional initial hidden state h_0 based on the ground truth gt_0 of the first frame f_0, ensuring a correct starting point. Assuming that the next three frames are predicted correctly, the tracker loses the object indicated by erroneous positional estimates in f_4 (i.e. predicted position deviates strongly from the ground truth). With the refine module, a corrected hidden state can be inserted based on ground truth information gt_3 for the previous frame. The trajectory is then refined either by incorporating gt_3 in all succeeding positional estimates (violet arrows) or in a bidirectional fashion (orange arrows). Bidirectional inference is implemented by using an exponential moving average that is stopped after fixed number of frames and by feeding them backwards into the recurrent module starting from the refined frame.

3.5. Loss Functions

There are three different kinds of outputs and therefore learning targets to train and test our models. For predicting the objects coordinate using regression we simply minimized the Mean Squared Error

$$L_{reg} = \|c_{pred} - c_{gt}\|_2^2 \tag{1}$$

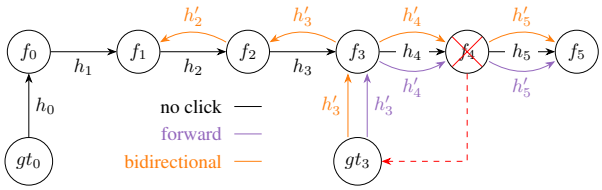

Figure 3. Impact of refine module. Given an erroneous detection at frame four (f_4) a user-specified (x, y)-location gt_3 can be incorporated. The impact on the forward and bidirectional inference refinements are given in violet and orange respectively. Frames are denoted as f_i, hidden states as h_i and corrected hidden states as h'_i.

between the predicted coordinate $c_{pred} = (x_{pred}, y_{pred})$ and the ground truth $c_{gt} = (x_{gt}, y_{gt})$ during training. For the classification tasks, which incorporate the two vector output of RC and the heatmap outputs from the HG networks, we minimize the focal loss variant proposed in [42]. For the 1D case of two separate vectors for the x and y position, we use an unnormalized 1D Gaussian

$$e^{-\frac{(x-x_{gt})^2}{2\sigma^2}} \quad \text{and} \quad e^{-\frac{(y-y_{gt})^2}{2\sigma^2}} \tag{2}$$

centered around the ground truth coordinate elements (x_{gt}, y_{gt}). Analogously we use a full 2D Gaussian for the heatmap generation in the HG architectures

$$e^{-\frac{(x-x_{gt})^2+(y-y_{gt})^2}{2\sigma^2}}. \tag{3}$$

The classification loss is defined as

$$L_{cls} = -\sum_{i=1}^{N} \begin{cases} (1-p_i)^2 \log(p_i) & , l_i = 1 \\ (1-l_i)^4 (p_i)^2 \log(1-p_i) & , l_i \neq 1 \end{cases} \tag{4}$$

where N is the number of spatial positions in the flattened prediction and l_i the ground truth label at flattened position i. The Softmax operation without temperature scaling is used along the spatial positions to convert the network's outputs at flattened position c_i into probabilities p_i.

4. Results

Dataset We perform all our experiments on dataset of 113,548 frames from different videos recorded using a hand-held camera with 60 frames per second and a resolution of 1024×1024 pixels. The dataset is a subset of the ontogeny dataset [11]. Each frame has a ground truth label of (x, y) coordinates in pixels for the exact ant position. The training dataset consists of 53,877 frames, while the validation dataset contains 8,883 frames. The remaining 50,788 frames are used as a test split.

Notation In the following paragraphs, we use short notations for different input configurations, architecture combinations, inference modalities (i.e. recurrent or single) and

refinement settings. The notations is derived from the configurations and modules described in Sec. 3: Images (I), Motion Maps (M), Images and Motion Maps (I+M), Hourglass (HG), Hourglass Spatial (HG-S), Hourglass Squeeze (HGS), Hourglass Squeeze Spatial (HGS-S), ResNet classification (RC), ResNet regression (RR), recurrent inference (Rec.) and single frame inference (Single). Hourglass Spatial (HG-S) was not used due to the high amount of training parameters.

