USING OCCLUSION TO TRACK MULTIPLE OBJECTS WITH SIMILAR APPEARANCE

by

Thomas Fasciano

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2015

Approved by:

Dr. Min C. Shin

Dr. Richard Souvenir

Dr. Shaoting Zhang

Dr. Jianping Fan

Dr. Stan Schneider

ABSTRACT

THOMAS FASCIANO. Using occlusion to track multiple objects with similar appearance. (Under the direction of DR. MIN C. SHIN)

Multiple object tracking is a fundamental problem within computer vision and has a wide range of applications. Although well studied, it remains a difficult task especially in scenarios which contain many occluding and highly similar appearing objects such as in videos of social insects. Data association based tracking methods have recently become popular due to improvements in object detection methods. Rather than tracking sequentially, detections found independently in each frame are first associated in short trackings or tracklets across adjacent frames which are further associated into longer trackings. A key component of this tracking method is the affinity model which measures the likelihood that two tracklets belong to the same object and incorporates appearance, motion and temporal information. First, we propose to improve the affinity model within insect videos by introducing a new set of irregular motion features. Second, we propose a method for filtering nearby confusing associations by using a sequence of foreground blobs called Occlusion Sub-Tunnels. Finally, we propose a method for online learning of the full affinity model by automatically generating a set of weakly labeled samples using connectivity within occlusion sub-tunnels.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Shin for all of his guidance, wisdom and (most importantly) patience during my PhD studies. I would also like to thank my lab mates, especially Scott Spurlock, Hui Wu, Rich Nguyen, and Lance Rice for all of their support and for always being available to lend an ear for discussing crazy ideas. Many thanks to the support of my family for encouraging me to follow through with this endeavor. Finally, a thank you to Danielle Cobos for always providing encouragement and strength during the hardest times that allowed me to continue moving forward.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Multi-object tracking is a fundamental problem within computer vision and has wide ranging importance for applications [64, 16]. In addition, tracking is often a vital first step to many other applications such as action recognition [51] and automated surveillance [64]. Although tracking has been well studied, tracking multiple objects simultaneously within videos is still a challenging task. This problem becomes particularly challenging in scenarios that include many similar appearing objects which frequently occlude.

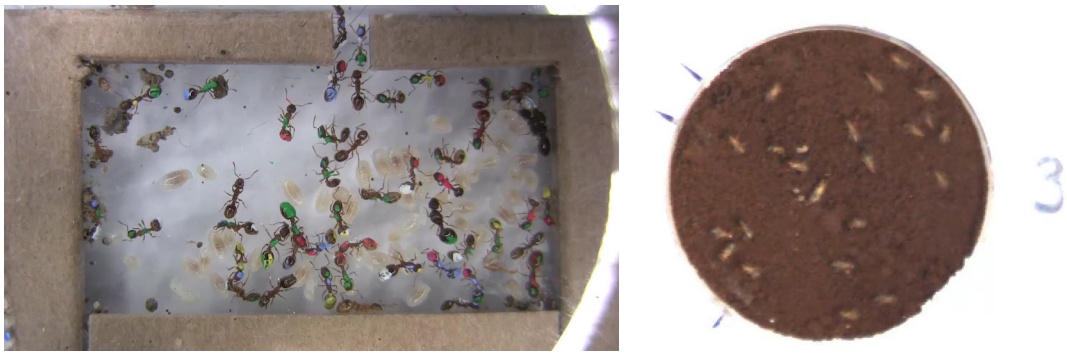
Many frameworks exists for multi-object tracking [16]. Although sequential tracking methods, such as particle filter based trackers, have traditionally been the frameworks of choice, data association based tracking methods have recently become popular. This is primarily due to two main reasons 1) object detection methods, which these methods rely on, have been continually improving [19, 54] and 2) their use of global information while solving over instances of occlusion. Rather than tracking frames sequentially, data association based trackers begin by running an object detector independently in each frame of the video. Detection in adjacent frames are then associated together into *tracklets* (or short trackings). Tracklets which belong to the same object are then iteratively associated together to form longer tracks.

An integral part of the association process is the affinity model which measures the likelihood that two tracklets belong to the same object. Typically the affinity

model is a combination of appearance, motion, and temporal measures comparing the tracklets [27]. Machine learning has been applied to learn more robust combinations of features offline [36] as well as learning more sophisticated appearance features [50, 32]. However, offline trained models and sophisticated features sets often do not transfer well to new unseen videos. For example, the visual queues which best distinguish between targets may change between datasets and may result in either poor tracking performance or require the collection of large amounts of labeled training data to retrain an appearance model. Methods for online training appearance models have been proposed to alleviate this problem and has been shown to greatly improve the performance of tracking algorithms in videos where appearance is a discriminating feature between targets, such as pedestrians [32, 33, 61].

However, there exist many domains where appearance between objects is highly similar. For example, multi-object tracking has been performed on sports videos such as soccer games [4] and basketball games [15] which contain many people wearing identical clothing. This causes appearance to be less reliable, especially during occlusion. A frequent solution within sports videos is to use multiple camera views simultaneously to disambiguate targets based on positional information [52, 4]. However, using multiple cameras isn't always an option in many domains such as when tracking biological objects.

Studying biological objects typically involves observing a large number of objects within a confined space [11]. Typically, researchers will gather their observations by recording videos which must be then analyzed which is traditionally a manual process. As such, automated tracking algorithms are quickly becoming vital tools to the study



(a) (b)

Figure 1: Example frames from videos representing my target video domain. (a) Paint marked ants which undergo frequent, long occlusions and share similar appearances. (b) Termites which show irregular motion and are identical in appearance.

of many biological objects including cells [7] and social insects such as ants [40] and bees [14].

In this dissertation, I focus on the application to social insects, specifically ants and termites, which are highly accessible models for studying complex systems [48, 20]. Like in sports videos, social insect videos contain a large number of highly similar (if not identical) objects as shown in Figure 1. In addition, they often move highly irregularly and occlude frequently due to their social behavior for long periods of time where detections are often unreliable or missing entirely. This causes traditional linear motion models to be unreliable during the association process. Additionally, like appearance in pedestrian videos, motion behaviors and patterns can often vary between experiments and datasets causing prior trained motion models to be unreliable thus requiring a large amount of manually labeled training data to train new models.

In this dissertation, I propose three contributions to multiple object tracking in

scenarios with irregular moving and similar appearing objects such as ants and termites. First, a new irregular motion feature based on random walks which is better suited to predicting the motion patterns of targets such as ants and termites over long occlusions. Second, leverage occlusion directly to help filter impossible associations between nearby objects which are not occluding, to reduce tracking errors. Finally, by leveraging the ability to filter incorrect associations based on occlusions, provide a method to online learn motion affinity models in addition to appearance models to improve tracking performance on new unseen datasets.

In the next chapter, I review related work to this dissertation. In Chapter 3, I review the hierarchical data association based tracking method which my work builds upon. In the following chapters, I discuss and evaluate each of my three stated contributions. Finally, I conclude in Chapter 7 and discuss any possible future work derived from this dissertation.

CHAPTER 2: RELATED WORKS

Multi-object tracking is a well studied area within computer vision [64]. For the purposes of this dissertation, we view most tracking frameworks as fitting into two broad categories: 1) sequential (or "real time") methods and 2) global methods. The goal of the sequential tracker is to obtain the target's location by combining a prior state model based on position, velocity and appearance information with a potentially noisy observation in the current frame [64, 3]. For multi-target tracking, many methods have been proposed to handle noisy observations such as Multi-Hypothesis Tracking [43] and Joint Probabilistic Data Association Filters [22] which maintain multiple hypothesis until enough evidence is collected to resolve ambiguities. However, these methods are unable to handle periods of missing observations. Kalman Filters and more recently Extended Kalman Filters, have been applied to estimate the target using a Gaussian estimated state model [8, 25, 45, 56]. Particle Filter methods are another popular approach which, rather than relying on detectors for its current state estimate, maintains each object's current state as a set of weighted particles. Every particle is updated at each frame based on prior motion and re-weighted using a scoring function often based on appearance [28, 41]. To improve identity maintenance during tracking, particle filters have been extended to measure the entire state space (current state of all objects) and use Markov chain Monte Carlo (MCMC) sampling to approximate the solution [21, 18, 30]. However, particle filters and Kalman filters

are prone to drifting during long periods of occlusion (which are frequent in insect videos), after which they rarely recover. For example in ant videos, reliable tracking may only last up to 1,000 frames (approximately 30 seconds of video) which then must be manually corrected [21, 42].

Global methods solve this drifting problem by incorporating information from the entire video to solve the tracking problem. Although no longer suitable for real time tracking, these methods often have lower tracking errors which is a reasonable trade off for many tracking applications [16]. To handle noisy observations, [35] coupled the detection and tracking problem by Quadratic Boolean Programming which is solved by an expectation-maximization (EM) style algorithm. Due to its combinatorial growth of the hypothesis search space, this method is limited to short videos.

Object detection methods have greatly improved in recent years [19, 53, 54, 57, 26, 13]. While still not perfect, this has lead to data association based tracking (DAT) becoming more feasible. In DAT approaches, detections are first obtained independently in all frames. These detections are then associated together across adjacent frames to form *tracklets* or short trackings. Finally, the tracklets are then associated together to form the final trackings. Because tracklets contain more than one detection, this allows the use of higher order motion and appearance features as well as past and future information when associating tracklets together. In addition to solving the association problem, these DAT methods must also account for imperfect detections (missing detections, noisy detections, and false alarms) and initialization and termination of trackings.

Many frameworks have been proposed for solving the DAT problem. Some frame-

works solve each object's tracking independently through a greedy approach such as iteratively solving for minimum cycles [46] or k-shortest paths [5]. However, this greedy approach can often lead to tracking errors in cases of similar appearing features. Rather than solving each tracking independently, global optimization approaches solve all associations simultaneously. In [2], the global optimization task was framed as an energy minimization problem. Another approach is to formulate the global association problem as a linear assignment problem [58] which could be solved using the Hungarian algorithm [31]. This formulation was further expanded to account for false alarms, initialization and termination of tracklets within a maximum-a-posteriori (MAP) problem [27]. The MAP formulation can be efficiently solved using the Hungarian algorithm [27], network flows [65] or linear programming [7].

A key component of the MAP formulation is the creation of the affinity score which measures the likelihood that two tracklets belong to the same object. Initially, the affinity score was a manually tuned combination of basic appearance (RGB histograms), motion (motion smoothness), and temporal features [27, 65]. To improve the robustness of the appearance modeling, offline trained appearance models composed of histogram of gradient (HoG), covariance, and rgb histogram features have been proposed [33, 50]. This was extended by [36] to include motion and temporal features to train all parts of the affinity model which included appearance, temporal and motion features using a hybrid classification and ranking machine learning approach called HybridBoost. To improve the robustness of appearance models between different datasets, methods which online train appearance models models have been proposed [32, 33, 62]. These methods train a detection level classifier which deter-

mines if two detections belong to the same object. Training samples are generated online through the observation that tracklets which overlap temporally cannot belong to the same object. This idea has been extended to train individual classifiers for each object in the video by using multiple instance learning [61] or by identifying globally unique appearance features [4]. However, these methods still rely on manually tuned linear motion models as part of their full affinity measure because the none of the methods for generating labels for detection pairs can be used for generating labels of the tracklet association pairs required for training a motion affinity model.

Even the most advanced affinity models are prone to failure in cases of highly similar appearing and moving objects. To further reduce ambiguity during the association process, recent methods have accounted for association dependencies, or how taking one association affect taking a second association. This task has been formulated in a number of ways including conditional random fields [60, 63], rank-1 tensor approximations [49], and by a Lagrangian relaxation to minimum-cost network flows [9]. However, these problems typically grow exponentially with the number of targets and must employ strict motion heuristics [60] or temporal constraints [49, 9] to reduce the problem size which precludes their use when videos contain long occlusions.

CHAPTER 3: HIERARCHICAL DATA ASSOCIATION BASED TRACKING

The data association based tracking framework can be broken down into three steps: 1) detection, 2) tracklet building, and 3) tracklet matching (or association) as shown in Figure 2. First, detection is performed independently at each frame of the video. The detection method used is independent of the overall tracking framework and is typically unique for each type of target. Detections found in all frames create the initial set of detections, $\mathcal{D}$.

Next, detections in adjacent frames are linked together to form an initial set of tracklets during the tracklet building step. To ensure there are no errors (*e.g.,* identity switches) within the initial set of tracklets, associations are performed conservatively using a double threshold technique. For each pair of adjacent frames, an association probability matrix, $\mathbf{S}$, measuring the similarity between two detections is computed where rows of $\mathbf{S}$ correspond to detections in frame f and columns to detections in frame $f+1$. The similarity measure, $\mathbf{S}(i,j)$, is the product of appearance, size, and position similarity measures between the detections. Detection pairs, $\mathbf{S}(i,j)$, whose similarity meet a minimum threshold, $\mathbf{S}(i,j) > \theta_1$, and are also θ_2 greater than any other value in row i and column j in $\mathbf{S}$ are linked together to form a tracklet T. The set of generated tracklets, as well as unassociated single detection tracklets, create the initial set of tracklets $\mathcal{T}^0$.

