BETTER MODELING OF MATCHING POSSIBILITIES AND UNCERTAINTY FOR OFFLINE VISUAL MULTIPLE OBJECT TRACKING

by

Lance Rice

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

2021

Approved by:

_____
Dr. Min C. Shin

_____
Dr. Minwoo Lee

_____
Dr. Gabriel Terejanu

_____
Dr. Robert Cox

©2021
Lance Rice
ALL RIGHTS RESERVED

ABSTRACT

LANCE RICE. Better modeling of matching possibilities and uncertainty for offline visual multiple object tracking. (Under the direction of DR. MIN C. SHIN)

The task of visually tracking multiple objects remains an active field of algorithm development even after several decades of research in the computer vision community. One reason it remains an active research area is that identifying and maintaining the location of multiple targets in a video recording can be approached from several perspectives depending on the application's needs. Another reason is simply that the general problem of automated tracking can be very challenging. Moreover, compared to tracking a single target, multi-target tracking can grow significantly more complicated. Similar appearances between different objects, crowded scenes, and inter-object occlusions among a sometimes unknown and fluctuating number of objects are additional difficulties in multi-object scenarios. Challenges such as these collectively manifest into three broader design decisions often faced by multiple object tracking (MOT) algorithms. First is how to handle what one could think of as "easy" and "hard" regions of a trajectory. The second is how to handle the sheer number of possible explanations of the data. The third is how do you model certainty. This dissertation aims to better model the uncertainty among possible answers to the tracking data in offline tracking scenarios (i.e., input is the entire video, not frame-by-frame). Furthermore, the method does so in a way that utilizes the information within the "hard to track" regions — information that is typically not used. The way we do this results in accurate tracking that is better suited for video analysis pipelines that may need to filter or correct any tracking errors that did occur.

DEDICATION

None of this would be possible without three very important people in my life:

Mary and Andy Rice — my mother and father — and Sarah Alexander.

I appreciate everything you have done for me.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Shin for all of his guidance, wisdom, and support and during my PhD studies. I would also like to thank the University of North Carolina at Charlotte's graduate school for their funding support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Bi-Forest Shorthand for Bi-directional hypothesis forest

HypTree Shorthand for hypothesis tree

LBP An acronym for loopy belief propagation

MOT An acronym for multiple object tracking

Na Shorthand for "No Answer"

SOT An acronym for single object tracking

Tracklet Term used to describe a trajectory fragment/segment

CHAPTER 1: INTRODUCTION

Currently, multiple object tracking (MOT) algorithms expedite several advanced methods of video analysis. To fully appreciate the applicability of MOT, it is important to realize that MOT algorithms are almost always a means to an end, rarely the end itself. Another way to state this is that tracking is often a single module within larger video analysis pipelines. For example, proxemics, which uses tracking to study how people use the spaces they work and live in [1, 2], personalized smart spaces [3], robotics [4], autonomous driving [5, 6, 7], traffic monitoring [8, 9], and security [10, 11].

The field of biology is another area of research that often employs automated tracking as part of analysis pipelines and could benefit significantly from further development within MOT. Evidence of this can be seen in the number of automated tracking software systems published for gathering animal tracking data [12, 13, 14, 15, 16]. For biologists, automatic tracking enables more thorough investigations towards understanding scenarios like subjects' response to stimuli [17], division of labor [18], and collective decision making within biological networks [19]. With this context in mind, we can think of the procedure for laboratory investigation on biological video recordings as three interdependent steps [20]: data acquisition, tracking, and analysis. Correct analysis results thus depend on the reliably gathering accurate representations from the tracking stage.

Under particular, often very strict, assumptions on the data, one may assert that tracking will not produce errors that would negatively affect the analysis. If these assumptions cannot be met, then the risk of errors in the tracking results should be expected. MOT is a very challenging problem, and without strict assumptions to

help narrow the range of possibilities, it quickly becomes unwise and impractical to believe tracking errors will not occur.

This raises the question: what can one do when the data does not fit the assumptions required to assert that tracking errors will not occur? One way would be to find another approach for gathering trajectory data that does not rely so much on tracking performance. For example, the video could be labeled completely manually by annotating all targets in the video from scratch. This is a very time-consuming process, and the methods proposed in this dissertation seek to provide a more efficient alternative. Another answer to this question is to find a way to filter or correct errors within the tracking results. But, to do so would mean identifying potential errors in the first place — a task that the tracking algorithm may be well suited for if formulated correctly.

This leads us to another question: how would a tracking algorithm identify potential errors in the results? One idea would be to look for abnormalities in the tracking results. Unfortunately, this assumes that some characteristic of the error would be able to be classified as abnormal. A better way would be to model possible explanations of the data (including the case that you missed a possible explanation), compare the favorability of each, and from this derive some measure of uncertainty. Furthermore, one can imagine how having the set of possible explanations in mind could be beneficial when requesting information to settle the uncertainty (e.g., user interaction). If we are going to identify potential errors in the tracking results automatically, we are essentially identifying uncertainty among possible explanations of the data.

As we've previously established, the situation involves a prerecorded video, and the information needed is accurate trajectory data. Cases such as these are well suited for what we will refer to as batch approaches to MOT. By batch, we mean the video is processed as a whole (i.e., not online/real-time, where frames are provided

sequentially). This aspect of the situation is noteworthy for two reasons: 1) it is an essential aspect concerning the scope of this work, but more so in terms of motivations, 2) if we wish to model uncertainty among possibilities, then the way the video is processed dictates the possibilities we can consider. Batch approaches are better suited for modeling possibilities across the entirety of a video simply because the input is the entire video.

Lastly, we should note that some portions of the video's trajectories are more difficult than others. This results in what we'll call "hard-to-track" and "easy-to-track" regions. Characteristics of the "easy-to-track" regions are high confidence detections and virtually no ambiguity as to whether or not two detections belong to a common target. We'll expand upon this more later in the dissertation. Still, the important thing to consider here is that the number of possibilities expands primarily due to the "hard-to-track" portions of the trajectories (depicted as the striped-red zones in Figure-2.2).

To summarize the points made up to this point that motivates this work — there are situations where individuals have collected video data and wish to use tracking to enable further analysis. Still, the data does not meet the requirements to assume tracking will be reliable for analysis. In this situation, it would be beneficial if the tracker could aid with identifying potential errors. For a tracker to do this, it will need to model and compare the possible explanations of the data such that some measure of uncertainty is available.

This dissertation ultimately proposes several contributions specific to multi-object tracking (MOT) but first outlines a single object tracking (SOT) approach. The proposed SOT approach models possible routes a target could have taken within the "hard-to-track" regions (depicted as the striped-red zones in Figure-2.2) by representing the hypothetical routes as branches in a tree-like data structure. These SOT trackers, which we refer to as hypothesis trees, are grown in a bi-directional fashion to

create a forest of trajectory possibilities. We then use this forest of hypothesis trees, which we'll refer to as a Bi-forest, to assist with constructing and scoring elements of the tracklet (i.e., trajectory-fragmented) matching graph. Additionally, we propose an approach that converts the tracklet matching graph into a probabilistic tracklet matching factor graph. The tracker can then estimate matching marginals and, in turn, use the estimated marginals to assemble a matching solution. Finally, we propose a procedure to more efficiently identify and correct tracklet matching errors. The proposed work leverages the estimated matching marginals and the Bi-forest to reduce correction time. In summary, the primary contributions of this dissertation are:

- Hypothesis forests for capturing the spatial and temporal extent of possible paths a target has taken, as well as the relative potential among these paths. These structures aid with tracking in several ways. Uses a modified Siamese network for appearance modeling during difficult regions within the MOT setting (i.e., we expect objects with similar appearances to be nearby the target).