Technical Details Our architecture described in Sec. 3 is trained in two stages. In the first stage, we train the single frame module, containing the backbone pretrained on ImageNet [8] combined with a loss as described in Sec. 3.5. We train the network using a batch size of 12, a learning rate of $1e{-}4$ using AdamW [27] and $\sigma = 3$ for the unnormalized Gaussian. For the second stage, we train the recurrent part of our architecture on video sequences with a length of 32, keeping the backbone fixed, while allowing parameter refinements of the predictor. Training the modules separately and keeping the single frame module fixed for the recurrent module, facilitates the training on longer sequences in contrast to training end-to-end. For the Hourglass architectures, we use a batch size of 4, while ResNet based architectures use a batch size of 12. All other parameters are unchanged. In both stages, the only data augmentations are independent random zeroing of image and motion maps input and random $90°$ rotations.

4.1. Comparison to State-of-the-Art

We compare our tracker with four different state-of-the-art methods: a probabilistic factor graph optimization based tracker described in [34] that was specifically designed for a similar dataset, YOLOv7 [36] in its biggest variant YOLOv7-E6E with 151.7M parameters as a framework for single-frame detection, ToMP [29] in its variant ToMP50 and KeepTrack as specified in [30]. The factor graph method uses only motion similar to our motion maps called unaries (see Sec. 3.1) as its input, while YOLOv7 uses only RGB images. To provide better comparability, we extend the YOLOv7 pipeline to process a stacked 4D input consisting of images and motion maps as in our method (see Sec. 3). YOLOv7 was pretrained using COCO [24] and trained further using our training dataset until convergence. Ground truth animal positions were transformed into bounding boxes, by using the position as the bounding box center with a fixed height and width throughout all frames. To infer only a single bounding box, non-maximum suppression was disabled and the bounding box with the highest confidence extracted in each time-step without linking detections over time. For ToMP and KeepTrack the fully trained model was used as-is without retraining, since both methods are generic trackers that generally do not require training on the current dataset. The first ground-truth position is supplied to the architecture and for each frame the center of the bounding box is extracted as the inferred position. The results are shown in Tab. 1.

	median [px]	mad [px]
I+M, HGS-S	**5.6**	**2.4**
I+M, HGS	5.8	2.7
I+M, YOLOv7 [36]	17.8	8.2
I, YOLOv7 [36]	26.9	13.5
M, Factor-Graph [34]	12.2	8.1
I, ToMP [29]	895.2	143.3
I, KeepTrack [30]	713.3	208.4

Table 1. Comparison of our architecture with YOLOv7 [36], the factor graph based method [34] ToMP [29] and KeepTrack [30]. Results are measured in median and median of absolute deviations given the L2-deviation in pixels from the ground truth.

Given the full input of images and motion maps, our tracker outperforms YOLOv7 in both input modalities as well as the factorgraph-based method, ToMP and KeepTrack. YOLOv7 being a very optimized architecture achieves an inference time of around 50ms per image on a consumer grade graphics card (NVIDIA RTX 3090) in comparison to our architecture with an inference time of around 55ms for HGS and 140ms for HGS-S. Although using a very optimized parallelized version of the factor graph method, it takes around 200ms per image on an AMD Ryzen Threadripper PRO 3995WX due to the need to solve a combined optimization problem for all frames. The authors also introduce a way to incorporate manual corrections into this optimization problem but in contrast to our architecture, the changes always affect the full solution leading to delayed updates of detections especially for longer videos. The high deviation from the ground-truth in ToMP and KeepTrack are caused by frequent and sometimes long-lasting losses of the object of interest. In fact both trackers often fixated on other moving objects such as plants, which could be caused by the very small size and low contrast of the object if interest.