Finally, the initial set of short confident tracklets are iteratively associated (or

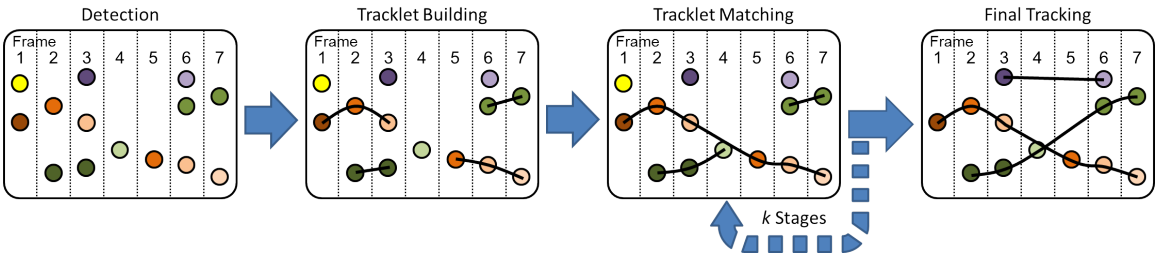


Figure 2: The data association based framework of [27]. This framework is broken into three steps: (1) Detections are found independently in each frame, (2) detections in adjacent frames are conservatively linked together to form short, confident *tracklets*, and (3) the tracklets are iteratively linked together over k rounds of matching to form longer and longer trackings.

matched) into longer and longer tracklets. Huang *et. al.* formulated this matching step as a MAP problem that takes into consideration the initialization and termination of full trackings, associations between pairs of tracklets, and likelihood of a tracklet being a false alarm that should be discarded. The matching step is performed iteratively in stages, k. At each stage k, the output from the previous stage, $\mathcal{T}^{k-1}$ becomes the input for the current stage of matching where associations of up to τ_k frames apart are considered. Each tracklet, $T_i^{k-1} \in \mathcal{T}^{k-1}$, is (1) matched to another tracklet, (2) labeled as the start or end of an object trajectory and therefore not associated to another tracklet, or (3) classified as a false alarm and ignored from further matching. The final association problem is formulated as follows:

$$\begin{aligned}\mathcal{T}^{k*} = \underset{\mathcal{T}^k}{\text{argmax}} \prod_{T_i^{k-1}:\forall T_j^k \in \mathcal{T}^k, T_i^{k-1} \notin T_j^k} P_-(T_i^{k-1}) \prod_{T_j^k \in \mathcal{T}^k} \Big[P_{init}(T_{i_0}^{k-1}) P_+(T_{i_0}^{k-1}) \dots \\ P_{link}(T_{i_1}^{k-1}|T_{i_0}^{k-1}) P_{link}(T_{i_{l_k}}^{k-1}|T_{i_{l_{k-1}}}^{k-1}) P_+(T_{l_k}^{k-1}) P_{term}(T_{i_{l_k}}^{k-1})\Big]\end{aligned} \tag{1}$$

where $P_-(T_i^k)$ and $P_+(T_i^k)$ indicate the probability of a tracklet being a false alarm or real tracklet respectively, $P_{init}(T_i^k)$ and $P_{term}(T_i^k)$ is the probability of the tracklet being the first or last tracklet in a real trajectory respectively, and $P_{link}(T_j^k|T_i^k)$ is the

affinity score (or association probability) of linking the end of T_i^k to the start of T_j^k. This MAP formulation is then solved using the Hungarian algorithm.

A key component of this framework is the affinity model $P_{link}(T_i, T_j)$ and is the focus of this dissertation. Our following chapters will describe in detail my contributions to the affinity model.

First, I will introduce the datasets as well as the implementation details used to generate the initial set of tracklets, $\mathcal{T}^0$, through the detection and tracklet building steps. These sets of tracklets are used as the input the the tracklet matching stage for all future results in this dissertation.

3.1 Data Sets & Implementation Details

I evaluate my contributions on two insect datasets which are shown in Figure 1. First, is a dataset of four 5,000 frame videos containing *Temnothorax rugatulus* ants taken at thirty frames per second. Each video contains between 30 to 50 ants within each frame. Some ants are painted to assist in identification, and most stay within the colony for the entirety of the video with only a few ants entering and leaving. Full groundtruths for each of the four videos was manually obtained.

The second dataset consists of two 5,000 frame videos containing *Macrotermes michaelseni* termites taken at fifteen frames per second. Each video contains 22 unmarked termites which are confined within a circular dish for the entirety of the video. A full groundtruth for each video was given by the biologists which provided the videos.

Detection - For both datasets, we look for foreground blobs which meet size and shape constraints to use a detections. For the ant videos, we use color classification to classify each pixel as foreground or background based on a Gaussian mixture model within the L*a*b colorspace. To minimize noise, a morphological close is performed on resulting foreground binary image. Connected components within the foreground image which meet an area (460-1000 pixels) and aspect ratio (2.5-6.5) constraint are kept as the initial detection set, $\mathcal{D}$.

Because ants are often interacting and occluding, this detection method can miss a large number of detections. To supplement the color classification based detections, I exploit the fact that some ants tend to remain motionless for long periods of time. First, the location of each ant is manually provided in the first and last frame of the video. For each ant, a duration of no motion is found by examining the difference in a SIFT feature [38] taken at the ant's location in the first frame against subsequent frames at the same location. Once the Euclidean distance between the SIFT features exceed a threshold (empirically set to 290 in my experiments), the ant is designated as moving. The manually initialized location in the first frame is duplicated as detections through the duration of no motion. The no motion detections are then added to $\mathcal{D}$. This process is repeated searching backwards in time by starting at the last frame. To ensure only one detection per ant, we remove any region detections in $\mathcal{D}$ which overlap with the no motion ant detections.

For the termite data, a difference of Gaussians is used to classify pixels as foreground and background. Connected component regions within the resulting foreground binary image which meet an area (50 - 250 pixels) are kept as initial detec-

tions, $\mathcal{D}$. We do not perform the still region detection on the termites because unlike the ants, most termites remain in motion almost the entirety of the video.

Tracklet Building - To define the association probability S, we use three affinity measures based on affinity, position, and size. For appearance, we use the Bhattacharyya coefficient between each detection's RGB histogram. We use 8-bins per color for the RGB histograms creating a 24 value feature vector. Size and position are defined as zero mean Gaussian distributions based on the difference between the detections values. Sigma values for each Gaussian are manually tuned based on a sub sample of the groundtruth. The final association probability is the geometric mean of the three similarity scores.

Finally, the two thresholds are manually tuned to create as long of tracklets as possible, while minimizing the occurrence of ID switches or when detections which belong to different objects are incorrectly associated together.

CHAPTER 4: IRREGULAR MOTION MODELS

The affinity model is the likelihood that two tracklets belong to the same target. It is typically the combination of appearance, motion, and temporal affinity measures [27, 65]. In many cases, the final affinity score is the product of these three measures [27]. However, it has become increasingly popular to train an affinity model using a large array of features [60]. The appearance similarity measures can range in complexity from color histogram comparison [27], to online learned individual part based appearance models including histogram of gradients and covariance texture features [61, 63]. While appearance is often a highly individualized and distinguishing feature among pedestrians, insects may appear highly similar, or they may undergo non-rigid deformation which drastically changes their visual appearance.

As such, motion is an important feature for matching within insect videos. Within pedestrian videos, linear motion models often sufficiently model the behavior of the targets [5, 27]. The motion smoothness metric is a popular linear motion based affinity model used within DAT frameworks [27]. The motion smoothness of two tracklets T_i and T_j is computed by predicting the velocity of T_i into the future and comparing the predicted location of T_i to the actual location of T_j and vice-versa. This metric is effective in pedestrian datasets where objects move in smooth linear patterns. However, insects often move in highly irregular motion patterns changing direction frequently [29]. In addition, occlusions are frequent and can often last for

large numbers of frames (> 100 frames) due to insect's social nature and close proximity. During occlusions, insects become increasingly difficult to detect. Parts based detectors may be employed to help detect ants within partial occlusion, but due to the similarity of the ants parts as well as the requirement of rotation invariance this becomes a highly challenging problem and often creates noisy detections which leads to inaccurate tracking. This creates large gaps (or frames without detections) which tracklets must be associated across. Due to insects' highly irregular motion behaviors, linear motion models become increasingly less reliable in predicting the target's location over such long gaps. This ultimately leads to ambiguity when distinguishing between correct and incorrect associations at the tracklet matching stage. Figure 3 shows and example of when a linear motion model will fail while trying to predict an ants location after an occlusion. The green and blue painted ant enters into the occlusion moving left to right, but abruptly changes direction while it is occluded with an non-painted ant. A linear motion model would expect the painted ant to exit the occlusion near where the occluding non-painted ant exists or near the similarly colored green and and red painted ant is located.

We propose a set of new motion features based on random walk theory [12]. Random walks have been used extensively by biologists to model the motion of ants [47] and many other biological entities such as cells, fish and other animals [12, 59]. First, we discuss how an affinity model can be learned using a set of affinity measures and the HybridBoost algorithm of [36]. Next, we discuss some background of random walk theory in Section 4.2 and how they can be used to create motion affinity models appropriate for data association based tracking in Section 4.3. Finally, we evaluate

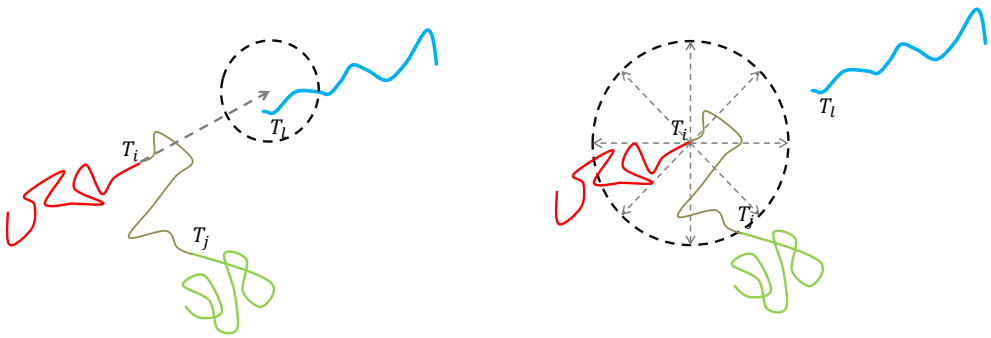
Figure 3: Shown is are two ants about to move into occlusion. The blue and green painted ant moves into the occlusion moving toward the right side of the image. However, it abruptly changes direction during the occlusion returning from where it entered into the occlusion. A linear motion model would be unable to predict this motion and would expect the green and blue painted ant to end up closer to where the occluding none painted ant or the nearby red and green painted ant are located after the occlusion.

the proposed features in Section 4.4.

4.1 Learning an Affinity Model with HybridBoost

We use HybridBoost to learn the term P_{link} for the MAP formulation as described in Chapter 3. HybridBoost combines the ranking ability of RankBoost [23] to determine how good an association is with the classification power of AdaBoost [24] to ensure poor associations are not given favorable scores. To collect training data in order to train the strong rank classifier, H, output of the previous round is associated to ground truth trajectories to determine good and bad associations to determine which tracklets should (not) be associated together.

Like AdaBoost and RankBoost, HybridBoost relies on a set of weak rank classifiers based on a set of features. We use the features proposed by [36] shown in Table 1. Note, we only include the top seven performing features as recommended by the authors of [36, 60]. These include features based on similarity in appearance and the motion smoothness metric. An 8 bin per channel RGB histogram of the pixels within an oriented bounding box is used to describe the appearance of the objects.



(a) (b)

Figure 4: When tracking is broken into multiple tracklets (T_i and T_j), a linear motion model (a) will often be unreliable in determing the correct association due to the insects irregular movements. A correlated random walk based measure (b), while no longer able to predict direction, provides a much better displacement measurement thus increasing the affinity score of the correct association due to its incorporation of the objects prior directional changes.

The Bhattacharyya coefficient between the mean of the last third of T_i and the first third of T_j is used as the appearance similarity measure. Only the first and last one thirds as used to account for changes in appearance over time and only compares the most temporally similar parts of each tracklet. The motion smoothness of T_i and T_j is computed by predicting the velocity of T_i based on its most recent positions and estimating where it will be by the start of T_j and vice versa. To reduce the effects of noise, we estimate the velocity of each tracklet (both forwards and backwards in time) using a Kalman filter.

4.2 Correlated Random Walks

A random walk follows the theory that a walker takes a series of discrete steps with a set distance or speed and an orientation. In our case, the walker is an ant whose step is the velocity traveled between adjacent frames. The simplest form of a random walk assumes the walker is unbiased and is equally likely to move any

direction with a set speed and it's direction is uncorrelated with previous steps. A more accurate model for ants is the Correlated Random Walk (CRW) model which accounts for persistence, or the likelihood of a walker to continue moving in the same direction [12]. Using the correlated random walk model, we can calculate an estimate of a walker's expected squared displacement, R^2, which we can use for association tracklets.

An expected displacement after a given number of steps, n, can be accurately calculated based on a walker's prior step history. Given the series of a walker's turning angles $\phi_f = \alpha_{f+1} - \alpha_f$ where α_f is the orientation of the ant in frame f, the persistence of the walk's direction of movement, c, and the relative likelihood of turning clockwise or counter-clockwise, s, are computed as $c = \int_{-\pi}^{\pi} \cos(\phi) g(\phi)\, \mathrm{d}\phi$ and $s = \int_{-\pi}^{\pi} \sin(\phi) g(\phi)\, \mathrm{d}\phi$ where $g(\phi)$ is a PDF of the likelihood of each turning angle. Then, the average turning angle of the walker is calculated as, $\phi_0 = \arctan(\frac{s}{c})$. Next, I describe three formulations for calculating the expected displacement of a correlated random walk given the walker's persistence c, directional bias s, the set of step speeds (or distances) L_T and number of steps the walker will take n.

Asymmetrical Correlated Random Walks. - The asymmetric fixed CRW is the most general formulation of a correlated random walk and is computed as:

$$\begin{aligned} R_a^2(T,n) &= n\bar{L_T^2} + 2(\bar{L_T})^2 \\ &\times \left(\frac{n(c - c^2 - s^2) - c}{(1-c)^2 + s^2} + \frac{2s^2 + (c^2+s^2)^{(n+1)/2}\gamma}{((1-c)^2+s^2)^2} \right) \end{aligned} \tag{2}$$

where $\gamma = ((1-c)^2 - s^2)\cos((n+1)\phi_0) - 2s(1-c)\sin((n+1)\phi_0)$, L_T is the set of all instantaneous speeds for tracklet T, $\bar{L_T}$ is the mean of the values in L_T , $\bar{L_T^2}$ is the

mean of the squared values in L_T , and n is the number of frames.