- An approach for constructing the tracklet matching graph which determines both the set of possible connections and their affinity scores using bi-directional hypothesis forests. Additionally, a prediction-based matching procedure that does not require an optimization procedure for constructing a matching solution is presented.

- A method for converting tracklet matching graph into a factor graph which can be used together with a loopy-belief propagation (LBP) message passing approach to estimate matching marginals and construct greedy matching solution.

- A correction procedure that reduces user correction time by utilizing matching marginals and possible paths within the hypothesis forests.

Relevant background information concerning MOT, evaluation, and datasets is provided in Chapter-2. The procedure for growing SOT hypothesis trees is given in Chapter-3. The construction of Bi-forests, as well as details on how tracklet matching information is extracted from them, is presented in Chapter-4. The method for estimating matching marginals and constructing a greedy solution is detailed in Chapter-5. The proposed correction procedure is given in Chapter-6. Finally, conclusions and future works are discussed in Chapter-7.

CHAPTER 2: BACKGROUND

It is helpful to introduce two broad dividing lines in terms of approaches when discussing visual tracking. The first division is sequential vs batch methods — the distinction between the two concerns how observations are gathered from the input video. By observation, we mean anything gathered from the image concerning the targets of interest, like image patch samples or detection responses. Sequential methods are designed to process a small window of observations, often frame-by-frame, and sequentially slide the observation window across the video to construct trajectories. Batch methods utilize a large window, often all video frames, and typically express object tracking as a combinatorial optimization problem to build trajectories. Sequential methods are usually shooting for real-time processing speeds, and batch methods tend to be more accurate given they have more available to work with, but that's not always the case. The second division is straightforward and concerns whether the method is a single object tracker (SOT) or a multi-object tracker (MOT). This dissertation focuses mostly on MOT approaches with a few exceptions in chapter-3. Several surveys exist for SOT trackers, including SOT trackers priors to the deep learning wave [21] and more recent surveys [22, 23, 24]. After discussing the broad set of challenges related to tracking, we then shift attention to data association-based MOT (section-2.2) and the essential elements within it that formulate the tracking problem. Section 2.3 describes input to the proposed method (i.e., detections and low-level tracklets), and section- details the ant colony dataset used during evaluations.

2.1 MOT tracking challenges

A tracker has to be prepared for several categories of challenging dynamics when determining or comparing possible explanations of the data. Even further difficulties can be faced when these categories co-exist in particular ways. Here, we partition the challenges into four categories: object dynamics, recording dynamics, situational factors, and computational constraints. Object dynamics refers to the characteristics of the targets in the scene and how these characteristics possibly change over time. Appearance is an example of object dynamics, including differentiating objects by appearance alone and the degree of visual change in an object's appearance over time (independent of other challenges, like occlusion). Motion is another example of object dynamics and concerns the movement patterns of an object. Motion patterns can range from predictable (e.g., smooth movement, like pedestrians) to abrupt (e.g., flight paths of fruit flies).

Situational factors include the video background and the density of objects in the scene. A few challenges can result from the scene's background that causes distractions or confusion when trying to locate a target. The background is a critical factor for several prior works because they heavily rely on accurate foreground estimation (i.e., pixels containing a target of interest) [25, 26, 16]. Occlusions and object density are two closely related examples of situational factors. Density impacts occlusions, but the activities the objects are exhibiting, and the specifics of the scene can also affect the degree of occlusions. Roughly speaking, occlusions can be inter-object (e.g., object occludes object), object-to-scene (e.g., walks behind a wall), or scene-to-object (e.g., garage door shuts, visually blocking object).

There are several challenges related to recording dynamics. Things such as camera perspective, camera movement, and imaging quality (e.g., frames per second and resolution) can negatively affect tracking performance if assumptions made by the tracking method are not met. The primary assumption in this work concerning recording

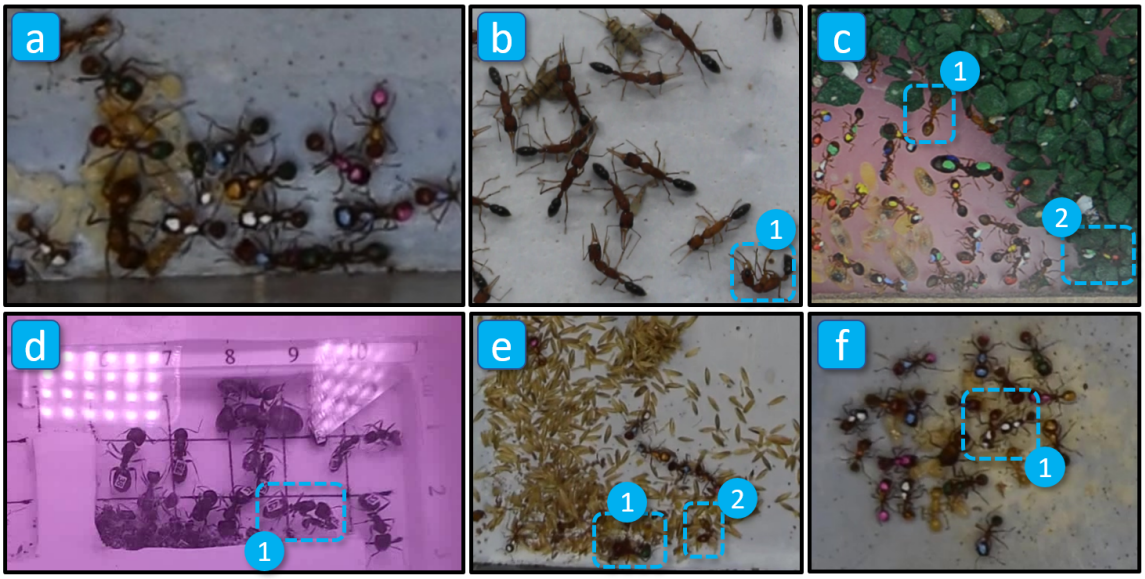

Figure 2.1: An illustration of tracking challenges with specific examples on the ant colony dataset. All of the examples, except for [e], have some degree of occlusion ranging from mild [b] to severe [a,c,d]. We can also see an example of nearby objects with similar appearances [f-1] and a case where we cannot assume distinct appearances [b]. Background clutter [c, e] can cause false-positive detections (Note that [e-2] is not an ant, [e-1] is an ant). Some videos have uncommon interactions between objects; for example, [d-1] highlights one ant carrying a dead ant. Other challenges include large pose variations [b-1], appearance variations (object in [c-1] has paint marks that are not visible due to the object being upside down while crawling on the underside of plexiglass cover), and difficult lighting ([d] has dark areas within the cluster while also having lights reflecting from plexiglass cover).

dynamics is that the camera is static. Finally, computational constraints are another concern of tracking methods. The challenges we just outlined, together with the fact a video contains a great deal of structured information by itself, amount to many possible explanations to deal with noisy, missing, or misleading information. This large set of possibilities only adds to the computational burden already felt by batch MOT methods that wish to use more advanced models of appearance and motion. Figure-2.1 illustrates several of the challenges outlined in this section with examples taken from ant colony datasets.

2.2 Data association and batch-based MOT

Roughly speaking, the typical approach to batch-based MOT approaches is to gather detections within the video frames, determine some set of connections between the detections that should be considered, calculate affinity measures, and solve for the final trajectories. With this perspective, detections form the groundwork for later stages of inference. Trackers will possibly need to deal with some amount of missing, noisy, or false detections. Missing and noisy detections are usually due to the "hard-to-track" regions of the video. Detections can be gathered in several ways, but the trend in the recent past has been to employ deep learning-based detection approaches [27]. From here, detections are associated to form *tracklets* (fragments of a trajectory), possibly over multiple stages [28, 29, 26, 16]. We wish to highlight the key elements here: determining the set of possible connections, calculating affinities, and solving the matching problem. But, before we discuss these three elements of the tracklet matching problem, we should point out two terms that can cause some confusion if not well defined.