4.2. Ablation Study

To evaluate each part of our architecture, we tested our pipeline against different input modalities, backbone choices and compared the performance from the recurrent module of our pipeline with its single frame detection counterpart. To show the effects of different choices on different backbones, we select two reference architectures and input modalities: HGS-S and RC both with images and motion maps as their input. The quantitative results are shown in Tab. 2.

In the first experiment, different backbones were tested, showing that the Hourglass Squeeze backbone performed

	Best	Different Arch.				Different Input				Single vs Recurrent	
Input	I+M	I+M	I+M	I+M	I+M	M	I	M	I	I+M	I+M
Architecture	HGS-S	HG	HGS	RC	RR	HGS-S	HGS-S	RC	RC	HGS-S	RC
Single/Rec.	Rec.	Rec.	Rec.	Rec.	Rec.	Rec.	Rec.	Rec.	Rec.	Single	Single
med [px]	5.6	6.0	5.8	12.0	23.9	5.8	6.3	29.0	27.4	7.3	15.5
mad [px]	2.4	2.9	2.7	6.2	12.1	2.8	3.3	21.9	15.4	3.3	7.9

Table 2. Comparison of different input modalities, backbones and inference modes (single vs recurrent) are shown. Columns are color coded for Hourglass Squeeze Spatial (HGS-S) in blue, as well as ResNet Classification (RC) in red for better visibility. The column marked 'Best' identifies the best performing architecture. Results are measured in median and median of absolute deviations given the L2-deviation in pixels from the ground truth.