Symmetrical Correlated Random Walk. - The affect of turning tendency (s) on Equation (2) is only significant for large n. Thus, we also include the symmetrical CRW which assumes $s = 0$ in Equation 2 and reduces to,

$$R_s^2(T, n) = (\bar{L_T})^2 \left(n \left(\frac{1+c}{1-c} \right) - \frac{2c(1-c^n)}{(1-c)^2} \right). \tag{3}$$

Symmetrical Variable Speed Correlated Random Walk. - To model the variability in speed, the coefficient of variation of step size $b^2 = \bar{L_T^2}/(\bar{L_T})^2 - 1$ may be incorporated into to Equation 3 resulting in:

$$R_v^2(T_i, n) = (\bar{L_T})^2 \left(n \left(\frac{1+c}{1-c} + b^2 \right) - \frac{2c(1-c^n)}{(1-c)^2} \right). \tag{4}$$

4.3 Motion Features based on Correlated Random Walks

We formulate the CRW feature as a probability of T_i dispersing to the start of T_j and vice-versa as follows:

$$\begin{aligned} P_{R_*}(T_i, T_j) =& \mathcal{N}\left(d(T_i, T_j); \sqrt{R_*^2(T_i, n)}, \sqrt{\sigma_{R_*}^2(T_i, n)} \right) \times \\ & \mathcal{N}\left(d(T_i, T_j); \sqrt{R_*^2(T_j, n)}, \sqrt{\sigma_{R_*}^2(T_j, n)} \right) \end{aligned} \tag{5}$$

where $\mathcal{N}$ indicates a Gaussian distribution, $d(T_i, T_j)$ is the point to point distance between the end of T_i and the start of T_j, R_*^2 can be Equation (2),(3), or (4), n is the number of frames between the end if T_i and start of T_j, and $\sigma_{R_*}^2$ is the variance of the CRW measure. A good approximation of the CRW's mean squared displacement

variance can be calculated using the Mean Dispersal Distance (MDD) measure [59]. A closed form of the MDD for a CRW is an open problem; however, a good approximation of the MDD, we refer to as R_D, is as follows

$$R_D(T,n) = \frac{\sqrt{\pi R_*^2(T,n)}}{2}. \tag{6}$$

This formulation can be combined with any of the above R_*^2 measures (more information on the MDD can be found in [12]). We use R_D to calculate the expected variance in displacement as follows:

$$\sigma_{R_*}^2(T) = R_*^2(T) - (R_D(T,n))^2 \tag{7}$$

This approximation of the variance has been shown to only slightly over-estimate the variance of simulated data [59] and is accurate enough for our purposes. We use Equation (5) using Equations (2), (3), and (4) as Features 8, 9, and 10 in Table 1 respectively.

Because the CRW feature becomes more accurate as the length of the tracklets increases, we supplement the CRW measure with a simple displacement measure, $D(T,n) = n\bar{L_T}$, for the earlier stages when most tracklets are short. We use this measure in the same manner as the CRW measure, replacing $\sqrt{R_*^2(T,n)}$ and $\sqrt{\sigma_{R_*}^2(T,n)}$ terms in Equation (5) with $D(T,n)$ and σ_D (found empirically based on ground truth) respectively and becomes Feature 11 in Table 1. Additionally, the differences in the simple displacement measure and the actual distance between two tracklets, $|2d(T_i,T_j) - D(T_i,n) - D(T_j,n)|$, is used as Feature 12, and we normalize this value

Table 1: List of association cost features.

	Id	Description
Features proposed by [36]	1	Bhattachayrra Coefficient between RGB Histograms of T_i and T_j. (App)
	2	Frame Gap between tail of T_i and head of T_J
	3	Number of miss detected frames in the gap
	4	Number of frames occluded by other tracklets in the gap
	5	Number of miss detected frames in gap divided by Frame Gap
	6	Number of occluded frames in gap divided by Frame Gap
	7	Motion Smoothness (MS)
Irregular Motion Features	8	Asymmetric CRW ($P_{R_a}(T_i, T_j)$)
	9	Symmetric CRW ($P_{R_s}(T_i, T_j)$)
	10	Variable Speed CRW ($P_{R_v}(T_i, T_j)$)
	11	Simple Displacement ($P_D(T_i, T_j)$)
	12	Difference in $D(T, n)$ and $d(T_i, T_j)$, $(D - d)$
	13	Feature 12 normalized over frame gap $((D - d)/n)$
	14	Difference in Mean Speed

over the frame gap, n, for Feature 13.

Finally, due to insects' tendency to move at consistent speeds before and after occlusions, we use the difference in mean speeds, $|\bar{L_{T_i}} - \bar{L_{T_j}}|$ as our final motion feature.

4.4 Evaluation

We evaluate the proposed irregular motion features on the termite and ant datasets described in the previous chapter. First, we describe a few implementation details of the tracklet matching step for each dataset. This is followed by a discussion of the our evaluation metrics. Finally, we discuss the results of our method.

4.4.1 Implementation Details

For the ant dataset, we perform four stages of tracklet matching. The maximum allowed frame gap at each stage, τ_k, is 8, 32, 128 and 512. For the termite dataset, we

only perform the first three stages of tracklet matching because it does not contain any gaps exceeding 128 frames within the data set.

For each stage, we train a strong rank classifier H by using a leave one out training strategy for each video in the datasets (i.e., three videos for training in the ant dataset and one video for training in the termite dataset). Each strong rank classifier is trained to contain 100 weak rank classifiers and the combination coefficient is set to 0.75 following [36]. Only associations which exceed a minimum threshold (set to 0 for our experiments) out of the strong rank classifier are considered within the tracking association process. Finally, true positive and false positive probabilities are based on the precision for each dataset.

4.4.2 Evaluation Metrics

We use the commonly used metrics of [36, 33] to evaluate the performance of our proposed method. They are as follows:

- Recall - The ratio of correctly matched detections to the total number of detections.
- Precision - The ratio of correctly matched detections to total detections in tracking results.
- False Alarms per Frame (FAF) - Average false alarms per frame.
- Groundtruth Targets (GT) - Total number of ground truthed targets within videos

Table 2: The tracking performance on each dataset. Including our irregular motion features (w/ Irr. Motion) reduced Fragments by 31% and ID switches by 57% compared to only using the baseline features proposed by [36] (w/o IrrFeat) on the ant dataset, and reduced Fragments by 17% and ID switches by 3% on the termite dataset.

Method	Recall	Prec	FAF	GT	Frag	IDS	MT	PT	ML
Ant Dataset									
w/o Irr. Motion	61.7%	99.0%	0.318	166	391	332	54	98	14
w/ Irr. Motion	61.6%	99.0%	0.310	166	269	143	54	98	14
Termite Dataset									
w/o Irr. Motion	93.2%	94.5%	1.179	44	68	32	39	4	1
w/ Irr. Motion	93.6%	94.5%	1.192	44	56	31	39	5	0

- Fragments (Frag) - The total number of times a ground truth trajectory is interrupted by tracking results.

- ID Switches (IDS) - The total number of times a tracked trajectory changes its matched ground truth identity.

- Mostly Tracked (MT) - The number of ground truth trajectories successfully tracked for more than 80% of their duration.

- Partially Tracked (PT) - The number of ground truth trajectories successfully tracked between 20% and 80% of their duration.

- Mostly Lost (ML) - The number of ground truth trajectories which are tracked for less than 20% of their duration.

4.4.3 Discussion of Results

Table 2 shows the comparison of tracking performance on each dataset using the features of [36] and our proposed irregular motion features. Note that the measures are the sum of the performance across all videos in the data set (e.g, there are 166 total

tracking targets across all four ant videos and 269 Fragments occurred during all four videos when using our irregular motion features). The addition of the irregular motion features decreased the number if ID switches by 57% and fragments by 31% in the ant dataset. In the termite dataset, ID switches were reduced by 3% and fragments by 17%. This is due to the ability of the proposed irregular motion features to more accurately model the motion of the insects. The smaller reduction in the termite dataset is primarily due to the shorter occlusion seen in the dataset which are often shorter than 100 frames. This is in contrast to the ant dataset which often contains occlusions lasting up to 500 frames. Figure 7 shows an example of correctly associating tracklets over a long occlusion lasting over 200 frames. Although our irregular motion features are accurate even with insects which dramatically change direction during these occlusions, they still have trouble predicting when ants dramatically change speed. Figure 8 and Figure 9 show two examples in the termite dataset where our irregular motion features prevented ID switches. Many of the fragments remaining are insects which suddenly begin moving after remaining still for several hundred frames or vice-versa.

Table 3 and Table 4 shows the top three features used by the strong rank classifier H at each association stage when training with the entire ant and termite datasets respectively. For all but one stage, one of our proposed irregular motion features was the first picked (most discriminating) feature. Of note, the first selected weak ranker had double the weight of the next selected weak ranker for every model. Figure 5 and Figure 6 shows a more granular breakdown of the effectiveness of each feature. On the ant dataset, one of our proposed irregular motion features had the highest cumulative

Table 3: Shown are the top three features selected at each stage of training from Table 1 when a HybridBoost ranker is trained with the entire ant dataset. One of our proposed irregular motion features is the top selected feature in all stages. Of note, the first selected weak ranker at each stage was weighted more than double the next selected weak ranker.

Max	**Feature**		
Frame Gap	1st	2nd	3rd
8	P_{R_s}	P_{R_v}	MS
32	P_{R_s}	MS	$(D-d)/n$
128	P_{R_s}	MS	$D-d$
512	P_{R_s}	MS	# Occ. Frames

Table 4: Shown are the top three features selected at each stage of training from Table 1 when a HybridBoost ranker is trained with the entire termite dataset. One of our proposed irregular motion features is the top selected feature in all stages.

Max	**Feature**		
Frame Gap	1st	2nd	3rd
8	P_{R_a}	MS	$(D-d)/n$
32	P_{R_s}	P_{R_a}	P_D
128	P_{R_v}	$(D-d)/n$	$D-d$

weight at any given stage with the exception of the first stage (max allowed gap of 8) where the motion smoothness metric showed most useful. As the gaps increase in size, the usefulness of the motion smoothness metrics lessens while our proposed metrics, especially the correlated random walk based metrics, continue to provide useful information in determining associations. The same trends can be seen in the termite dataset. Although the motion smoothness metric is more generally useful at all stages in the termite dataset, the combined weights of our three proposed correlated random walk based metrics are larger than the cumulative weight of the motion smoothness metric.

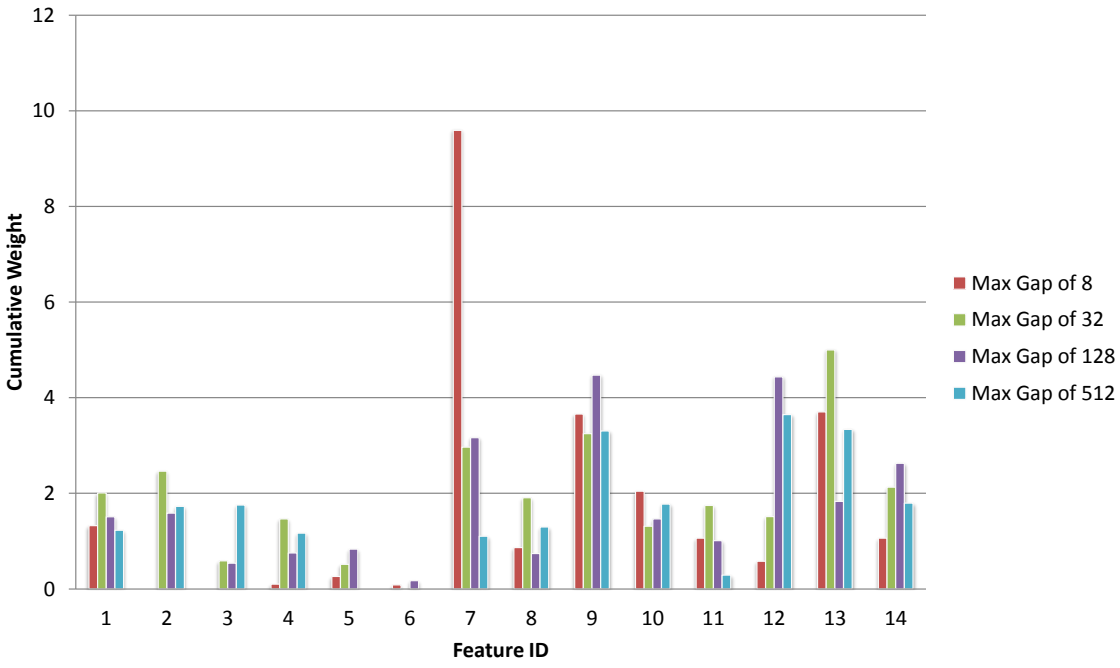


Figure 5: The cumulative weights of all the selected weak rankers based on each feature when a HybridBoost model is trained using the entire ant dataset. With the exception of the first stage of tracking (maximum allowed gap size of 8), our proposed irregular motion features, specifically the correlated random walk features (features 8 to 10), provide more information than the initial set of features proposed by [36].