2.2.1 Head vs. tail endpoints

Here we introduce terms relied on throughout the dissertation. Since batch approaches consider all, or large portions of the video, we often need to discuss something

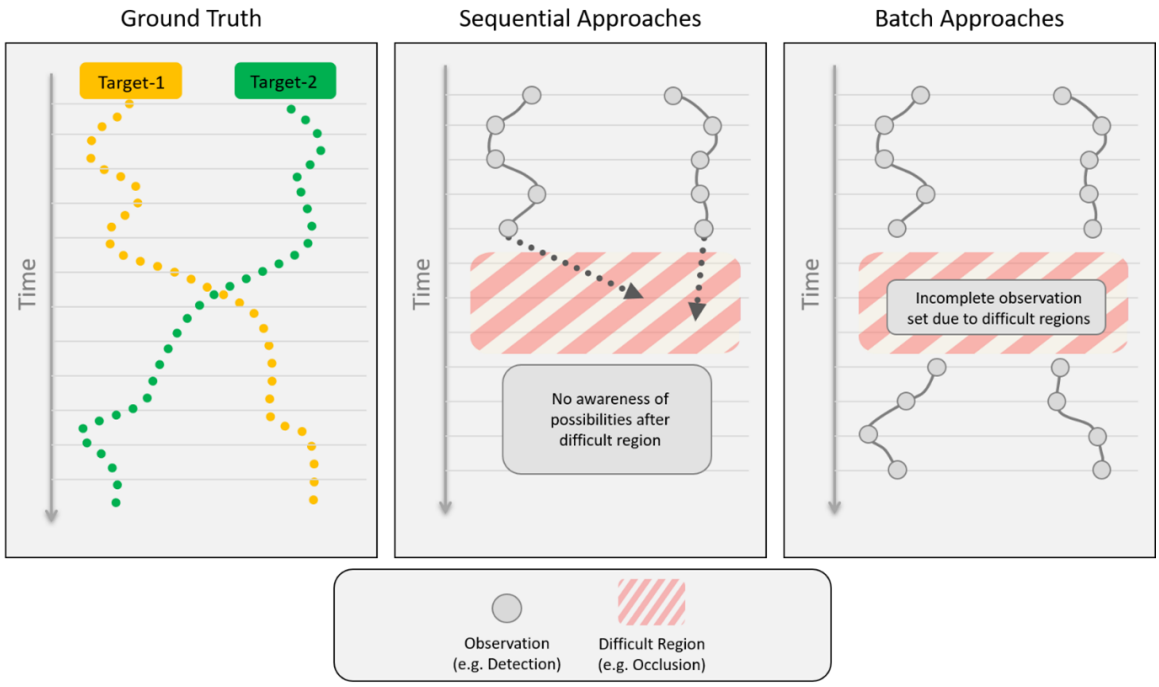

Figure 2.2: Illustration of hard regions of tracking when using a sequential or batch approach.

that has two ”endpoints” which define a temporal region. The two most predominant examples within this document are the two endpoints of a tracklet or a connection. We will refer to the *head* and *tail* endpoints as the endpoint that occurs earlier or later, respectfully. For example, if we assume some tracklet begins in frame one and ends in frame 10, then the head end of the tracklet would be the endpoint occurring at frame 1. With connections, we will be discussing the head and tail tracklets where the head tracklet of a connection is the tracklet that occurs earlier. Another way to look at it is that a connection head is always the tail of some tracklet and a connection tail is always the head of some tracklet. Keep in mind that we will sometimes refer to these endpoints in a compound manner because of the surrounding context (e.g., the tail endpoint of the connection's head tracklet).

2.2.2 Possible connections set

A critical step in batch-based MOT is determining which connections should be considered. If detections were perfect, meaning no detection information is missing, we could consider detections between adjacent frames. Unfortunately, detections will be missing to some degree within the "hard" regions of a trajectory. As such, a tracker is going to have to consider connection gap sizes larger than one. This concept of defining possible connections in terms of a maximum gap size is frequently used in MOT literature [29, 26, 28]. As the maximum connection gap size grows, so does the set of possible connections. This can cause an excessive number of connections, given that each possible connection will have to be scored before determining the matching solution. Several works follow a procedure like [29] and attempt to address these issues by performing matching over several stages. Here, each stage of matching considers connection gaps larger than the last stage. The difficulty with this approach is that affinity scores are largely relative, meaning there are times when it is hard to tell if a connection is correct without considering the other possible explanations. Furthermore, if a connection made in the previous stage is incorrect, then a "chain-reaction"-like situation can occur because correct explanations in later stages won't be considered.

Another means of determining possible connections is by using motion information. The simplest version, which is also used in sequential methods that account for missing information, is to define a gating region. Here, some maximum motion threshold, often simple displacement, defines a spatial radius of what should be considered. More often than not, this mechanism is used together with the maximum gap size approach mentioned. Thus, the shortcoming previously mentioned associated with using a gap size threshold still remain.

In this work, we propose an exploration-like approach such that sequential trackers, run both forward and backward from tracklet endpoints, are used to define what

should be considered in the possible connections. The process uses a gating mechanism within the SOT trackers but avoids thresholding connections based on a single predefined maximum gap size. In a way, this results in a dynamically defined maximum gap size, determined independently for each matching situation (i.e., tracklet endpoint). Furthermore, this allows us to measure how likely the correct connection is not represented (details in chapter-4).

2.2.3 Affinity scoring

Some measure of appropriateness is needed to determine a matching solution from the set of possible connections. This is often referred to as tracklet affinity or the association score. There are several methods proposed in the MOT literature for calculating tracklet affinities. Most prior works on estimating tracklet affinities rely on information that can be gathered from the "easy-to-track" regions. This includes appearance similarity measures (e.g., ReID [30]) and motion predictions (e.g.,motion smoothness [26] or advanced methods using learned models [31, 32]).

This dissertation proposes to estimate two forms of tracklet affinity. The first form of affinity is less conventional and uses predictions from forward and backward SOT trackers to develop a binary form of affinity (similar to [16]). This binary form of affinity is used to identify agreement between independent trackers such that the matching solution can be determined without an optimization step. The second form of affinity is more traditional in that it can be used within an optimization procedure. Here, we use a similarity measure between forward and backward SOT trackers to calculate affinity (details in chapter-4).

2.3 Low-level tracklets T_0

This section outlines input to the proposed tracklet matching process and our assumptions on the input. Specifically, input to the matching process is what we refer to as low-level tracklets T_0. These are short trajectory segments created by making

confident connections between consecutive frames of detections. The approach we use for building these tracklets follows [16] and can be summarized by the following. First, a user defines the location of all targets in three video frames, specifically the first, last, and middle frames. The procedure for defining these locations is what [16] calls user-marking. A simple to perform procedure of three clicks, together with a brush size, allows the user to define the location and spatial extent of a target quickly. These marks supply everything needed to tune all parameters related to foreground estimation, particle swarm optimization of the foreground, detection model training, and general size information on the targets. The foreground is estimated using background subtraction (again, assumes static camera) and particle swarm optimization. Foreground blobs that meet particular size requirements in each video frame are classified using a support vector machine. The foreground is also used to construct occlusion tunnels [26]. The foreground tunnels allow the system to stitch detections into short tracklets confidently. The process allows us to reliably assume no detections belonging to separate targets are connected. Furthermore, by using a two-stage detection classification procedure, we are able to assume false-positive detections are rare. While the precision of this process for both detecting and stitching is very high, recall suffers. Concerning stitching, this means trajectories are often very fragmented.

To reiterate, we have the following assumptions on the low-level tracklets: 1) false-positive detections are rare, meaning a high rate of false negatives is common, and 2) tracklet stitching will not mismatch detections from different targets, meaning trajectories are often very fragmented. These assumptions result in a matching situation where the tracker will need to fill in missing information rather than filter noisy detections and false positives.