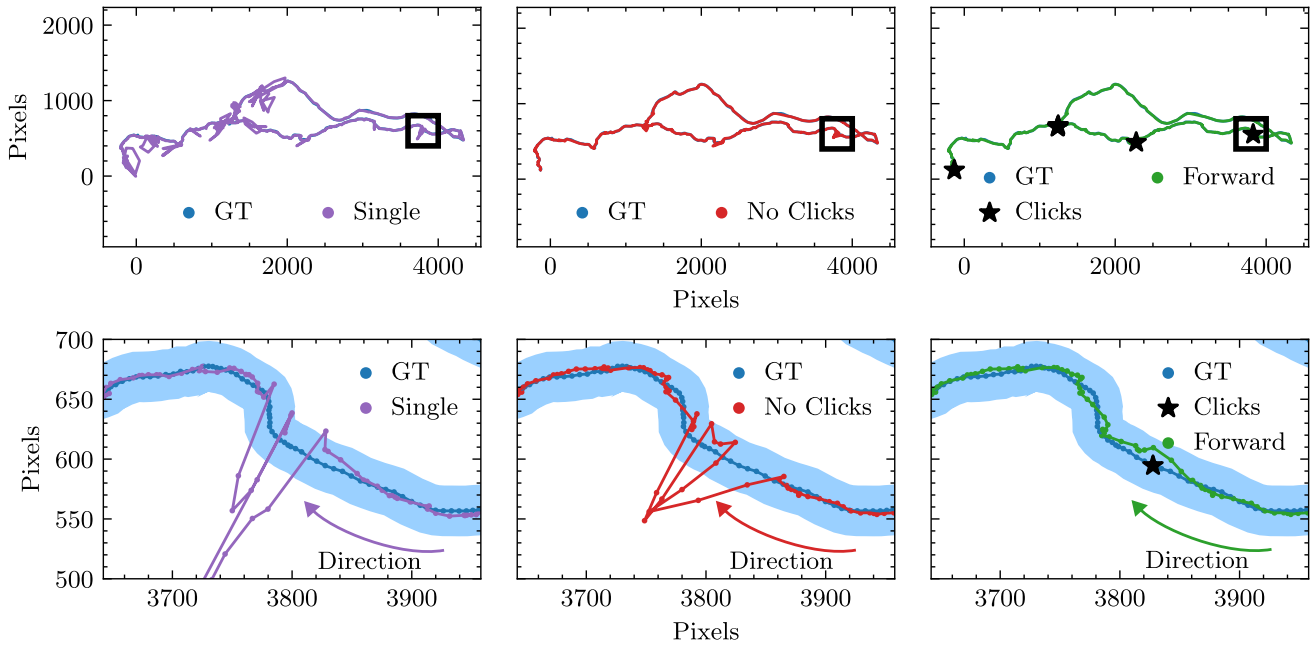

Figure 4. Performance comparison of single frame inference (left column; purple [■]), recurrent inference without refinement (middle column; red [■]), and recurrent inference with refinement (right column; green [■]). The top row depicts a complete trajectory of a tracked object and the bottom row a zoomed section outlined by the black rectangle. Ground-truth trajectories are shown in blue [■] (Zoom in for top row). The shaded area defines a constant error band around the ground truth containing all detections of the recurrent inference with refinement.

best. Evaluating different input modalities, revealed that all architectures benefited from a combination of images and motion maps. Interestingly, supplying only motion maps to HGS-S delivered better results than providing only images. Finally, we tested the performance gain from using the recurrent part of our architecture, showing improvement in all tested architectures.

Qualitatively, these improvements are illustrated in Fig. 4. Single frame inference is close to the ground truth most of the time, with many outliers in more complex environments. In contrast, recurrent inference without any manual corrections reduces the amount of outliers significantly. A single refinement in form of a supplied location is then able to remove remaining inaccuracies by resetting a faulty hidden state impacting successive frames. Note that the manual correction is intentionally not integrated as a hard constraint for the new inference. Since results from the LSTM are added and not replaced (see Fig. 2), the supplied correction is able to guide the model towards the ground-truth without strictly enforcing it, resulting in a smooth integration rather than a potentially abrupt positional change over time.

To study the impact of refinements in more detail, we compare the heatmaps before the final prediction of single inference, recurrent inference with refinements and recurrent inference without refinements in a challenging scenario

as shown in Fig. 5. Heatmaps of the single frame inference include strong noise which aggravates accurate predictions. In contrast, heatmaps of the recurrent inference without refinements, are much more directed towards the actual object while still maintaining a higher amount of uncertainty. Supplying a single manual location in time frame t_i and no additional refinements in the following frames, reduces the uncertainty significantly and allows following frames to be predicted more precisely.