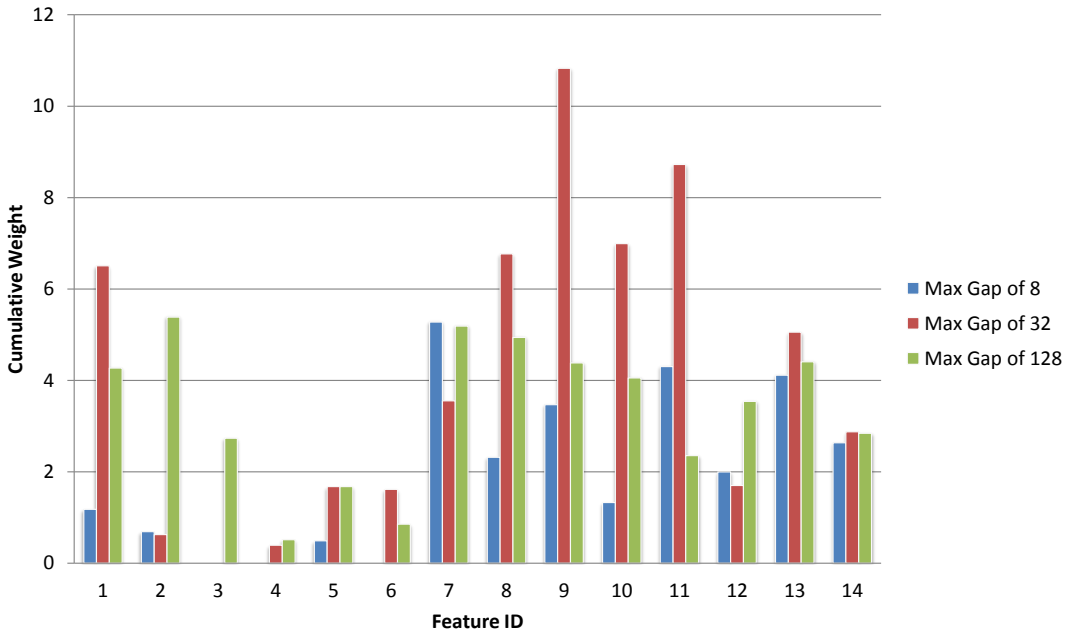
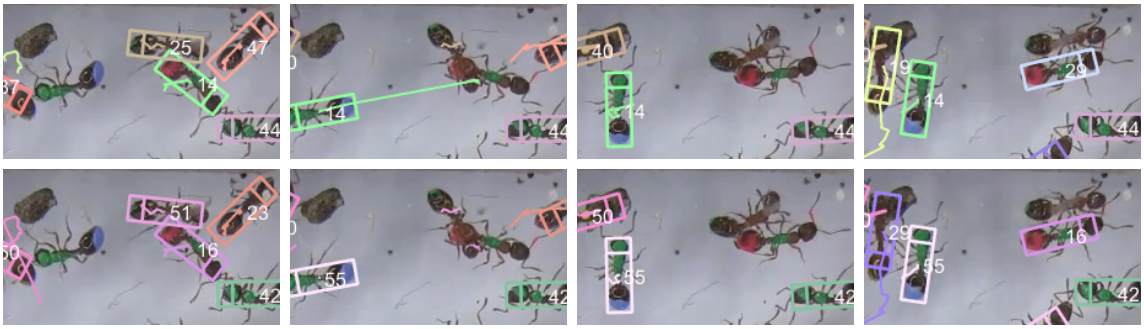
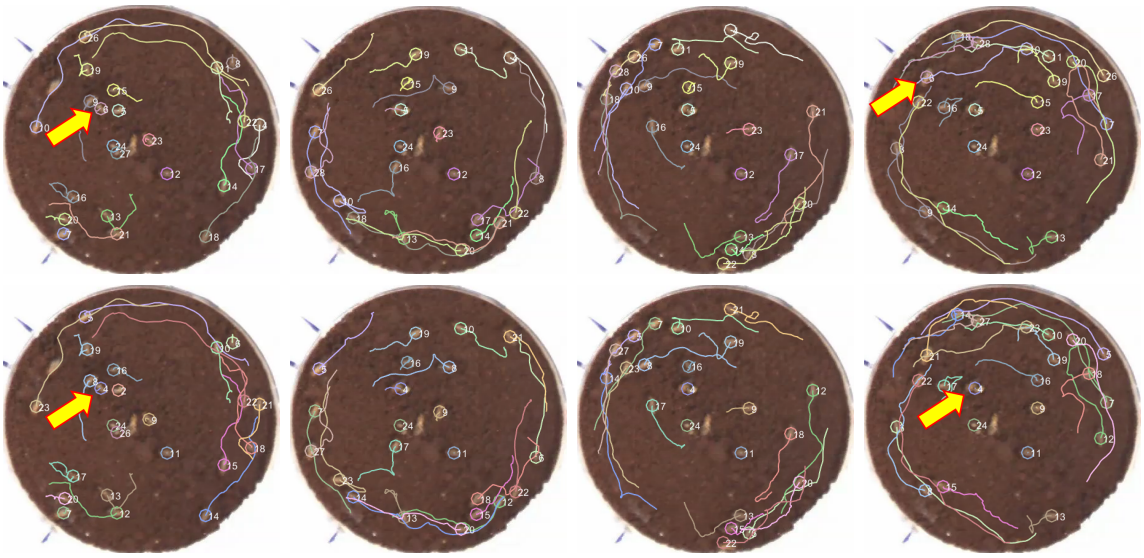


Figure 6: The cumulative weights of all the selected weak rankers based on each feature when a HybridBoost model is trained using the entire termite dataset. At each tracking stage, our proposed features (features 8 to 14) provided more information (higher weights) than the features of [36]. Although the motion smoothness feature (feature 7) had a higher weight than any individual irregular motion feature in the first and last stages, our proposed correlated random walk features (features 8 to 10) provided more information combined.



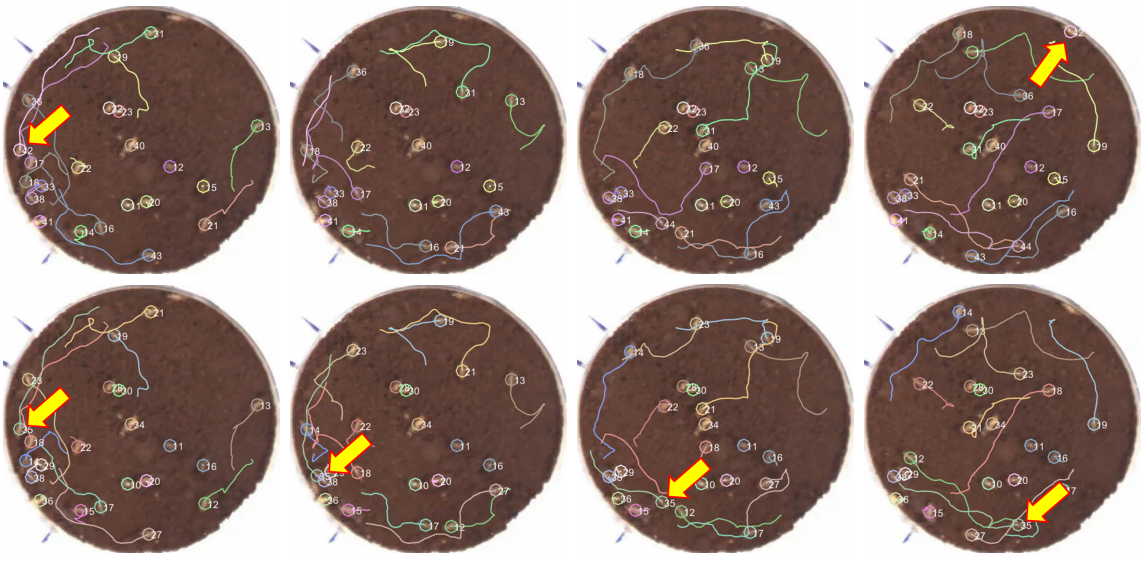
Frame 990 Frame 1070 Frame 1150 Frame 1215

Figure 7: Sample tracking results of using our irregular motion features (bottom row) and only using the features proposed by [36](top row). In the top row, tracking 14 in green produces an ID switch due to the similarity in appearance (both have green paint) and the length of the occlusion. Our proposed irregular motion features are able to better predict how far the ant will move during the occlusion and taking the correct association at the end of the occlusion.



Frame 565 Frame 610 Frame 655 Frame 705

Figure 8: Sample tracking results of using our irregular motion features (bottom row) and only using the features proposed by [36](top row) in the termite dataset. In the top row, tracking 6 in pink (indicated by arrow) produces an ID switch due to an abrupt change in position right before tracking is lost and tracking is reappears later on a different in the final frame (indicated by arrow). Our irregular motion features are able to reduce the effect of the noisy detection and track through the duration as tracking 4.



Frame 990 Frame 1070 Frame 1150 Frame 1215

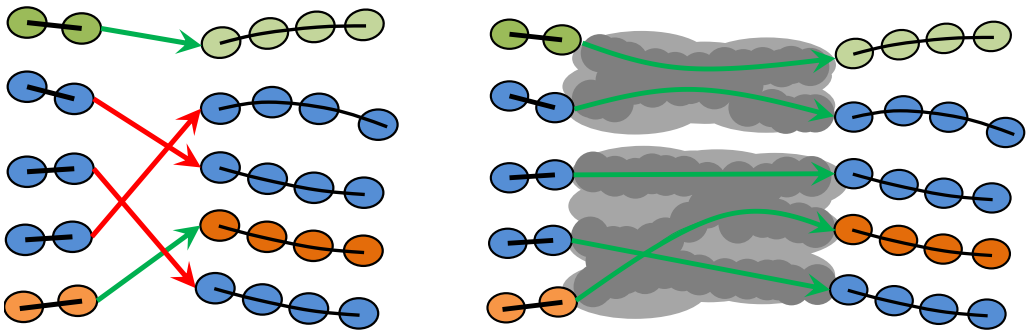
Figure 9: In the top row, tracking 42 in pink (indicated by arrow) produces an ID switch as it changes direction quickly to move around other targets and is incorrectly associated to another termite on the other side of the image. Our irregular motion features are able to track continually through these quick directional changes.

CHAPTER 5: FILTERING ASSOCIATIONS WITH OCCLUSION SUB-TUNNELS

Due to the density and frequency of occlusions within the social insect videos, each tracklet can have a large number of potential associations. The advanced features described in the previous chapter can, at best, reduce the ambiguity between potential associations by increasing the difference between the correct and incorrect matches. Similarity in appearance and motion between the many nearby, occluding insects may still cause the affinity model to confuse two or more objects. This can lead to incorrect associations (ID Switches) or early termination (fragments).

One approach to reduce this ambiguity further is to incorporate dependencies between associations [60, 9, 49]. However, the size of the problem grows exponentially with the number of objects present within the video. It is not uncommon for insect videos to contain 50+ objects, often making these methods intractable.

We propose a method to filter incorrect associations by leveraging occlusion. Although tracking individual objects through occlusions can be difficult, the occlusion itself can provide information on spatial paths available for tracklets. By limiting associations between tracklets which enter and exit from the same occlusion, we can filter many impossible nearby associations from being considered during the tracklet matching process. We call these distinct instances of interacting tracklets *occlusion tunnels*. However, finding distinct occlusion tunnels is challenging, therefore we ap-



(a) Typical Matching Problem (b) Proposed Occlusion Sub-Tunnel Filtering

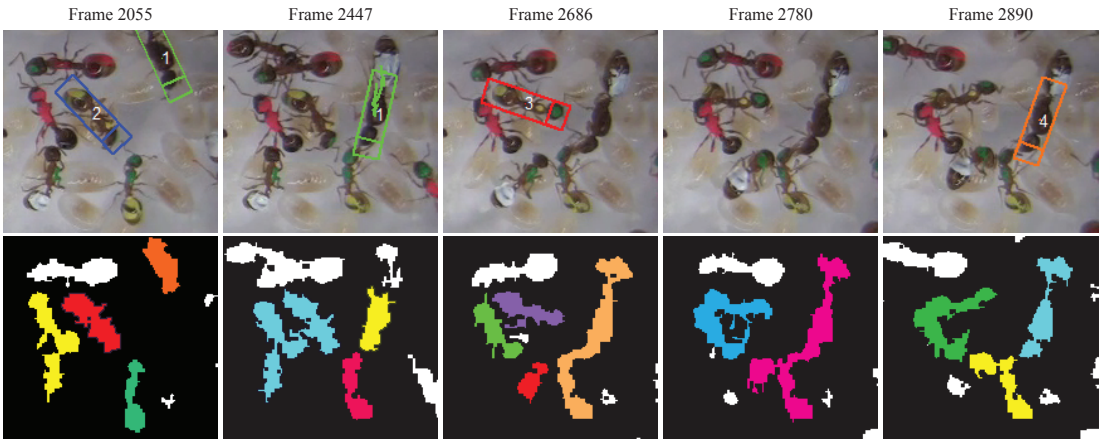
Figure 10: Due to the difficulty of detection during occlusion, trackings are often fragmented into many smaller tracklets. Due to similarity of features (shown as color) between targets in, identity switches may occur as shown by the red arrows in (a). Even though detections are difficult to obtain, foreground is still relatively easy to obtain and can be used to find instances of occlusion as shown by the gray region in (b). The occlusion sub-tunnels based on foreground blobs (shown in dark gray) can be used to filter incorrect association between tracklets.

proximate these areas of by using *occlusion sub-tunnels* as shown in Figure 10.

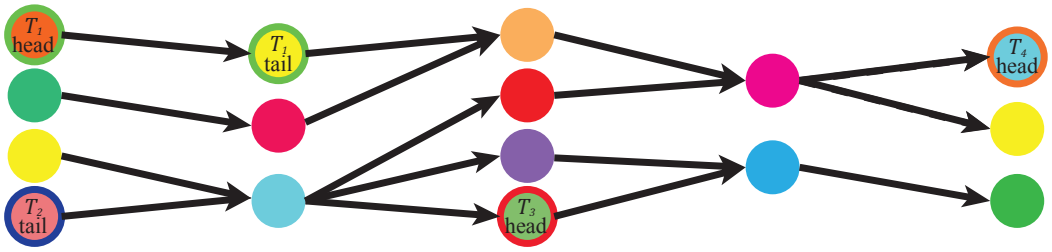
In the following section, we describe our method for finding and exploiting occlusion sub-tunnels. Next, we demonstrate the effectiveness of occlusion sub-tunnels in Section 5.2.

5.1 Occlusion Sub-Tunnels

The occlusion sub-tunnel approach aims to improve the data association process by filtering incorrect associations. Occlusion tunnels are a sequence of spatial regions of occlusion which are defined by the assumption that a set of targets 1) enter into an occlusion, 2) move through the occlusion, and finally 3) exit from the same occlusion. Therefore, tracklets which enter into a given occlusion must be associated with a tracklet which exits the same occlusion as shown in Figure 10b. Tracklets not involved within the same occlusion (defined by gray regions in Figure 10b), cannot



(a) The top row shows the original frames where each colored bounding box is a tracklet (only tracking from two ants are shown for clarity). Detected foreground blobs are shown below (foreground blobs involved with the occlusion are colored). Note that each foreground blob can contain one or more ant.



(b) The foreground tunnel F generated based on the foreground blobs for Figure 11a. Each foreground blob becomes a vertex in the graph (e.g, the yellow foreground blob from Frame 2055 is represented as the yellow circle). Edges between nodes denote the foreground blobs meet the temporal and spatial constraint. Next, the head and tail detections of each tracklet are corresponded with a node in the F denoted by text and outline color of the vertex pixel.

Figure 11: Example of constructing foreground tunnels on an occlusion involving five ants over 800 frames.

be associated and therefore have their associations filtered from the tracking process. Note, the goal of occlusion tunnels is not to perform tracking explicitly, but to filter associations based on a spatial constraint irrespective of features.

First, it is assumed that a set of tracklets, $\mathcal{T} = \{T_i\}$, and set of foreground blobs at each frame are given. We make a distinction between foreground blobs and detections. For our purposes, foreground blobs are simply connected foreground pixel regions within a frame which may contain one or more objects where as detections are assumed to only reference a single object. This is demonstrated in Figure 11a where the top row show detections of tracklets which correspond to a single object and the second row shows foreground responses (colored regions) which contain no objects, one object or multiple objects. (Note, our method is agnostic to the particular method used for detecting foreground blobs and details of the methods used for our datasets are described in Section 5.2.1). If the foreground blobs are viewed as being stacked over time, they form "tunnels" through the video which targets can move through. These tunnels are represented as a directed graph $F = (V, E)$ where $V = \{v_1, ..., v_p\}$ denotes the set of nodes (or vertices) and $E = \{e_1, ..., e_q\}$ denotes the set of edges. Each node, $v_i = \{p_i, t_i\}$, in F represents a foreground blob containing the set of pixels, p_i, at frame t_i. Next, an edge is added between nodes which are considered to contain one or more of the same targets. This is approximated by a spatial and temporal proximity. Two nodes v_i, v_j are temporally close if $0 < t_j - t_i \leq n$ where n is a temporal threshold. For our datasets, nodes are considered spatially close if $p_i \cap p_j \neq \emptyset$. This is due to the relatively high frame rate of our videos which limits the distance objects are able to move between frames. More sophisticated mea-

sures may be used to determine spatial proximity in cases of faster moving targets or slower frame rates. Figure 11b shows a foreground graph constructed from the colored foreground regions in Figure 11a. Note that more than one detection can be corresponded to a single foreground blob in F. In the case that a foreground blob does not exist at the location of a detection, a foreground blob based on the size and location of the detection may be inserted into F.