2.4 Ant dataset

We use a challenging ant colony dataset throughout this dissertation for evaluating the proposed method. This dataset is well suited because it is a real-world use

Table 2.1: Ant colony dataset testing and training split. Ground truth time represents the amount of time it would take to watch each target in each of the videos from beginning to end. Ground truth time factors in video frame rate (Recording frame rate differs for each video — ranges from 25 to 60 frames per second).

	Num. Videos	Video Time	Total Video Frames	Ground Truth Tracks	Ground Truth Points	Ground Truth Time
Train set	12	33m 50s	55,887	540	2,376,132	24h 41m 52s
Test set	5	10m 34s	21,508	209	941,366	7h 29m 9s
Total	17	44m 24s	77,395	749	3,317,498	32h 11m 1s

case of the motivations made in the introduction– biologists captured these videos to analyze complex behavior in colonies of ants. Furthermore, ant colonies have several challenges that are not typically found in MOT benchmark datasets [33], which focus on pedestrian tracking. For one, the appearances of the ants within the dataset are very similar, and in some cases, practically identical. Secondly, the motion patterns of ants are much less predictable than tracking scenarios such as pedestrians [26]. A noteworthy consequence of these challenges is that typical affinity measures are less reliable. The reason is that these affinity measures almost entirely rely on discriminative features gathered from 'easy' to track regions. If appearance is assumed to be very similar, then features such as ReID are less effective. If the motion is difficult to predict, simple measures such as motion smoothness will also be less effective, especially over larger gap sizes. A few examples of the ant dataset are shown in figure-2.1.

CHAPTER 3: MODELING POSSIBLE TRAJECTORY PATHS WITH HYPOTHESIS TREES

This chapter presents a procedure for growing a tree-like data structure that models possible paths a target may have taken in video recording. While the process described in this chapter results in a single object tracker, the end goal is to apply it within a multiple object tracking setting — particularly the "hard-to-track" regions of a trajectory. We assume that "hard-to-track" regions will likely have several challenges, including heavy occlusions, appearance changes, abrupt motion, and distracting appearance similarities. The key concept is to leverage continuity of image information to capture debatable trajectory hypotheses and their relative feasibility when uncertainty is encountered. We first detail a Siamese network architecture for modeling target appearance and then outline the method for constructing hypothesis trees.

3.1 Appearance Modeling Siamese Networks

Siamese networks have seen a lot of attention in recent years within several areas of computer science research due to their ability to calculate similarities among sets of data [34, 35]. In this work, we use a fully convolutional Siamese neural network to calculate appearance similarity heatmaps. These appearance heatmaps, together with a simple displacement motion model, are the sources of information from which we determine continuity and uncertainty. Our Siamese network architecture follows an approach is similar to [36], which uses a three-part structural design including the network backbone, neck, and head. Note that [36] proposes and region proposal approach. For this research, we are interested in the classification heatmap output and

Figure 3.1: Example of Siamese network template, search instance, output. Here we show the template patch that has been rotated to a vertical alignment and sides masked using average color in patch (left), search instance with target of interest highlighted in red (middle), and output heatmap with target of interest highlighted in red (right). Notice that while many of the object responses are suppressed in the output heatmap, the highest response is not the target of interest but rather an isolated object with similar paint markings. Cases such as these motivate the need to consider distractions while tracking.

do not include the region proposal portions of [36]. Our backbone uses a Resnet34 [37], with reduced stride amounts (from 32 to 8), and center feature cropping on template features to reduce computation during the correlation (head module) step. The neck of our network compresses the feature dimension of both the template and instance from 256 to 64. The classification head of the network uses a depth-wise cross-correlation model [36].

As previously stated, this process will ultimately use this within a multiple object tracking setting. We'll need to consider three factors in an MOT setting, which are often less frequently seen in single object tracking datasets. First, we should expect objects similar in appearance to be nearby the target of interest. The second factor is that a square patch around a target will likely have distractions (e.g., parts of other targets). The third factor is that the orientation of the target can change over time. This last factor is more predominant in biological tracking datasets but certainly possible in any dataset. To deal with these, we propose several techniques to help the network during training and testing. We use a larger size ratio between template

and instance during training. We use a 5:1 size ratio between template and instance as opposed to a 2:1 ratio in [36, 38]. This allows the network to be exposed to many targets with similar appearances during training. To deal with ambiguities in the template, we ensure that the alignment of the target is vertical and mask the sides. This means that the network sees the target in a vertical alignment (although, could be either of the 180-degree orientations), and any potential visual distractions outside the target's rectangular extent are minimized.

An example is shown in figure-3.2. We modify the total weight of negative labels when calculating the binary cross-entropy, specifically to be half the total weight of the positive labels. This is because the network is seeing more negative examples in each training instance. For data augmentation, we follow the findings of [36] and perform random translations to prevent any center bias during prediction. Instance patches are randomly translated such that the target location is anywhere within ± 72 pixels from the center of the patch.

Selecting the right strategy for sampling during training can have a significant impact when learning a Siamese network [36]. We use a simple density measure when selecting the template and instance frame. We calculate this density as the number of objects within a certain radius of the target. We define this radius as two times the largest dimension of a target, specified by the ground truth. Densities are normalized for a target, and frames with lower density are more likely to be selected as templates, while higher densities are more likely to be used as search instances.

3.2 Trajectory Hypothesis Tree

The process of constructing the hypothesis tree involves three steps. We gather responses from the classification heatmap, then grow tree state with nearby responses and perform branch management. We repeat these steps until the stopping condition has been reached.

3.2.1 Gathering Heatmap Responses

The initial location of the target is provided as input and serves as the template location for appearance scoring with the Siamese network. At some later frame t, the location(s) of the target in frame $t-1$ are used to determine the search instance area. The size of the search area is based on the set of hypothesized locations in t-1 and an estimation of maximum movement. Here, we assume the maximum motion is provided as input. Later chapters will determine maximum motion automatically. The classification heatmap is computed given the template and search instance. The heatmap values are scaled to be within the range $[-1, 1]$ — these scaling parameters are automatically calculated while training the Siamese network. We reduce the number of possible locations considered within the heatmap by determining local maximums. We find the local maximum positions by performing image dilation, then comparing the original heatmap with the dilation and checking which pixels had no value change. The kernel size for dilation is set to be roughly half the smallest dimension of the target. We adjust the locations of the local maximums to better represent the surrounding evidence by calculating the center of mass within a radius of the response. This radius is equal to the size of the dilation kernel. Let $\mathbf{R^t}$ be the detected responses at frame t.

3.2.2 Tree Growth

Initially, the hypothesis tree contains a single leaf, the root leaf, which represents the initial location of the target. The initial location of the target is provided as input. For each subsequent time step t, the process for growing the tree given the set of detected responses $\mathbf{R^t}$ is as follows. For every leaf in the tree at t-1, child leaves are created for responses that are reachable by that leaf at time step t. The potential

of a child leaf l having parent p is defined as:

$$\delta(l,p) = \alpha(l) * \frac{\gamma(l,p)}{z} * \beta(l,p) \tag{3.1}$$

Where $\alpha(l)$ is the appearance score of the response reachable by parent p, $\gamma(l,p)$ is the displacement probability measure, and $\beta(l,p)$ represents a penalty for using interpolation between l and p. Displacement probability is calculated as a normal distribution $\mathcal{N}(\mu,\sigma^2)$ and z is equal to the maximum value of the distribution â meaning the most probable amount of displacement will result in the term value of 1. Interpolation nodes in the tree represent a duration in which the available responses fail to explain the situation to some level of certainty. Specifically, interpolation nodes are created when no responses are reachable by a leaf or when all reachable responses have a potential lower than parameter π. Because of this, interpolation nodes have no associated response and can propagate for several time steps until it finds an acceptable response and transition to a response node. Moreover, until an interpolation node transitions into a response node, no explicit location is defined. This is because we require an ending location and the interpolation node's parent to fill the missing location values. Another point to realize is that while interpolation nodes are necessary to cope with occlusions, they represent periods when no image information was available to explain the decisions made. Because of this, we penalize the score during interpolation with the following:

$$\beta(l,p) = \begin{cases} 1 & \text{if } \Delta^t == 1 \\ c_0 * (c_{dr})^{\Delta^t - 1} & \text{if } \Delta^t > 1 \end{cases}$$

Where C_0 represents the initial potential for all new interpolation leaves. c_dr represents the rate at which C_0 drops as the interpolation node propagates. δ^t is the number of time steps between interpolation leaf l and its parent p. The potential of

interpolation leaves is $\beta(l, p)$ until transition, at which point the value $\beta(l, p)$ is used to compute $\delta(.)$.