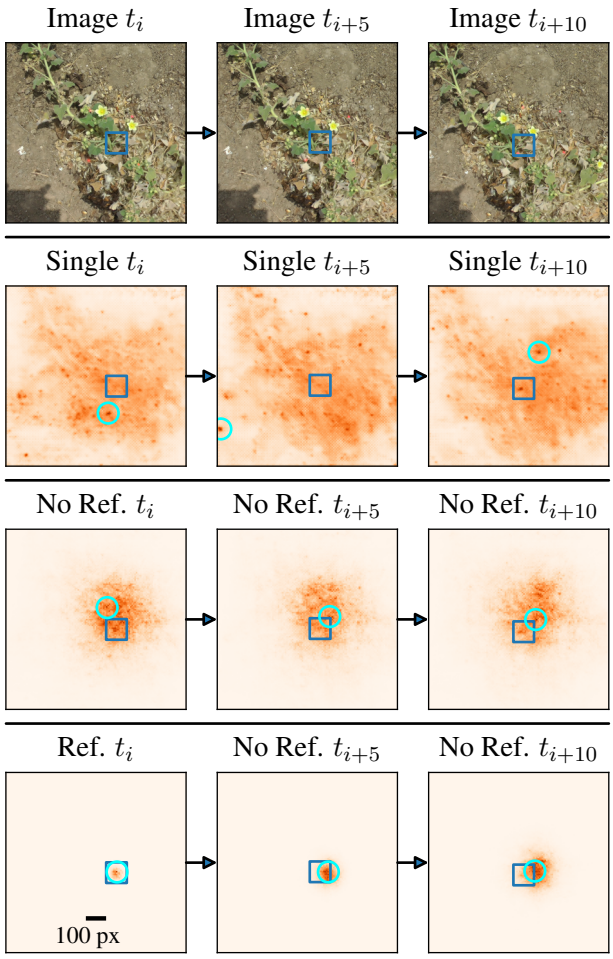

Figure 5. Heatmap visualization of a challenging scene for three timestamps comparing single inference, recurrent inference with refinements and recurrent inference without refinements. Ground-truth positions are shown in boxes in dark blue [■] and inferred positions in circles in light blue [■]. See Supp. Video S2 for a full video.

Across all our experiments we observed that refinements aid the reduction of outliers in the recurrent module which however does not affect the median in most circumstances. Analyzing the mean $L2$-deviation from the ground truth of HGS-S with images and motion maps as its input, we observe a mean of 19 pixels, which can be reduced to 8.7 pixels with 0.64% of refined frames selected by a maximal allowed deviation threshold of 60 pixels from the ground truth. When using a bidirectional inference, where we also allow corrections backwards from a refined frame, this mean reduces to 8.4.

5. Conclusion

In this paper, we proposed a novel tracking framework for tiny insects in natural environments. Our solution outperforms two state-of-the-art algorithms on a challenging dataset. A fast integration of manual corrections combined with the direct and interpretable feedback in the form of heatmaps, make our solution a valuable system for a variety of animal behavior studies. We tested our framework on an extremely challenging ant dataset and demonstrate that the combination of Hourglass Networks with LSTMs outcompete all other tested architectures.

The modular design of our system also enables a variety of straight-forward extensions. For example, the resultant heatmaps could also be used to guide a user to potential frames with high uncertainty so that the refine module could be used in a targeted fashion. Moreover, the Hourglass architectures are well-suited to be extended for multiple animals with a small change of the inference modality. Instead of an arg max predicting only one position, multiple positions could be extracted from the heatmaps. Since we focused on moving camera scenarios, in which the camera is constantly kept over a moving object of interest, the scope of this paper was on single objects over time.

In the future, we will explore these possibilities and address some open challenges such as further optimized bidirectional inference. Specifically, this type of inference could be incorporated into the training to make the model more robust. Moreover, different weighting strategies can be evaluated to further increase the performance of bidirectional inference.

6. Acknowledgments

BR, ST and LH would like to thank the Human Frontier Science Program `[RGP0057/2021]`. BR would like to thank the *Ministerium für Kultur und Wissenschaft des Landes Nordrhein-Westfalen* for the AI Starter support `(ID 005-2010-0055; KI Starter)` and the WildDrone MSCA Doctoral Network funded by EU Horizon Europe under grant agreement `#101071224`. LH also acknowledges support by the Heinrich Böll Foundation.

References

[1] Yancheng Bai, Yongqiang Zhang, Mingli Ding, and Bernard Ghanem. Finding tiny faces in the wild with generative adversarial network. In *Proceedings of the IEEE conference on*

computer vision and pattern recognition, pages 21–30, 2018. 2

[2] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6182–6191, 2019. 2

[3] Brais Bosquet, Manuel Mucientes, and Victor M Brea. Stdnet: Exploiting high resolution feature maps for small object detection. *Engineering Applications of Artificial Intelligence*, 91:103615, 2020. 1

[4] Xiaoyan Cao, Shihui Guo, Juncong Lin, Wenshu Zhang, and Minghong Liao. Online tracking of ants based on deep association metrics: method, dataset and evaluation. *Pattern Recognition*, 103:107233, 2020. 1, 2

[5] Changrui Chen, Yu Zhang, Qingxuan Lv, Shuo Wei, Xiaorui Wang, Xin Sun, and Junyu Dong. Rrnet: A hybrid detector for object detection in drone-captured images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2