Occlusion tunnels exist as a set of disjoint subgraphs within F. However, in order to find these subgraphs directly, each node in F would have to be accurately identified as containing an occlusion (foreground blob contains more than 1 object), contains a single object or contains no object (e.g., noise).This is a challenging task, especially within insect videos where severe and total occlusions between objects are not uncommon. Instead, we approximate the location of the occlusion tunnels by finding occlusion sub-tunnels.

The occlusion sub-tunnels are represented as a directed graph, O_s, which contains groups of tracklets which are connected (or reachable) through the foreground blob tunnels, F, and their associations. The occlusion sub-tunnels are found by finding groups of tracklets which are connected through the foreground blob tunnels, F.

First, we create a directed graph $C = \{\mathcal{T}, A\}$ which has a node for each tracklet in $\mathcal{T}$. The set of edges $A = \{a(i, j)\}$ indicates the head (first detection) of T_j is reachable by the tail (final detection) of T_i through the foreground tunnels, F. To determine the set of the edges A, the tracklets are first corresponded to nodes in the graph, F. Next, we must correspond detections within $\mathcal{T}$ to foreground blobs in F. Because our criteria for A is based on reachability between the heads and tails of tracklets,

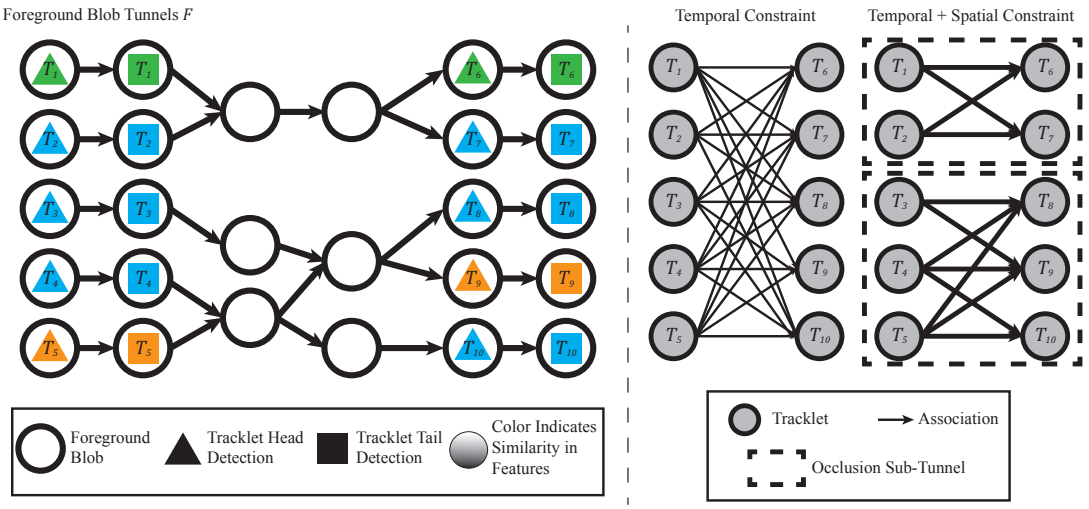


Figure 12: Given the graph of foreground blob tunnels F (shown on the left as a network of white circles) and a set of tracklets, $\{T_1, ..., T_{10}\}$. The heads (triangles) and tails (squares) of each tracklet are mapped to nodes in F. The middle graph shows all potential associations between tracklets with only a temporal constraint (e.g., τ_k at each tracking stage). Then, potential associations between tracklets are filtered based on reachability between tail and head nodes through the foreground blob tunnel. The resulting graph, C (right most graph), contains two disjoint sub-graphs each with its own sub-tunnel network, O_s. This filters confusing associations across separate occlusions (T_2 to $\{T_8, T_{10}\}$, and $\{T_3, T_4\}$ to T_8) and within occlusion, (T_3 to T_{10}).

we only correspond each tracklet's head and tail detections. Each detection d_i is corresponded to a single foreground blob v_i where v_i and d_i exist in the same frame and d_i is located within the foreground region represented by v_i. This correspondence is demonstrated in Figure 11b.

We refer to the set of nodes of F corresponded to tracklet heads as $\{v_i^h\}$ where v_i^h is the node in F corresponded to the head detection of tracklet T_i. The set of nodes of F corresponded to tracklet tails are referred to as $\{v_i^t\}$. A toy example of this mapping is shown in the left half of Figure 12. With the heads and tails corresponded, the conditional for adding edges can be stated as follows: an edge $a(i, j)$ is added to C if there exists a path through F from v_i^t to v_j^h. This condition is easily checked using a breadth first search through F from the tail node, v_i^t. Because F is directed, this conditional is both a temporal (all edges in F are directed forward through time) and spatial (edges in F define the foreground tunnels within the video) constraint on the set of potential associations between tracklets. Figure 12 demonstrates the process of finding occlusion sub-tunnels.

5.2 Evaluation

As in the prior chapter, we start by describing the implementation details used for our datasets in Section 5.2.1. A discussion of the performance of the two proposed methods are discussed in Section 5.2.2.

5.2.1 Implementation Details

Tracklet Matching - We use the same tracklet matching details as described in the prior chapter. Four rounds of tracklet matching are performed with maximum gaps

of 8, 32, 128 and 512 allowed at each stage for the ant dataset and only three rounds of matching for the termite dataset. We continue to use the trained HybridBoost affinity model with the described irregular motion features following the same training method as in Chapter 4.

Foreground Blob Detection - As mentioned in Section 5.1, our occlusion tunnel and occlusion sub-tunnel finding methods are agnostic to the foreground blob detection method. It is only assumed that the method has a high recall and moderate precision. We use the same color classification method used to perform ant detection as described in Section 3.1. Color classification is chosen due to its high recall of foreground within the video. Background subtraction is not usable due to the number of motionless ants which makes it difficult to generate an accurate background image. Only blobs which do not meet a minimum size constraint of 50 pixels (approximately the size of an ant's head) are discarded. All other blobs are used to create the foreground tunnels, F.

For the termite data, we use background subtraction as it provides a high recall within these videos. The average image of the video is used as the background and empirically set a threshold that maximized the recall of foreground blobs. We again set a minimum size threshold of 25 pixels (approximately the size of a termite's head) on the set of foreground blobs.

5.2.2 Discussion of Results

Results of the Occlusion Sub-Tunnels method is shown in Table 5. By using occlusion sub-tunnels to filter associations, we were able to reduce fragments by an

Table 5: The tracking performance on each dataset while. Our proposed occlusion tunnel and occlusion sub-tunnel methods reduced fragments by an additional 5% and ID Switches by an additional 21% over using the irregular motion features alone in the ant dataset. On the termite dataset, ID Switches were reduced by 23% but fragments increased by 14%.

Method	Recall	Prec	FAF	GT	Frag	IDS	MT	PT	ML
Ant Dataset									
Irregular Motion	61.6%	99.0%	0.310	166	269	143	54	98	14
Occ. Sub-Tunnels	61.7%	99.0%	0.318	166	256	113	54	98	14
Termite Dataset									
Irregular Motion	93.6%	94.5%	1.192	44	56	31	39	5	0
Occ. Sub-Tunnels	94.6%	95.6%	1.173	44	64	24	39	5	0

additional 5% and ID switches by an additional 22% compared to tracking without filtering. This is due to the occlusion tunnel and occlusion sub-tunnel methods' ability to to filter incorrect associations as shown in Table 6. At each matching stage, on average 4,000 associations are considered and scored by the trained affinity model with the vast majority of these being incorrect associations. The occlusion sub-tunnel method is able to filter nearly 85% of the incorrect associations at each tracking stage while only incorrectly filtering 0.25 correct associations per tracking stage. Filtering of correct associations occurs when a series of frames has poor foreground response which breaks paths through the foreground tunnel F. Figure 13 shows three prevented ID switches due to occlusion sub-tunnel filtering. As the ant in the top right occludes with each of the three subsequent ants, tracking without occlusion sub-tunnels has a chain reaction of ID switches (one tracklet takes another tracklets correct match and therefore takes an incorrect "good enough" association instead). By filtering incorrect associations, all three tracklets are associated correctly over the occlusions.

In the termite dataset, ID switches were reduced by 23% as well, but increased

Table 6: Filtering ability of proposed occlusion sub-tunnels (OsT) on the ant and termite datasets. Average of all tracking stages and and videos are shown. On average, occlusion sub-tunnels filter almost 85% of incorrect associations and only incorrectly filter one correct association on the ant dataset. In the termite dataset, 58% of associations are filtered, but due to lower recall of the foreground blob detection, it incorrectly removes 4.6 correct associations per tracking stage.

	# Associations	% Filtered	# Correct Assoc. Filtered
Ant Video	4,145	84.9%	0.25
Termite Video	357	58.0%	4.6

fragments by 14%. As in the ant dataset, the occlusion sub-tunnel method is able to filter 85% of associations, but often filters a larger number of correct associations than in the ant video (Table 6) due to lower recall of foreground blobs in these videos.. However, even with the increased fragments, both precision and recall were increased by 1% in the termite video. Precision is increased when fewer false positive tracklets exist at the end of tracking. Our occlusion sub-tunnel helped by eliminating associations between these false positive tracklets and actual tracklets thus allowing the algorithm to correctly remove them. At the same time, this also allowed for more difficult associations between shorter tracklets preventing them from being filtered as false positives and increasing recall. The same response can be seen in the precision of the ant video, however, because most of the false positive tracklets are very short they do not have a appreciable affect on the percentage values. Figure 14 shows an instance where an ID switch was prevented but remained as a fragment.

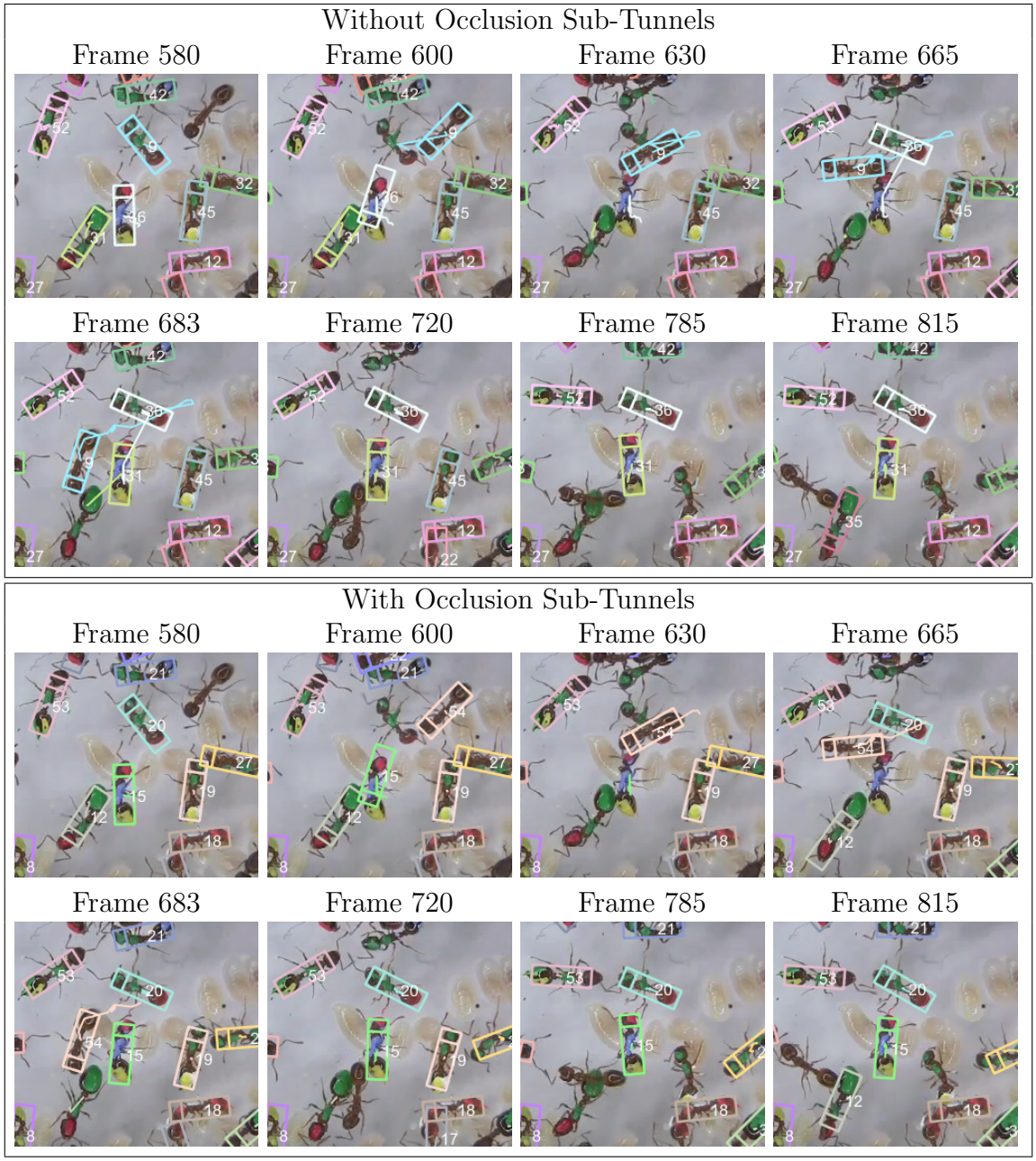
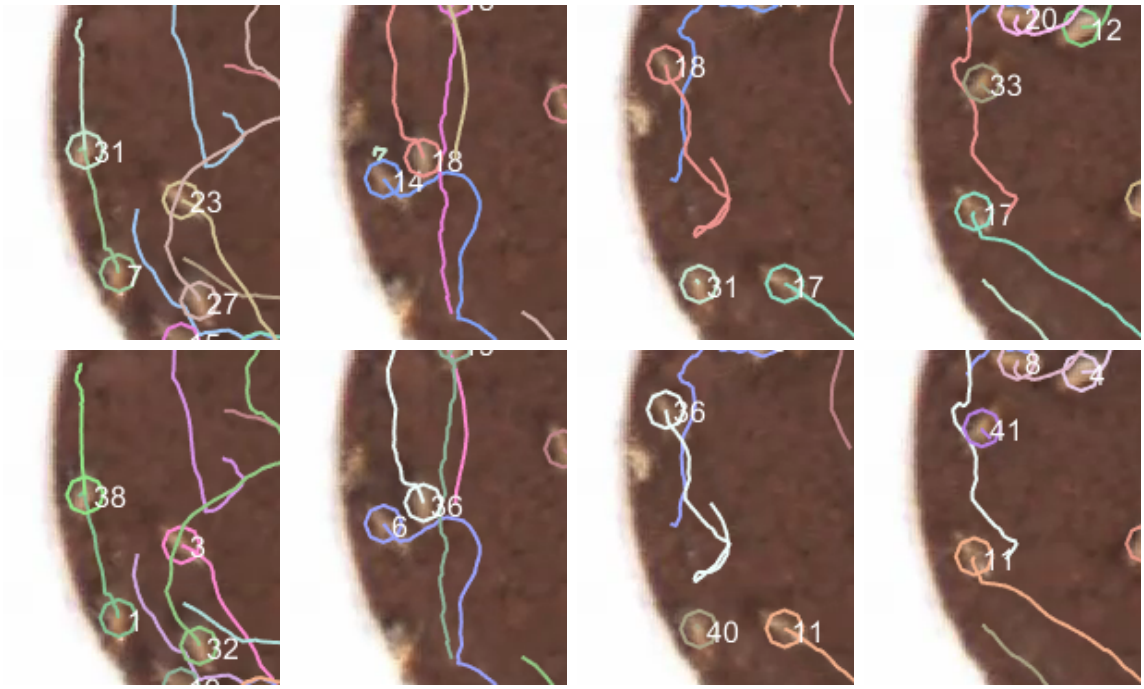