At this point in the explanation of tree growth, it is helpful to reiterate that many situations can occur in MOT where our information sources, like appearance, are unreliable by themselves. For example, situations can occur where a similar-looking distractor (i.e., some target other than the target of interest) may have a higher heatmap score simply because the target of interest is undergoing a difficult dynamic (e.g., illumination change, occlusion, or large deformation). We rely on relative potentials for several decision-making processes throughout tree growth to deal with this. Let $\mathbf{L}^p$ represent the set of children for node p. We define the relative value of a node l having parent p as:

$$\overrightarrow{\delta}(l) = \frac{\delta(l)}{\delta(l^*)} \tag{3.2}$$

where l^* the child node with the highest potential in $\mathbf{L}^p$. A branch in the tree is defined as sequence of nodes from a leaf to the root. The score of a branch b is:

$$\sum_{l_i \in b} Cs(l_i) * \overrightarrow{\delta}(l_i) \tag{3.3}$$

Where $Cs(l_i)$ is the sum of potentials from l_i onward. The term within the summation of this equation can be thought of as the amount of potential gathered from l_i onward scaled by the relative potential of l_i. The tree growing process continues until the provided stopping criteria Φ have been met. In this chapter, we use a simple stopping criterion — maximum tree height — and discuss more advanced methods in chapter-4.

3.2.3 Branch management

Without branch management, the tree would grow to an unreasonable width (i.e., number of branches) quickly. We use two forms of branch management, merging and pruning. Merging is the primary form of branch management and largely handles

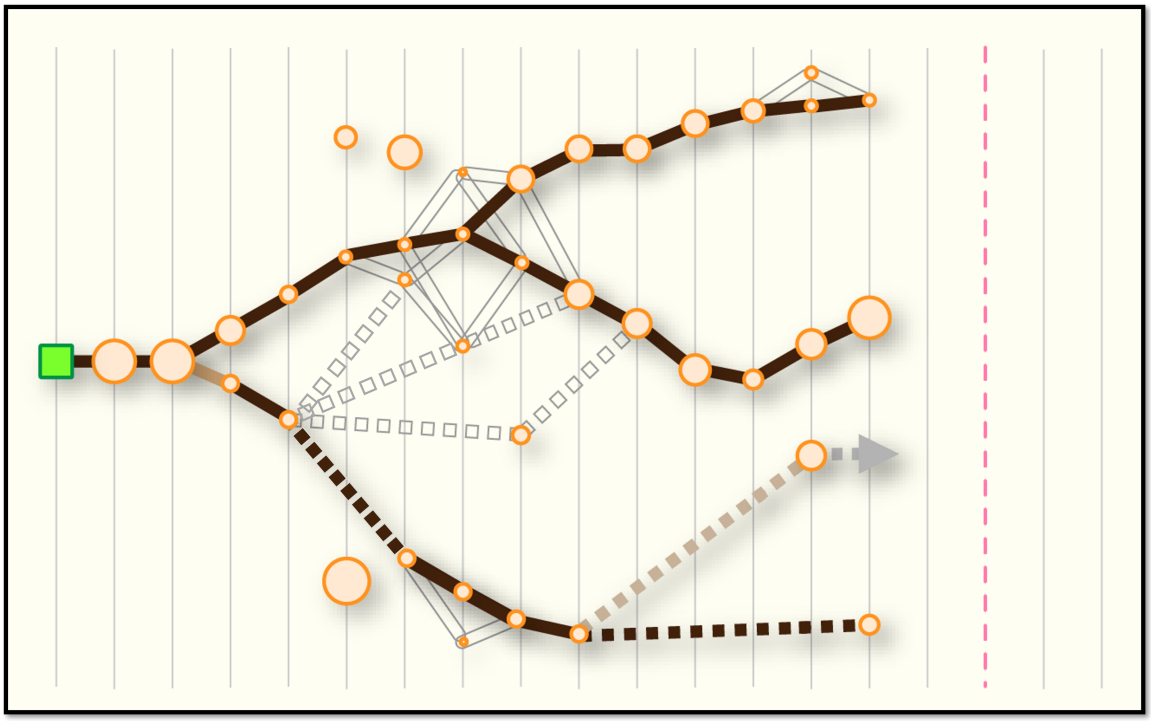

Figure 3.2: Illustration of HypTree construction. Green square is initial location and tree's root. Circles (nodes) represent detected responses in the appearance heatmap (section-3.2.1), where circle size depicts the appearance score. Edges between two circles means a response is reachable and the edge color represents relative score (eq.-3.2, darker is better score). Dashed edges represent interpolation. Gray edges that are hollow (both dashed and solid) represent branch paths that were merged according to section-3.2.3 (no pruning is required for this example). Nodes without edges are distractions captured in larger search radius (section-3.2.4).

redundancies within small temporal windows. When a single response $r \in \mathbf{R}^t$ is associated with more than π_0 leaves, a merge is performed. Other than the leaf with the highest branch score, all leaves are removed. In practice, we set $\pi_0 = 1$, meaning a response will be associated with a most one leaf. Pruning is required when merging cannot keep the number of branches below some predefined maximum. Three parameters control the rate of branch pruning. The maximum number of interpolation nodes at any time step π_1, the maximum number of leaves at any given time step π_2, and the over-pruning rate. When either maximum π_1 or π_2 has been reached, the number of leaves to remove is calculated according to the over-pruning rate and the leaves with the lowest branch scores are removed.

3.2.4 Factoring in Distractions

Up to this point, the score of a branch has been calculated using relative potential, specifically relative in the forward direction. This means that while a transition may be promising from a forward-facing perspective, it may turn out to be a lousy transition from a backward perspective. This point is essential to grasp given that interpolation nodes are permitted (i.e., regions where no information was available, or worse, ignored). Imagine the following situation, for example. A target has undergone heavy occlusion with the scene, forcing an interpolation node to be created. Now imagine that some number of time steps later, a high scoring response suddenly becomes reachable to the interpolation node, but in reality, the target has not reappeared. Here, it would be helpful to know if the high-scoring response agrees that the transition is good relative to its options in the backward direction. If it does not agree, this may indicate a distractor's presence, and that additional explanations should be considered. We model distractors by incorporating relative potentials in the backward direction. Specifically, the search instance area used for heatmap calculation is increased to cover three units of "maximum movement." This allows us to capture responses within one unit of "maximum movement" at time step t that could be reached in the backward direction from responses at time step $t + 1$. We define the backward relative potential of leaf l to response $r \in \mathbf{R}^t$ as:

$$\overleftarrow{\delta}(l) = \frac{\delta(l)}{\delta(r^*)} \tag{3.4}$$

where $\delta(r^*)$ is the highest potential in the backwards direction from r to responses in set Rt-1. We modify the branch score equation to include distractor information:

$$\sum_{l_i \in b} Cs(l_i) * \overrightarrow{\delta}(l_i) * \overleftarrow{\delta}(l_i) \tag{3.5}$$