[6] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7183–7192, 2020. 2

[7] Anthony I Dell, John A Bender, Kristin Branson, Iain D Couzin, Gonzalo G de Polavieja, Lucas P J J Noldus, Alfonso Pérez-Escudero, Pietro Perona, Andrew D Straw, Martin Wikelski, and Ulrich Brose. Automated image-based tracking and its application in ecology. *Trends in Ecology & Evolution*, 29(7):417 – 428, 07 2014. 1

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *CVPR*, pages 248–255, 2009. 6

[9] Christian Eggert, Dan Zecha, Stephan Brehm, and Rainer Lienhart. Improving small object proposals for company logo detection. In *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, pages 167–174, 2017. 2

[10] Yuqi Gong, Xuehui Yu, Yao Ding, Xiaoke Peng, Jian Zhao, and Zhenjun Han. Effective fusion factor in fpn for tiny object detection. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1160–1168, 2021. 2

[11] Lars Haalck, Michael Mangan, Antoine Wystrach, Leo Clement, Barbara Webb, and Benjamin Risse. Cater: Combined animal tracking & environment reconstruction. *Science Advances*, 9(16):eadg2094, 2023. 5

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. 4

[13] Mingbo Hong, Shuiwang Li, Yuchao Yang, Feiyu Zhu, Qijun Zhao, and Li Lu. Sspnet: Scale selection pyramid network for tiny person detection from uav images. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2021. 2

[14] Peiyun Hu and Deva Ramanan. Finding tiny faces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 951–959, 2017. 2

[15] Md Amirul Islam, Matthew Kowal, Sen Jia, Konstantinos G. Derpanis, and Neil D. B. Bruce. Global pooling, more than meets the eye: Position information is encoded channel-wise in CNNs. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2021. 4

[16] Shivani Kapania, Dharmender Saini, Sachin Goyal, Narina Thakur, Rachna Jain, and Preeti Nagrath. Multi object tracking with uavs using deep sort and yolov3 retinanet detection framework. In *Proceedings of the 1st ACM Workshop on Autonomous and Intelligent Mobile Systems*, pages 1–6, 2020. 1

[17] Mate Kisantal, Zbigniew Wojna, Jakub Murawski, Jacek Naruniec, and Kyunghyun Cho. Augmentation for small object detection. *arXiv preprint arXiv:1902.07296*, 2019. 2

[18] Harish Krishna and CV Jawahar. Improving small object detection. In *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 340–345. IEEE, 2017. 2

[19] Matej Kristan, Jiří Matas, Aleš Leonardis, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Hyung Jin Chang, Martin Danelljan, Luka Cehovin, Alan Lukežič, et al. The ninth visual object tracking vot2021 challenge results. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2711–2738, 2021. 1

[20] Hei Law and Jia Deng. CornerNet: Detecting objects as paired keypoints. *ECCV*, 2016. 2, 3

[21] Hei Law, Yun Teng, Olga Russakovsky, and Jia Deng. Cornernet-lite: Efficient keypoint based object detection. *CoRR*, abs/1904.08900, 2019. 2, 3

[22] Jianan Li, Xiaodan Liang, Yunchao Wei, Tingfa Xu, Jiashi Feng, and Shuicheng Yan. Perceptual generative adversarial networks for small object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1222–1230, 2017. 2

[23] Jeong-Seon Lim, Marcella Astrid, Hyun-Jin Yoon, and Seung-Ik Lee. Small object detection using context and attention. In *2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 181–186. IEEE, 2021. 2

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. *ECCV*, 2014. 6

[25] Gen Liu, Jin Han, and Wenzhong Rong. Feedback-driven loss function for small object detection. *Image and Vision Computing*, 111:104197, 2021. 2

[26] Jingwei Liu, Yi Gu, Shumin Han, Zhibin Zhang, Jiafeng Guo, and Xueqi Cheng. Feature rescaling and fusion for tiny object detection. *IEEE Access*, 9:62946–62955, 2021. 2