Figure 13: The top series of images shows a series of three ID switches which occur when not using Occlusion Sub-Tunnel filtering. As the unmarked ant in the top right corner moves to the bottom right, it occludes with trackings 9 (cyan), 36 (white) and 31 (yellow) which all switch to a different ant after the occlusions. By using occlusion sub-tunnels, these incorrect associations are prevented, thus allowing each the trackings to maintain their identities (trackings 20, 15, and 12) after the occlusion as shown in the bottom series of images.



Frame 2585 Frame 2640 Frame 2690 Frame 2715

Figure 14: The top row of images show an ID switch which occurs when no using occlusion sub-tunnel filtering. The termite tracking by tracking 31 presses against the outside of the enclosure in frame 2640 and stops where detection is lost. Due to the the prior motion of the termite, it is incorrectly associated to a different termite at frame 2690. Occlusion sub-tunnels prevent this association occurring and instead leaves a fragment.

CHAPTER 6: ONLINE LEARNING AFFINITY MODELS

There are a few options in relation to the affinity model when tracking on a dataset which is different from the exiting set of groundtruthed videos used for training. First option is to train a model using a the exiting dataset which has a groundtruth and apply it to the new video domain (e.g., train a model using the ant dataset and apply it to the termite dataset). However, affinity models do not always transfer between datasets well as motion patterns and descriptive appearance indicators can change between datasets leading to decreased tracking performance. Second, a new model can be trained, but this requires collecting a large number of training labels. For example, the HybridBoost models trained in prior chapters use in excess of 70,000 training samples when training an affinity model on the ant dataset. Active learning can be applied to reduce the number of samples required [44], but this still requires a user to label upwards of 200 training samples per tracking stage (or 800 labels total) to achieve reasonable performance.

Recently, methods which train an appearance affinity online without user labels have been investigated. These methods automatically generating training data based on the current state of tracking. In DAT tracking, the observation that tracklets which overlap temporally out of the tracklet building stage cannot belong to the same object can be used to generate a large set of training labels. Using this observation, [33, 32, 63] trained a classifier to determine if two detections belong to the same

object using pairs of detections within a tracklet as positive samples and pairs of detections between temporally overlapping tracklets as negative samples. This was later extended by [61] which used a motion heuristic to identify a set of potential future matches for each tracklet. Pairs of detections between a tracklet and a potential future match are given a weak positive label as some pairs are actually negative samples. Multiple instance learning is used with the weakly labeled positive samples to an appearance model.

However, appearance is not a reliable metric in tracking social insects due to their similarity. Motion features (as demonstrated in Chapter 4) are important features in tracking insects, but the methods for obtaining training labels do not extend to labeling association between tracklets which are required to train motion features. Put simply, any machine learning method would merely learn the motion heuristic used for generating the training data. However, the occlusion sub-tunnel method (Chapter 5) is suitable for creating such weak labels for training motion features online because it is based on a spatial constraint. In the next section, we describe how the occlusion sub-tunnel method can be used to generate the weakly labeled bags suitable for training motion affinity models. Next, in Section 6.2 we describe a number of multiple instance learning methods which may be suitable for training an affinity model. Finally, we evaluate our proposed online affinity model training method in Section 6.3.

6.1 Using Occlusion Sub-Tunnels to Generate Weak Labels

Multiple Instance Learning (MIL) is a variation of the classification task for problems with incomplete knowledge of training labels [17]. Rather than each instance of an absolute positive or negative, individual instances are only weakly labeled indicating that some labels could be incorrect. Instances are then grouped together into sets, or *bags*. In the binary case, a bag is given a positive label if *at least* one instance in that bag is thought to be positive, and a bag is labeled as negative if *all* instances in the bag are negative. The goal of MIL is to label new unseen instances or bags of instances using the labeled bags as training data (where again, no individual instance is given a hard label) [17, 55]. These methods work by assuming that there exists overlap in the feature space between the hard labeled negative instances in negative bags and the negative instances which are placed in the positive bags.

The occlusion sub-tunnel method described in the previous chapter is capable of producing the weak labels usable by MIL and suitable for training motion models. Associations which are temporally valid but are filtered by the occlusion sub-tunnel method are given a negative label. Unfiltered associations are given a weakly positive label. Because our filtering is based on occlusion, associations between objects which are close by but in separate occlusions (as shown in Figure 16) can be filtered and given negative labels as shown in Figure 15. These nearby filtered associations are what allow the MIL method to learn a decision boundary as they will be near incorrect associations which are not filtered by occlusion sub-tunnels in feature space.

After each association is given a negative or weak positive label, they must be

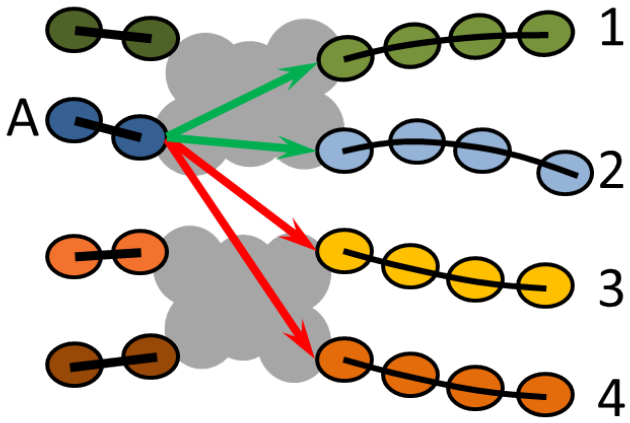


Figure 15: The assumption for online learning an affinity model is that associations filtered by occlusion sub-tunnels (red arrows) can be used to learn which remaining unfiltered associations (green arrows) are incorrect matches. For example, the filtered association between tracklet A and 3 may be similar in feature space to the unfiltered association between A and 1. This will allow the trained model to learn that the association between A and 1 is incorrect and take the association between A and 2.

bagged together for multiple instance learning. We propose two methods for generating positive and negative bags which satisfy the MIL bag constraints listed prior as shown in Figure 16.

- Occlusion Level - For each occlusion sub-tunnel O_s, all unfiltered associations within the occlusion sub-tunnel ($a(i,j) \in O_s$) are placed into a positive bag, B_i. All filtered associations which involve a tracklet in O_s ($a(i,j)|T_i \in O_s$ or $T_j \in O_s$ and $a(i,j) \notin \{O_s\}$) are placed into a negative bag, N_i.

- Tracklet Level - For each tracklet T_i, the set of filtered associations are placed into a negative bag, N_i, and all of its unfiltered associations placed into a positive bag, B_i.

The two bagging methods provide different types of bag compositions which can

have a dramatic effect on the learned affinity model. The occlusion sub-tunnel bags generate fewer bags overall but each bag will contain large number of samples. In contrast, tracklet level bags will generate a larger number of bags with each bag containing fewer samples. In particular concern to learning an affinity model is the existence of positive bags which do not contain any positive associations which we call "bad" positive bags. Bad positive bags occur when all the samples inside a positive bag relate to a single tracklet and the tracklet either 1) is a false positive tracklet which should not link to anything or 2) the tracklets correct link occurs at a larger gap size (e.g., when training an affinity model at the first stage with a maximum allowed gap of 8 frames, this tracklet's correct link is 10 frames in the future). The composition of the bags (number, size and presence of "bad" bags) can have a dramatic effect of the classification boundary learned. We discuss the effects of each bagging method on one of our datasets in the results section.

6.2 Multiple Instance Learning Methods

Once the bags are generated, a multiple instance learning method can be used to learn an affinity model. We test four different multiple instance learning methods for training. They are as follows:

- mi-SVM - This is the support vector machine based multiple instance classification method of [1]. Specifically, this is the mi variant of the training algorithm which is detailed in Algorithm 1. This is an iterative approach to solving the MIL problem while using the standard SVM model as a sub-routine. Initially, all instances are given the same label as their bag and used to train a standard

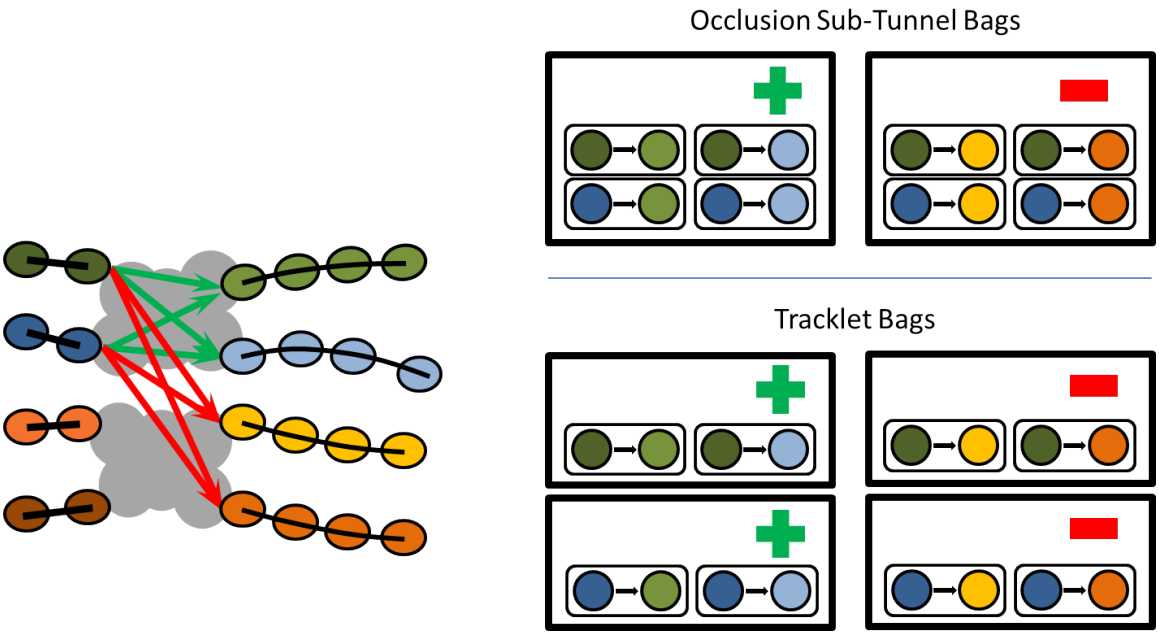


Figure 16: The left images shows the labeled associations for using occlusion sub-tunnels where red indicates a negative label and green indicates a weak positive label. On the right are the results of the two bagging methods. On top, weakly positive associations are grouped together by occlusion sub-tunnels (bags indicated by dark rectangles). Tracklet bags are grouped by individual tracklets.

SVM model. Next, labels for instances in positive bags are updated based on the trained SVM. Instances in positive bags which are classified as negative by the model are given a negative label and instances in positive bags which are classified as positive by the model are given a positive label. At least one instance in each positive must have a positive label, therefore if all instances in a given positive bag are classified as negative by the model the "most positive" instance in the bag is given a positive label Labels for instances in negative bags do not change. A new SVM model is then trained on the instances with their updated labels. This is repeated until labels labels on instances stop changing. We use libSVM [10] to solve for the support vectors at each iteration.

Input: Training Bags, $B^i = \{x_j, y_j\} \& N^i = \{x_j, y_j\}$; Bag Labels, $Y^i \in \{1, 0\}$
set $y_j = Y_i$ where $y_j \in B_i$
repeat
  train SVM model, f, using current labels, y_j
  foreach $x^j \in B_i$ *such that* $Y_i = 1$ **do**
    $y_j = sgn(f(x_j))$
  end
  foreach *positive bag* B_i **do**
    if $\sum_{y_j \in B_i} (1 + y_j)/2 == 0$ **then**
      find $x_j^* = \underset{x_j \in B_i}{\operatorname{argmax}} f(x_j)$
      set $y_j = 1$
    end
  end
until *labels* y_i *do not change*
Output: SVM model f

Algorithm 1: mi-SVM

- MIC - Another multiple instance classification problem by [39]. Rather then using a heuristic to solve for the support vectors, MIC formulates the multiple instance learning problem as a bilinear problem. First, we assume we are given m training bags consisting of k positive bags $\{B^1, \ldots, B^k\}$, and $m - k$ negative bags $\{N^{k+1}, \ldots, C^m\}$. the positive and negative bags are represented as matrices $B^i \in R^{m^i \times n}$ and $N^i \in R^{m^i \times n}$ respectively, where each row of the matrix represents a sample in the bag $B_l^i \in R^n$ and m^i is the number of samples in the bag. The non-linear separating hyper plane is represented denoted as

$$K(x', H')u = \beta \tag{8}$$

where $u \in R^m$ is a dual variable and the $m \times n$ matrix H is defined as $H' = [B^{1'}, \ldots, B^{k'}, C^{k+1'} \ldots C^{m'}]$ and $K(x', H')$ denotes a kernel map function. For example, if a linear kernel is used then $K(x', H') = x'H'$. The problem for

solving the classification problem is formulated as follows:

$$
\begin{aligned}
\min_{u,\beta,\gamma,v^i,\ldots,v^k} \quad & \upsilon e'\gamma + ||u||_1 \\
\text{s.t.} \quad & v^{i'} K(B^i, H')u - \beta + \gamma_i \geq 1, \qquad i = 1, \ldots, k, \\
& -K(N_l^i, H')u + \beta + \gamma_l^i >= 1, \quad i = k+1, \ldots, m, l = 1, \ldots, m^i, \\
& e'v^i = 1, \quad v^i \geq 0, i = 1, \ldots, k, \\
& \gamma \geq 0
\end{aligned}
\tag{9}
$$

Here, the vector γ with components $\gamma_i, i = 1, \ldots, k$ and $\gamma_l^i, k = k+1, \ldots, m, l = 1, \ldots, m^i$ represents the non-negative slack variables that are driven towards zero by the object function and e denotes a vector of ones. The vectors $v^1, \ldots, v^k$ are denote the convex combination of points within positive bags such that $v^i \in R^{m^i}$. Finally, the value υ is the Cost Factor which measure the trade off between the model error and generality of the learned model.