3.3 Post-processing Trees

Once the tree has met the stopping criteria to discontinue growth, post-processing is performed to determine a more thorough measure of favorability at the tree's node and branch levels. Applying the following during tree growth would be very computationally demanding due to the recursive nature of the functions and is thus why they are performed in a post-processing manner. The following recursive function defines a variant of the branch score that penalizes accumulated scores along routes that were less likely than the alternatives. Let l_n represent the last leaf along a branch, and l_i be the leaf corresponding to frame index i.

$$\hat{\delta}(l_n) = C(l_n) * (\overrightarrow{\delta}(l_n) * \overleftarrow{\delta}(l_n)) \tag{3.6}$$

$$\hat{\delta}(l_i) = (\hat{\delta}(l_{i+1}) + C(l_i)) * (\overrightarrow{\delta}(l_i) * \overleftarrow{\delta}(l_i)) \tag{3.7}$$

In this way, every leaf l_i of a branch has a value calculated that depicts the amount of accumulated potentials beyond leaf l_i, scaled by the relative potential of l_i. A key difference between this function and the functions used during tree growth is that potentials beyond a relatively bad move will be harshly penalized and result in little score accumulated from leaf l_i onward.

3.4 Related works

Traditionally, appearance modeling within SOT has been approached as a learning problem. These works often propose intricate procedures for determining sets of positive and negative training examples followed by model construction performed once or progressively as tracking is performed. For example, [39] proposes a method for determining sets of training examples that can be used with multiple instance learning. Hare et al. [40] propose a structured output SVM that employs a budgeting mechanism while progressively updating the appearance model online. A central

theme in these works is determining suitable training data for learning and adapting appearance models online. Numerous works have been proposed to utilize Siamese network architectures for appearance modeling in visual tracking [35, 34, 41]. These works leverage the feature extraction abilities of deep learning to develop a generalized model of appearance. Bertinetto et al. [38] propose a Siamese network for extracting features in both the template patch and the search instance such that the template feature can be used as a filter on the search instance features. This allows the model to produce a heat map-like response matrix representing appearance similarities at many translated locations within the search image. Several works have built upon this concept by incorporating things such as region proposal sub-networks [36, 42], segmentation [43], higher-order information sources (e.g., appearance history and motion) [44], and offline training procedures that account for distractions [45].

Similar work to the SOT procedure described in this chapter is multiple hypothesis tracking (MHT) [46]. In his paper, Reid [46] proposed to track multiple targets simultaneously in a sequential manner such that track hypotheses are maintained over k frames of the video. The set of possible track hypotheses are then considered jointly to determine global hypotheses, where a global hypothesis is a set of track hypotheses that are not in conflict. The motivating idea is to postpone decisions (i.e., assigning frame observations to targets) until ambiguities are hopefully resolved. Kim et al. [47] later proposed several improvements to work developed by Reid by incorporating modern appearance modeling techniques using deep convolutional neural networks. There are several differences between our work and MHT, but the key distinctions are the following. The first difference involves scoring possible explanations of the data. Our work uses relative scoring for both frame observations (i.e., leaf siblings) and whole trajectories (i.e., branches of the hypothesis tree). The second significant difference is that our work explicitly models the possibility of distractors (section-3.2.4). Lastly, branch management in our work does not rely on the global configuration of

Table 3.1: HypTree construction accuracy over 1000 frames of tracking. A tree was grown for every target in the ant colony testing set (209 total), initialized at the first frame of the video. The distance between branches of the tree and the ground truth are calculated. Branches with distance less than a threshold (width of the target) are recorded at set frame intervals (height column). Below shows the percentage of trees with a branch in the Top-K maintaining the assigned target. At a tree height of 1000, 33% of the trees had correctly maintained the target as the best branch (Top-1) and 58% of the trees maintained the target in at least one of the branches (Top-75).

Height	Top-1	Top-3	Top-5	Top-10	Top-20	Top-30	Top-50	Top-75
5	0.99	0.99	0.99	0.99	0.99	0.99	0.99	1.00
50	0.92	0.93	0.93	0.93	0.93	0.94	0.94	0.96
100	0.86	0.87	0.87	0.87	0.87	0.87	0.88	0.90
200	0.80	0.81	0.82	0.82	0.82	0.82	0.83	0.87
300	0.67	0.69	0.70	0.70	0.71	0.72	0.77	0.79
400	0.56	0.59	0.61	0.61	0.61	0.63	0.67	0.70
500	0.53	0.56	0.58	0.58	0.59	0.61	0.64	0.66
600	0.52	0.56	0.57	0.58	0.58	0.60	0.64	0.66
700	0.51	0.53	0.54	0.55	0.55	0.58	0.64	0.66
800	0.48	0.50	0.51	0.53	0.53	0.56	0.63	0.66
900	0.37	0.40	0.42	0.43	0.47	0.50	0.59	0.62
1000	0.33	0.34	0.37	0.40	0.43	0.49	0.53	0.58

targets in the scene and is concerned explicitly with ambiguities relating to a single target.

3.5 Evaluation

This chapter focuses on evaluating the proposed hypothesis tree method on its ability to track a single target in the presence of multiple distractions. Furthermore, we evaluate its ability to capture multiple hypothetical paths the target may have taken. We use the ant dataset outlined in section-2.4 for evaluation.

We test the ability of hypothesis tree construction to maintain an assigned target by growing trees to a height of 1000 (frames) and checking if the target is still captured within the tree at set frame intervals. Specifically, for each target in each of the videos within the ant colony test set, we initialize a tree. This results in a total of 209 trees grown to a height of 1000 frames. At set frame intervals, the distance between branches of a tree and the ground truth target assigned is calculated. We

recorded if the assigned target is captured by one of the top-k branches in the tree, where k ranges from one to 75 (maximum number of branches). Table-3.1 show the average number of trees containing the assigned target in it's top-k scoring branches (equation-3.5). At height 200, 80% of the trees maintain the target in its best branch, and an additional 7% of the trees maintain the target in at least one branch. At height 1000, only 33% of the trees maintain the target in their best branch, but an additional 25% of the trees maintained the target in at least one branch — resulting in 58% of trees capturing the assigned target at the height of 100 frames.

3.6 Summary

This chapter presented a procedure for growing a tree-like data structure that models possible paths a target may have taken in video recording. We detailed how responses are gathered from a classification heatmap using a Siamese network appearance model, how the tree state is grown in the presence of (or lack of) nearby responses, and how branch management is performed through a combination of two techniques: merging and pruning.

It is worth noting that several promising areas of improvement are happening in the SOT appearance modeling community, such as [48, 49]. This approach should fit well with anything that can provide a heatmap-like output. Furthermore, parts-based appearance modeling methods could be explored [50] to improve performance during heavy occlusions and provide more detailed records of the target's state (e.g., orientation information).