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ICLR*, 2019. 6

[28] Seyed Mojtaba Marvasti-Zadeh, Li Cheng, Hossein Ghanei-Yakhdan, and Shohreh Kasaei. Deep learning for visual tracking: A comprehensive survey. *IEEE Transactions on Intelligent Transportation Systems*, 2021. 1

[29] Christoph Mayer, Martin Danelljan, Goutam Bhat, Matthieu Paul, Danda Pani Paudel, Fisher Yu, and Luc Van Gool. Transforming model prediction for tracking. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022. 2, 3, 6

[30] Christoph Mayer, Martin Danelljan, Danda Pani Paudel, and Luc Van Gool. Learning target candidate association to keep track of what not to track. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2021. 2, 6

[31] Guanghan Ning, Zhi Zhang, Chen Huang, Xiaobo Ren, Haohong Wang, Canhui Cai, and Zhihai He. Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking. *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, 2017. 3

[32] Junhyug Noh, Wonho Bae, Wonhee Lee, Jinhwan Seo, and Gunhee Kim. Better to follow, follow to be better: Towards precise supervision of feature super-resolution for small object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9725–9734, 2019. 2

[33] Jiangmiao Pang, Cong Li, Jianping Shi, Zhihai Xu, and Huajun Feng. R^2-CNN: Fast Tiny Object Detection in Large-Scale Remote Sensing Images. *IEEE Transactions on Geoscience and Remote Sensing*, 57(8):5512–5524, 2019. 2

[34] Benjamin Risse, Michael Mangan, Luca Del Pero, and Barbara Webb. Visual Tracking of Small Animals in Cluttered Natural Environments Using a Freely Moving Camera. *ICCVW*, pages 2840–2849, 2017. 3, 6

[35] Kang Tong and Yiquan Wu. Deep learning-based detection from the perspective of small or tiny objects: A survey. *Image and Vision Computing*, page 104471, 2022. 1, 2, 3

[36] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv*, 2022. 2, 6

[37] Jinwang Wang, Wen Yang, Haowen Guo, Ruixiang Zhang, and Gui-Song Xia. Tiny Object Detection in Aerial Images. *2020 25th International Conference on Pattern Recognition (ICPR)*, 00, 2021. 2

[38] Zijie Wang, Jianwu Fang, Jian Dou, and Jianru Xue. Small object detection on road by embedding focal-area loss. In *International Conference on Image and Graphics*, pages 657–665. Springer, 2019. 2

[39] Ze Yang, Shaohui Liu, Han Hu, Liwei Wang, and Stephen Lin. Reppoints: Point set representation for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9657–9666, 2019. 2

[40] Xuehui Yu, Zhenjun Han, Yuqi Gong, Nan Jan, Jian Zhao, Qixiang Ye, Jie Chen, Yuan Feng, Bin Zhang, Xiaodi Wang, et al. The 1st tiny object detection challenge: Methods and results. In *European Conference on Computer Vision*, pages 315–323. Springer, 2020. 2, 3

[41] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*, pages 474–490. Springer, 2020. 1, 2

[42] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as Points. *arXiv*, 2019. 2, 5

[43] Kun Zhu, Xiaodong Zhang, Guanzhou Chen, Xiaoliang Tan, Puyun Liao, Hongyu Wu, Xiujuan Cui, Yinan Zuo, and Zhiyong Lv. Single Object Tracking in Satellite Videos: Deep Siamese Network Incorporating an Interframe Difference Centroid Inertia Motion Model. *Remote Sensing*, 13(7):1298, 2021. 2

[44] Yabin Zhu, Chenglong Li, Yao Liu, Xiao Wang, Jin Tang, Bin Luo, and Zhixiang Huang. Tiny Object Tracking: A Large-scale Dataset and A Baseline. *arXiv*, 2022. 1, 2