Solving this optimization problem is done iteratively in a bilinear manner. First, the the convex combination vectors, v^i are fixed and the solution is optimized for β, γ, and u. Next, u is fixed and the solution is optimized for β, γ, and the combination vectors v^i

One issue with this formulation is it's scaling with large numbers of samples as the kernel matrix grows quadratically. To solve this issue, we implemented the reduced support vector machines approach to drastically decrease the memory requirements during runtime [34]. This method has shown that a set of samples

can be sub selected as potential support vectors rather than having to use all samples as potential support vectors. This reduces the the matrix H from $R^{n\times\bar{m}}$ to $R^{n\times\hat{m}}$ where $\hat{m} \ll \bar{m}$. We implemented this method in Matlab using the open source COIN-OR Linear Programming library (CLP) to solve the optimization subroutines [37].

- MIRank - A ranking variant of the multiple instance problem by [6]. Rather then trying to classify bags, the goal is to determine which bags or instances are preferred to other bags. To do this, sets of bags are placed into boxes, where one bag is "preferred" to all other bags within the box. For our method, bags which are generated from the same occlusion tunnel or tracklet are placed within a box. We denote these boxes by ordered pairs (I, J) which indicates bag B^i is preferred to bag C^j.

  Following the same notation as MIC, the formulation for MIRank is as follows:

$$\begin{aligned} \min_{u,\beta,\gamma,v^i,\ldots,v^k} \quad & \upsilon e'\gamma + ||u||_1 \\ \text{s.t.} \quad & v^{i'}K(B^i, H')u - K(N_l^j, H')u \geq 1 - \gamma_{I,J}, \quad \forall (I,J) \\ & e'v^i = 1, \quad v^i \geq 0, i = 1, \ldots, k, \\ & \gamma \geq 0 \end{aligned} \tag{10}$$

  This is solved in the same bilinear fashion as MIC. Again, we implemented this algorithm within Matlab using CLP library and also used the reduced support vector machine algorithm to reduce memory requirements.

- MIHybrid - This is a hybrid approach of ranking and classification which we have devised. This combines the constraints of both MIC and MIRank. During optimization, we weight the errors from the MIC and MIRank formulations such as done in HybridBoost.

6.3 Evaluation

In this section, we evaluate our online affinity model learning method. First, we describe some implementation details for our online learning method. Next, we describe the methods we compare against, this include transferring offline trained models as well as a number of online appearance model training methods. Finally, we discuss the results in Section 6.3.3.

6.3.1 Implementation Details

For the described online method, an affinity model is trained for each stage of tracking just as was done for the offline training method. All four of our MIL methods are support vector machine based and we use the same parameters across each model. We use the radial basis function (RBF) for our kernel function and set the Cost Factor to 75 (found empirically).

6.3.2 Comparison Method

We compare our online training method to the following:

- Offline HybridBoost Model - This is the same method as used in Chapter 5 including irregular motion features and filtering of associations using occlusion sub-tunnels. First, we compare our method to the scenario described at the start

of the Chapter where training data consisting of the same dataset is unavailable. We train an affinity model using videos in the ant dataset and apply it to videos in the termite dataset and vice versa. Finally, we compare when training and testing using videos from within the same dataset as done in prior chapters.

- OLDAM - This is the Online Learned Discriminative Appearance Model (OLDAM) method of [32]. First, labels for detection pairs are generate by the observation that tracklets which overlap temporally cannot belong to the same tracklet. Two detections which belong to the same tracklet are labeled as a positive pair, and detections pairs from separate tracklets which overlap temporally are labeled negative. Each detection is represented by a collection of RGB Histograms, Histogram of Oriented Gradients [13], and Covariance texture features sampled from a set of discrete parts locations within the detection. A classifier is trained on a vector of similarity scores between each detections feature representations (e.g., the correlation coefficient between the histogram of oriented gradients feature at part i of each detection is an element of the vector). The classifier is then applied at run time as the appearance affinity component of the affinity score. This is combined with a manually tuned linear motion model based on motion smoothness.

- PIRMPT - This is the Person Identity Recognition based Multi-Person Tracking (PIRMPT) of [33]. Rather than training a single global classifier to determine if pairs of detections belong to the same object as done in OLDAM [32], PIRMPT trains a single classifier for each tracklet that meets a minimum length threshold.

For tracklets which do not meet the minimum threshold, a single classifier is made combining the training labels generated by these tracklets. Labels for detection pairs are collected using the same methodology of OLDAM. Again, this is coupled with a manually tuned linear motion model to form the full affinity score.

- MILPIRMPT - This method further extends PIRMPT by incorporating more temporally distant detection pairs for training [61]. Many tracklets in PIRMPT often suffer from having few positive training samples with minimal diversity causing the learned affinity models to over fit. For each tracklet, MILPIRMPT employs a motion heuristic to generate a list of potential matching tracklets in the future. The pairs of detections which contain one detection from the original tracklet and one detection from a potential match, are given a weak positive label and placed in a positive bag. Negative detection pairs a collected using the same temporal overlap method as OLDAM and PIRMPT. Multiple instance learning is then used to train a classifier for each tracklet which exceeds a minimum length threshold. Following PIRMPT, a single classifier is trained and used by all the tracklets which do not meet this minimum length. Following PIRMPT, this is coupled with a manually tuned linear motion model to form the full affinity score.

6.3.3 Discussion of Results

Comparison of MIL and Bagging Methods - First, we discuss the affect of the two bagging methods proposed and the multiple instance learning methods on tracking

Table 7: Composition of bags generated by occlusion sub-tunnels as described in Section 6.1 occlusion sub-tunnel level (OsT) and tracklet level (Tlet) from a video in the ant dataset. OsT level bagging produces fewer, larger bags where Tlet level bagging produces a much larger number of smaller bags.

Bag Method	# of Pos Bags	Median Bag Size	% Pos Samples in Bag	% "Bad" Bags
OsT	802	13.59	84.1%	7.5%
Tlet	4696	2.32	82.4%	10.3%

performance. First, we discuss the composition of the bags generated by each method. The choice of the bagging method can have a large effect on the performance of the learned affinity model as our MIL method of choice can be sensitive to bag composition. Table 7 shows how the weakly labeled positive samples out of the occlusion sub-tunnels are broken into bags. Bagging based on occlusion sub-tunnels generally creates fewer but larger bags where bagging based on tracklets produces more but smaller bags. The presence of "bad" positive bags or positive bags which do not contain any positive samples. The occlusion sub-tunnel bagging method reduces the chances of such bags existing as each bag often contains associations between many pairs of tracklets.

Table 8 shows the tracking performance when using each of the bagging methods and multiple instance learning methods described. As shown in the table, tracklet level bagging performs better overall for all training methods with only an increase in ID switches. The MIL methods produce fewer ID switches while using occlusion sub-tunnel bagging, but at the expense of far more fragments and a reduction in recall. For example, when using the mi-SVM methods, occlusion sub-tunnel bagging results in 35% fewer ID switches than tracklet bagging but results in 40% more fragments

and one more mostly lost target. Using MIRank the difference is more significant with occlusions sub-tunnel bagging resulting in 122% more fragments, 8 fewer most tracked targets and 3 additional mostly lost targets compared to tracklet bagging. The mi-SVM method is the least sensitive to the bagging method of the tested MIL methods having the smallest variation between the results of the two bagging methods. This is mostly due to the fact that MIC, MIRank, and MIHybrid represents each positive bag as a single point, therefore in occlusion sub-tunnel bagging there exist fewer positive points for solving the optimization routines while the number negative samples remains the same. From the results, we see that the MIL methods perform better with a larger number of smaller bag even with the increased number of "bad" positive bags. Based on these results, the results of mi-SVM with tracklet level bagging are used as the final results for our proposed online training method for the remainder of the evaluation section.

Tracking Performance - Next, we assess the effectiveness of our online learning method compared to training a HybridBoost model using video from a different dataset than the testing video. The results from offline training a HybridBoost affinity model with 5,000 frames and 10,000 frames of video from one data set and then testing on the videos in the other dataset are shown in Table 9. Our online learning method (using the mi-SVM method with tracklet bagging) produced 17% fewer ID switches and only 1 additional fragment compared to offline training an affinity model using the termite dataset and testing on the ant dataset. When the training and testing sets are reversed, our online training method resulted in 21% fewer ID switches and 4% fewer fragments. When the amount of data provided for training HybridBoost

Table 8: Comparison of the two bagging methods described in Section 6.1 Occlusion Sub-Tunnel (OsT) level bagging and Tracklet (Tlet) level bagging on the ant dataset using the four described MIL methods. Tracklet level bagging provides overall better performance with more ID switches than occlusion sub-tunnel bagging for all methods which results in fewer ID switches but a greater number of Fragments and lower recall.

Bag Method	MIL Method	Rec	Prec	FAF	GT	Frag	IDS	MT	PT	ML
OsT	mi-SVM	61.2%	99.0%	0.301	144	470	64	54	97	15
Tlet	mi-SVM	61.7%	99.0%	0.304	144	336	99	54	98	14
OsT	MIC	60.1%	98.8%	0.350	144	691	42	48	102	16
Tlet	MIC	61.7%	98.8$	0.381	144	409	116	54	97	15
OsT	MIRank	59.6%	98.7%	0.383	144	721	30	46	103	17
Tlet	MIRank	61.7%	99.0%	0.313	144	324	115	54	98	14
OsT	MIHybrid	59.6%	98.8%	0.354	144	711	27	46	104	16
Tlet	MIHybrid	61.6%	98.9%	0.346	144	371	118	54	98	14

Table 9: Comparing the tracking performance of our proposed online learning method versus transferring an offline learned affinity model to a different dataset (HB). For the ant datset, the HybridBoost model was trained using video from the termite dataset and vice versa. Our online method produces 18% fewer ID switches and only one additional fragment on the ant dataset when training with 5k frames of video. On the termite dataset, our method produces 3% fewer fragments and 21% fewer ID switches. When using 10,000 frames to train the offline model, our method still produces 4% fewer ID Switches and only 4% additional fragments while requiring no manual labels for training on the ant dataset. On the termite dataset, our method produces 27% fewer ID switches and only one additional fragment.

Dataset	Method	Recall	Prec	FAF	GT	Frag	IDS	MT	PT	ML
Training with 5k Frames										
Ant	HB	61.7%	98.8%	0.382	166	335	120	55	97	14
Ant	Proposed	61.7%	99.0%	0.302	166	336	99	54	98	14
Termite	HB	94.3%	94.6%	1.174	44	78	26	39	5	0
Termite	Proposed	94.6%	94.6%	1.180	44	76	20	39	5	0
Training with 10k Frames										
Ant	HB	61.7%	98.8%	0.384	166	322	103	54	98	14
Ant	Proposed	61.7%	99.0%	0.302	166	336	99	54	98	14
Termite	HB	94.5%	94.6%	1.181	44	75	28	39	5	0
Termite	Proposed	94.6%	94.6%	1.180	44	76	20	39	5	0

Table 10: The tracking performance of the online learning approaches on each dataset.

Method	Recall	Prec	FAF	GT	Frag	IDS	MT	PT	ML
Ant Dataset									
OLDAM	60.6%	98.7%	0.320	166	476	110	50	99	17
PIRMPT	60.0%	98.6%	0.419	166	624	112	49	99	18
MILPIRMPT	57.6%	99.0%	0.289	166	689	35	44	101	21
Proposed	61.7%	99.0%	0.302	166	336	99	54	98	14
Termite Dataset									
OLDAM	91.5%	94.6%	1.151	44	73	10	39	4	1
PIRMPT	90.5%	94.6%	1.143	44	128	16	39	4	1
MILPIRMPT	86.1%	94.3%	1.138	44	225	16	39	3	2
Proposed	94.6%	94.6%	1.180	44	76	20	39	5	0

increased to 10,000 frames, our online learning method still resulted in 4% fewer ID switches but increased fragments by 4%. When testing on the termite dataset, our the online training method still outperformed the HybridBoost model with 27% fewer ID switches. The poor performance when transferring is due to two main reasons: 1) difference in motion behavior and 2) difference in appearance reliability. However, the largest difference is in appearance. Because the ants are painted, appearance is more reliable in this dataset, but the learned reliance on appearance does not transfer over to the unmarked low resolution termite videos. Our method is able to learn the best set of metrics for each video online.