Procedure 1 Tree growth procedure

Input: Hypothesis tree x, the detected responses $\mathbf{R}^t$, minimum child potential θ, and maximum number of leaves λ

Output: Hypothesis tree x grown by one time step

Initialize empty set for new leaves, $\mathbf{L}^t$
Let $\mathbf{L}^{t-1}$ be the set of leaves grown in the previous time step for x
for leaf p **in** $\mathbf{L}^{t-1}$ **do**
    Determine set of responses $\mathbf{R}^* = \{r_i\} \subset \mathbf{R}^t$ reachable by p
    if $\mathbf{R}^*$ is empty **then**
        Create interpolation child leaf within $\mathbf{L}^t$
    else
        for r_i **in** $\mathbf{R}^*$ **do**
            Calculate child leaf l with potential $\delta(l, p)$ ▷ Eq-3.1
            if p represents an interpolation node **then**
                Adjust potential of c ▷ Eq-3.2.2
                Redirect parent of c, to be parent of p
            Add l to leaf set $\mathbf{L}^t$
        if no child of p created has $\delta(l, p) > \theta$ **then**
            Create additional interpolation child leaf within $\mathbf{L}^t$
Update relative potentials for leaves in $\mathbf{L}^t$ ▷ Eq-3.5
Merge out leaves in $\mathbf{L}^t$ covering common response in $\mathbf{R}^t$ ▷ section-3.2.3
if $\|\mathbf{L}^t\| > \lambda$ **then**
    Prune $\mathbf{L}^t$ ▷ section-3.2.3
if any leaves were pruned or merged **then**
    Update relative potentials for remaining leaves in Lt
Update time step of x to be t
Assign $\mathbf{L}^t$ as the current leaf set in x
return x

CHAPTER 4: DETERMINING POSSIBLE TRACKLET CONNECTIONS AND AFFINITIES WITH BI-DIRECTIONAL HYPOTHESIS FORESTS

This chapter proposes a process for determining the set of connections that a tracklet matching approach should consider and a means of scoring the affinity between pairs of tracklets these connections represent. In the tracklet matching context, an affinity measure is a likelihood that two tracklets originate from a common target in the video sequence. Often, the affinity between a pair of tracklets is measured independently, without considering other matching possibilities. As such, the task of tracklet matching is then to take the set of possible connections and their affinities and determine a subset of these connections to make. In other words, the possible connections and their affinities are the elements that describe the matching problem to be solved, and it is tracklet matching's job to solve it.

From this perspective, it should be clear that both the set of possible connections and their affinities are critical to the success of tracklet matching. Having an accurate representation within the set of possible connections together with poor affinity estimates will likely result in poor tracking performance. The same can be said for the reverse, where quality affinity measures are used with a possible connection set lacking representation. The following sections detail how hypothesis trees can be grown bi-directional to construct the association graph and score possible connections in the graph. The underlying motivation for the remaining sections of this chapter is that correct tracking should be independent of the direction it was performed, which is similar to arguments made by [51, 16]. Moreover, an agreement between independent trackers that start at different points and run in opposite directions is meaningful information for performing tracklet matching.

We first describe the process for growing bi-directional hypothesis forests. From there, we outline how a high-precision possible connection set can be determined with the forests. We then discuss two types of affinity measures derived from the bi-directional forests, one of which can be used to directly find a matching solution via cycle detection among forward/backward predictions, without the need for an additional optimization step. The affinity measures are then used within a traditional tracklet matching optimization approach where we compare against a commonly used affinity measure, ReID [30].

4.1 Bi-directional hypothesis forests

The basic idea behind bi-directional hypothesis forests is that independent sequential trackers (i.e., the hypothesis trees), initiated from tracklet endpoints, are run both forward and backward to estimate which associations should be considered and their affinity. We'll describe the process for growing the forest in the forward direction. The only differences between forward and backward forests are the direction the trees are grown and the endpoint of a tracklet from which trees are initiated. Trees in the forward forest are initiated from the tail end (i.e., latest time point) of tracklets, while trees in the backward forest are initiated from the head end (i.e., earliest time point).

Given the initial tracklet set T_0, we first initiate empty sets $\mathbf{F}^*$ and $\mathbf{F}$ representing trees currently growing and completed trees respectfully. For each time step t of the video sequence, tracklets which terminate at time step t have a new tree initiated and added to the set $\mathbf{F}^*$. As we continue to process the frame sequence, we will check if trees that are actively growing within $\mathbf{F}^*$ have met certain "landing" conditions. The "landing" concept here represents part of the stopping condition for tree growth mentioned in chapter-3 and means that branches of a tree are spatially within a certain distance of another tracklet's endpoint. Since we are describing forward forests, the branch locations at a particular time step t are near the head end (i.e., earliest time

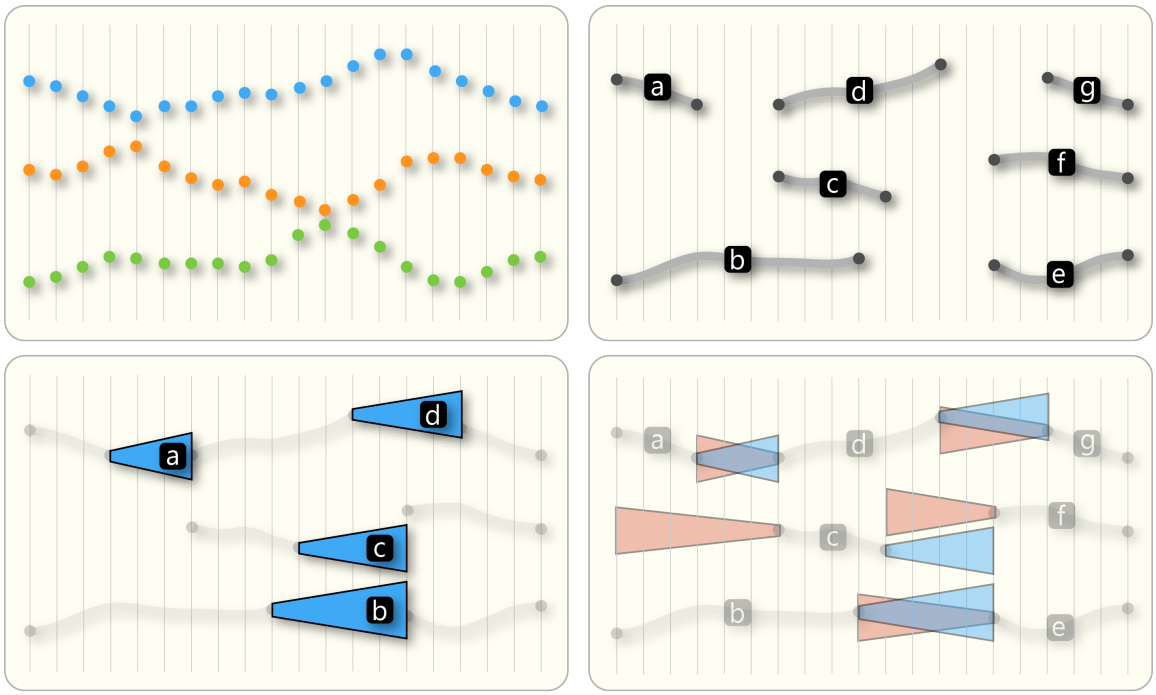

Figure 4.1: Illustration of Bi-Forest applied to a tracklet matching toy example. The four elements shown here are the ground truth trajectories for three targets (upper-left), low-level tracklets that might result (upper-right), a depiction of a forward hypothesis forest grown from the tail endpoints of the low-level tracklets (bottom-left), and a depiction of the Bi-forest after forward and backward forests have been grown. Note that the shape of blue and red trapezoids here only represent tree length (i.e., temporal extent) and do not represent the spatial extent of the tree. Spatial information is not used while determining the set of possible connections.