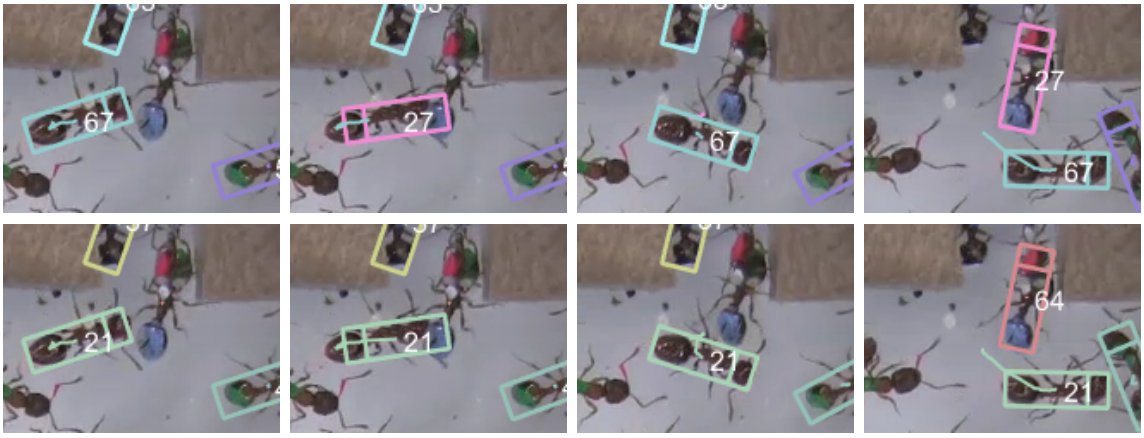
Our method also outperforms previous online appearance learning approaches (Table 10). On the ant dataset, our method reduced fragments by 41%, ID Switches by 11% and increased mostly tracked targets by 4 over the best performing online appearance learning method, OLDAM. The performance of the prior online training methods is primarily due to their reliance on manually tuned linear motion model. As we demonstrated in Chapter 4, linear motion models are not well suited to tracking

Table 11: Tracking performance (with standard deviation) on one ant video with varying amounts of training labels per tracking stage.

# Labels per Stage	Proposed (0)	200	400	600	800	1000	All
Frag	140	178.3 ± 54.14	123.9 ± 22.4	125.3 ± 16.9	118.3 ± 11.3	115.5 ± 14.7	99
IDS	49	89.9 ± 57.7	66.1 ± 17.7	53.7 ± 10.9	54.4 ± 8.7	52.9 ± 5.9	47

in insect videos and even though the ants are painted, appearance is not a reliable feature within this dataset. Figure 17 and Figure 18 show instances where the linear motion model used by prior were insufficient and lead to either an ID switch or fragment, but our proposed method was able to learn a motion model and prevent these errors. However, our proposed method is able to learn an affinity model which incorporates our irregular motion features. On the termite dataset, our method outperformed both PRIMPT and MILPRIMPT by reducing fragments by at least 68% and mostly lost targets by at least 2 while increasing ID switches by only 4. Compared to OLDAM, our method does increase both fragments and ID switches by 14% and 58% respectively, but OLDAM achieves these at the expense of mostly lost targets and recall.

Finally, we determine the amount of manual labels our method is able to save. Table 11 shows the tracking performance on one ant video when using varying amounts of labeled data for each tracking stage. Our method maintains fewer ID switches than with training and offline model with 1,000 manually labeled samples per stage (or 4,000 labels total). For fragments, our method performs performs within the



Frame 1030 Frame 1035 Frame 1090 Frame 1130

Figure 17: The tracking results of the OLDAM method are shown in the top row. At frame 1035, the detection becomes noisy and throws off the learned appearance model, thus creating a fragment Track 27. This fragment is later associated to a different ant which contains many of the colors captured by the noisy detection. Our online affinity model learning method (bottom row) correctly tracks over the entirety of the frames.

standard deviation of using 400 tracking samples per stage (or 1,600 samples total). However, our method does not require any manually labeled samples and trains solely on automatically generated weak labels.

Frame 2135 Frame 2290 Frame 2425 Frame 2520

Figure 18: The top row shows the results of the OLDAM tracking method. The linear motion model is unable to cope with the almost 400 frame occlusion and generates two fragments. Our method is able to learn a motion model which correctly associates the tracklets over the long gap preventing two fragments.

CHAPTER 7: CONCLUSION

In this dissertation, we have described three contributions to multi-object tracking within videos containing frequently occluding and similar appearing objects. First, we introduced new irregular motion features based on correlated random walks to improve motion prediction over long occlusions. These motion features were able to reduce ID switches by up to 57% and fragments by up to 31% in insect videos. Next, we formulated a spatial constraint based on occlusion which can filter a large number of incorrect associations between tracklets called occlusion sub-tunnels. By filtering many potentially confusing incorrect associations, ID switches were reduced by 22% in insect videos. However, occasionally poor foreground responses could lead to errors in the occlusion sub-tunnel construction which lead to an increase in fragments (up to 14% in the termite dataset). Finally, we introduced a method to online train an affinity model by leveraging the filtering ability of occlusion tunnels. This new method is capable of learning both appearance and motion features as compared to previous online learning methods which were limited to learning appearance only. Our described method outperforms the previous online learning methods in almost every metric on the ant dataset with 41% fewer fragments and 11% fewer ID switches than the next best performing online method.

There a few areas for future work within each of these methods. Additional motion features can be explored to handle extremely long gaps between tracklets. More

importantly would be exploring features which can account for drastic changes in speed. Many of the remaining fragments and ID switches are due to objects either suddenly stopping after longs periods of motion or vice versa.

Within occlusion sub-tunnels, work needs to be done to work around periods of poor foreground blob recall. For example, our current method is unable to handle static scene occluders (e.g., a pillar in a pedestrian video which objects move behind). Being able to handle such situations is required for extending this method to other domains.

Finally, for the online affinity model learning method a challenge for learning an model with MIL is the presence of "bad positive" bags. These are positively labeled bags of instances which do not actually contain a correct association. Currently, approximately 10% of the positive bags generated fall into the "bad positive bag" category. Methods to either identify and remove these bags before training could greatly improve performance. Another alternative is to adapt the multiple instance learning methods directly to account for this type of noisy data.

REFERENCES

[1] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems*, pages 561–568, 2002.

[2] A. Andriyenko and K. Schindler. Multi-target tracking by continuous energy minimization. In *Computer Vision and Pattern Recognition, 2011, IEEE Conference on*. IEEE, 2011.

[3] Y. Bar-Shalom and C. Brown. *Tracking and Data Association*. Academic Press, Inc., 1988.

[4] H. Ben Shitrit, J. Berclaz, F. Fleuret, and P. Fua. Multi-commodity network flow for tracking multiple people. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(8):1614–1627, 2014.

[5] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(9):1806–1819, 2011.

[6] C. Bergeron, J. Zaretzki, C. Breneman, and K. P. Bennett. Multiple instance ranking. In *Proceedings of the 25th international conference on Machine learning*, pages 48–55. ACM, 2008.

[7] R. Bise, Z. Yin, and T. Kanade. Reliable cell tracking by global data association. In *Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on*, pages 1004 –1010, 30 2011-april 2 2011.

[8] T. J. Broida and R. Chellappa. Estimation of object motion parameters from noisy images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):90–99, 1986.

[9] A. A. Butt and R. T. Collins. Multi-target tracking by lagrangian relaxation to min-cost network flow. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1846–1853. IEEE, 2013.

[10] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[11] D. Charbonneau, B. Blonder, and A. Dornhaus. Social insects: a model system for network dynamics. In *Temporal Networks*, pages 217–244. Springer, 2013.

[12] E. Codling, M. Plank, and S. Benhamou. Random walk models in biology. *Journal of the Royal Society Interface*, 5(25):813–834, 2008.

[13] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[14] A. Decourtye, J. Devillers, P. Aupinel, F. Brun, C. Bagnis, J. Fourrier, and M. Gauthier. Honeybee tracking with microchips: a new methodology to measure the effects of pesticides. *Ecotoxicology*, 20(2):429–437, 2011.

[15] D. Delannay, N. Danhier, and C. De Vleeschouwer. Detection and recognition of sports (wo) men from multiple views. In *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, pages 1–7. IEEE, 2009.

[16] A. I. Dell, J. A. Bender, K. Branson, I. D. Couzin, G. G. de Polavieja, L. P. Noldus, A. Pérez-Escudero, P. Perona, A. D. Straw, M. Wikelski, et al. Automated image-based tracking and its application in ecology. *Trends in Ecology & Evolution*, 29(7):417–428, 2014.

[17] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1):31–71, 1997.

[18] A. Feldman, M. Hybinette, and T. Balch. The multi-iterative closest point tracker: An online algorithm for tracking multiple interacting targets. *Journal of Field Robotics*, 29(2):258–276, 2012.

[19] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[20] J. H. Fewell. Social Insect Networks. *Science*, 301(5641):1867–1870, Sept. 2003.

[21] M. Fletcher, A. Dornhaus, and M. C. Shin. Multiple Ant Tracking with Global Foreground Maximization and Variable Target Proposal Distribution. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 570–576. IEEE, 2011.

[22] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *Oceanic Engineering, IEEE Journal of*, 8(3):173–184, 1983.

[23] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *The Journal of machine learning research*, 4:933–969, 2003.

[24] Y. Freund, R. Schapire, and N. Abe. A short introduction to boosting. *Journal of Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[25] A. Gelb. *Applied optimal estimation*. MIT press, 1974.

[26] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.

[27] C. Huang, B. Wu, and R. Nevatia. Robust object tracking by hierarchical association of detection responses. In *Proceedings of the 10th European Conference on Computer Vision: Part II*, pages 788–801. Springer-Verlag, 2008.

[28] M. Isard and A. Blake. Condensationconditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[29] P. Kareiva and N. Shigesada. Analyzing insect movement as a correlated random walk. *Oecologia*, 56:234–238, 1983.

[30] Z. Khan, T. Balch, and F. Dellaert. MCMC-based particle filtering for tracking a variable number of interacting targets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(11):1805–1819, 2005.

[31] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[32] C.-H. Kuo, C. Huang, and R. Nevatia. Multi-target tracking by on-line learned discriminative appearance models. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 685–692. IEEE, 2010.

[33] C.-H. Kuo and R. Nevatia. How does person identity recognition help multi-person tracking? In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1217–1224. IEEE, 2011.

[34] Y.-J. Lee and O. L. Mangasarian. Rsvm: Reduced support vector machines. In *Poc. SIAM Int'l Conf. Data Mining*, volume 1, pages 325–361, 2001.

[35] B. Leibe, K. Schindler, and L. Van Gool. Coupled detection and trajectory estimation for multi-object tracking. In *Computer Vision, 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[36] Y. Li, C. Huang, and R. Nevatia. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *Computer Vision and Pattern Recognition, 2009, IEEE Conference on*, pages 2953–2960. IEEE, 2009.

[37] R. Lougee-Heimer. The common optimization interface for operations research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1):57–66, 2003.

[38] D. Lowe. Object recognition from local scale-invariant features. In *iccv*, page 1150. Published by the IEEE Computer Society, 1999.

[39] O. L. Mangasarian and E. W. Wild. Multiple instance classification via successive linear programming. *Journal of Optimization Theory and Applications*, 137(3):555–568, 2008.

[40] D. P. Mersch, A. Crespi, and L. Keller. Tracking individuals shows spatial fidelity is a key regulator of ant social organization. *Science*, 2013.

[41] K. Okuma, A. Taleghani, N. De Freitas, J. J. Little, and D. G. Lowe. A boosted particle filter: Multitarget detection and tracking. In *European Conference on Computer Vision (ECCV) 2004*, pages 28–39. Springer, 2004.

[42] C. Poff, H. Nguyen, T. Kang, and M. Shin. Efficient tracking of ants in long video with gpu and interaction. In *Applications of Computer Vision (WACV), 2012 IEEE Workshop on*, pages 57 –62, jan. 2012.

[43] D. B. Reid. An algorithm for tracking multiple targets. *Automatic Control, IEEE Transactions on*, 24(6):843–854, 1979.

[44] L. Rice, A. Dornhaus, and M. C. Shin. Efficient training of multiple ant tracking. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, pages 117–123. IEEE, 2015.

[45] R. Rosales and S. Sclaroff. 3d trajectory recovery for tracking multiple objects and trajectory guided recognition of actions. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2. IEEE, 1999.

[46] D. Salvi, J. W. Waggoner, A. Temlyakov, and S. Wang. A graph-based algorithm for multi-target tracking with occlusion. In *Applications of Computer Vision (WACV), 2013 IEEE Winter Conference on, pages=489–496, year=2013.*

[47] F. Schweitzer, K. Lao, and F. Family. Active random walkers simulate trunk trail formation by ants. *BioSystems*, 41(3):153–166, 1997.

[48] T. D. Seeley. *Honeybee democracy*. Princeton Univ. Press, 2010.

[49] X. Shi, H. Ling, J. Xing, and W. Hu. Multi-target tracking by rank-1 tensor approximation. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2387–2394. IEEE, 2013.

[50] G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah. Part-based multiple-person tracking with partial occlusion handling. In *Computer Vision and Pattern Recognition, 2012, IEEE Conference on*. IEEE, 2012.

[51] S. Spurlock, J. Shan, and R. Souvenir. Discriminative poses for early recognition in multi-camera networks. In *Proceedings of the 9th International Conference on Distributed Smart Camera*, pages 74–79. ACM, 2015.

[52] S. Spurlock and R. Souvenir. Dynamic subset selection for multi-camera tracking. In *Proceedings of the 50th Annual Southeast Regional Conference*.

[53] M. Villamizar, F. Moreno-Noguer, J. Andrade-Cetto, and A. Sanfeliu. Efficient rotation invariant object detection using boosted random ferns. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1038–1045. IEEE, 2010.

[54] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition (CVPR), 2001 IEEE Conference on*, volume 1, pages I–511. IEEE, 2001.

[55] J. Wang and J.-D. Zucker. Solving multiple-instance problem: A lazy learning approach. 2000.

[56] S.-K. Weng, C.-M. Kuo, and S.-K. Tu. Video object tracking using adaptive kalman filter. *Journal of Visual Communication and Image Representation*, 17(6):1190–1208, 2006.

[57] B. Wu and R. Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *Computer Vision, 2005. (ICCV). Tenth IEEE International Conference on*, volume 1, pages 90–97. IEEE, 2005.

[58] B. Wu and R. Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, 75(2):247–266, 2007.

[59] H. Wu, B. Li, T. Springer, and W. Neill. Modelling animal movement as a persistent random walk in two dimensions: expected magnitude of net displacement. *Ecological Modeling*, 132(1):115–124, 2000.

[60] B. Yang, C. Huang, and R. Nevatia. Learning affinities and dependencies for multi-target tracking using a crf model. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1233–1240. IEEE, 2011.

[61] B. Yang and R. Nevatia. Multi-target tracking by online learning of non-linear motion patterns and robust appearance models. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1918–1925. IEEE, 2012.

[62] B. Yang and R. Nevatia. An online learned crf model for multi-target tracking. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2034–2041. IEEE, 2012.

[63] B. Yang and R. Nevatia. Online learned discriminative part-based appearance models for multi-human tracking. In *European Conference on Computer Vision (ECCV) 2012*, pages 484–498. Springer, 2012.

[64] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys (CSUR)*, 38(4):13, 2006.

[65] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *Computer Vision and Pattern Recognition, 2008, IEEE Conference on*, pages 1–8. IEEE, 2008.