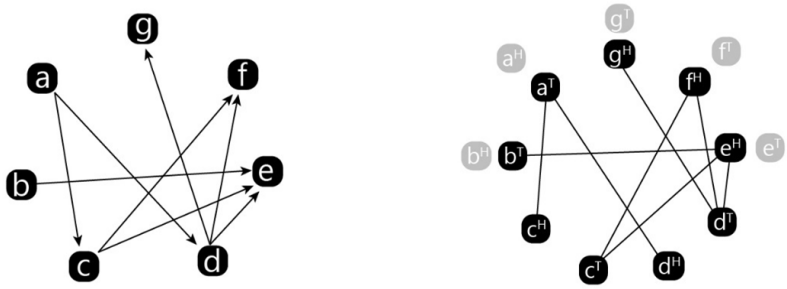

Figure 4.2: Illustration of directed and undirected tracklet matching graph on a toy example. The two graphs above represent what would result from using the Bi-forest illustrated in figure-4.1 to construct the tracklet matching graph. Left shows the directed tracklet matching graph where each tracklet is represented by a single vertex. Right shows the undirected tracklet matching graph where each of the two endpoints for every tracklet is given a vertex. Grey vertices represent tracklet endpoints with no possible connections.

point) location of another tracklet that also occurs at time t. When a tree branch has "landed", we record such and propagate this property to all descendants of that branch. A tree stops being grown when one of two stopping conditions are met. Namely, when a particular amount of branches within the tree have "landed" or when the tree's height (i.e., number of frames processed) reaches some predefined maximum. In practice, we define a weighted percentage of branches needed to have landed, 0.5, where the weight of each branch in the tree equals their branch's score normalized across the branches currently in the tree. We allow trees to grow to a maximum height of 1000. It is worth noting that most of the trees terminate due to the "landing" condition and rarely grow to a height of 1000.

4.2 Tracklet affinity and matching through prediction cycles

This section details how bi-directional hypothesis forests can be used to perform tracklet matching directly, without the need for traditional optimization techniques using the maximum a posteriori (MAP) formulation of tracking. We can view the landing point of the best branch within a hypothesis tree as a prediction. The predictions of these trackers are then represented as directed edges on an association graph $G = (V, E)$. This allows for a binary form of affinity to be measured through graph cycle detection. Given a set of tracklets, we grow both forward and backward hypothesis forests from tracklet endpoints. Each tracklet is represented by two vertices in G corresponding to the head (beginning) and tail (end) of the tracklet — no edge is defined between the two endpoints. Directed edges in G represent predictions from the trackers (i.e., targets to track which "landed"). We first detect prediction agreements by finding graph cycles with a length of 2. We then filter all detected cycles which have a vertex with more than one inbound edge. A vertex with more than one inbound edge would mean that multiple predictions "landed" on that tracklets endpoint and thus would represent an unreliable association. Finally, we associate the tracklet endpoints corresponding to the filtered cycles detected in G. Note that

Procedure 2 Forward hypothesis forest construction

Input: Initial tracklet set $\mathbf{T}_0$, Siamese network backbone B and head H, tracklet landing criteria Φ
Output: Hypothesis Forest $\mathbf{F}$

Let $\mathbf{F}$ be the set of completed trees, initially empty
Let $\mathbf{F}^*$ be the set of trees currently being grown, initially empty
for frame t **do**
    Calculate the frame level features ξ_{frm} with backbone B
    for tree x in forest $\mathbf{F}^*$ **do**
        Calculate response heatmap $h = H(\xi_{frm}, \xi_{\hat{\tau}_i})$ ▷ section-3.2.1
        Determine region responses $\mathbf{R}^t$ within h ▷ section-3.2.1
        Apply tree growth procedure to x using $\mathbf{R}^t$ ▷ Procedure-1
        if x meets the landing criteria Φ **then**
            Remove tree x from set $\mathbf{F}^*$
            Add x to the set of completed trees $\mathbf{F}$
    Let $\hat{\mathbf{T}} = \{\hat{\tau}_i\}$ be tracklet tail endpoints in $\mathbf{T}_0$ occurring at t
    for $\hat{\tau}_i$ in $\hat{\mathbf{T}}$ **do**
        Calculate template features $\xi_{\hat{\tau}_i}$ with appearance model backbone B
        Initialize a new tree having template $\xi_{\hat{\tau}_i}$ and add to set $\mathbf{F}^*$
Post-process trees in $\mathbf{F}$ ▷ section-3.3
return F

while this approach is practical at matching, it has a few key drawbacks that will be discussed in chapter-5.

4.3 Constructing possible connections set

As previously mentioned, it is important to represent the correct connections within the possible connection set — meaning recall in critical. There is little to no hope for tracking to perform well with a low recall on the possible connection set. Additionally, prior works have shown that it can be to the matching algorithm's advantage if higher levels of precision can be obtained without sacrificing recall [26]. Meaning false positives have been filtered within the possible connections set. Here we describe how the Bi-forests can construct a high precision set of possible connections without sacrificing recall. We do this by examining the temporal extent of trees in the Bi-forest to define what connections should be considered for matching.

Let $\overrightarrow{x}_i$ represent the forward tree generated for the tail end of tracklet τ_i, and $\overleftarrow{x}_k$

be the backward tree generated for the head end of some other tracklet τ_k. Further, let the $\overrightarrow{x}_i^S$ and $\overrightarrow{x}_i^E$ the beginning and ending time-steps for tree $\overrightarrow{x}_i$ respectfully. A connection between τ_i and τ_k is possible if the following holds:

$$\overrightarrow{x}_i^S > \overleftarrow{x}_k^E \; and \; \overleftarrow{x}_k^S \leq \overrightarrow{x}_i^S \; and \; \overleftarrow{x}_k^E \leq \overrightarrow{x}_i^E \tag{4.1}$$

Note that this definition does not use any spatial information defined by the trees for determining if a connection is possible. Doing so would risk lower recall performance within the possible connection set. During our evaluation, we will show that the temporal extent of trees filters a large portion of incorrect connections without suffering losses in recall.

4.4 Tracklet affinity as tree similarity

This section details how a traditional form of affinity can be measured given the Bi-Forests and set of possible connections in the association graph. For each possible connection in the association graph, we can compare the trees that stem from the connection endpoints and calculate a few similarity measures. Assume that the association graph contains the possible connection between tracklets τ_i and τ_k. Also, let $\overrightarrow{x}_i$ and $\overleftarrow{x}_k$ represent the forward and backward trees generated for the proper endpoints of tracklet τ_i and τ_k respectfully. We define the following four tree similarity affinity measures:

Best branch landing distance (bbld): For both forests $\overrightarrow{x}_i$ and $\overleftarrow{x}_k$, we identify the best scoring branch. Specifically, we find the best scoring branch at the time-step when the opposite tracklet begins. The L2 distance between the landing location (i.e., the tracklets initial location in that direction) and the best branches location is calculated. The average between these two distances serves as the best branch landing distance affinity.

Most similar branch pair distance (sbd): For every pair of branches between

$\overrightarrow{x}_i$ and $\overleftarrow{x}_k$, the average L2 distance is calculated for the temporally overlapping segment. The distinction of the temporally overlapping segment is necessary since a tree can grow beyond (but never shorter than) the temporal range of the connection being considered. The smallest average distance between all branch pairs serves as the most similar branch pair distance affinity.

Min relative weighted pair distance (brwd): For every branch pair ($b_n \in \overrightarrow{x}_i$, $b_m \in \overleftarrow{x}_k$) of branches between trees $\overrightarrow{x}_i$ and $\overleftarrow{x}_k$, we calculate the average L2 distance over the temporally overlapping segment as well as the relative score of each branch. The relative score is the branch score divided by the maximum scoring branch in its associated tree. Let $rel(b_n)$ and $rel(b_m)$ represent these relative branch scores. Let $avg_D(b_n, b_m)$ represent the average distance over the temporally overlapping segment. The following equation calculates the relative weighted pair distance, and the minimum across all pairs serves as the min relative weighted pair distance:

$$avg_D(b_n, b_m) * (1.001 - \min(rel(b_n), rel(b_m))) \tag{4.2}$$

Normalized weighted pair distance (bnwd): For every branch pair ($b_n \in \overrightarrow{x}_i$, $b_m \in \overleftarrow{x}_k$) of branches between trees $\overrightarrow{x}_i$ and $\overleftarrow{x}_k$, we calculate the average distance over the temporally overlapping segment as well as the normalized score of each branch. Let $nrm(b_n)$ represent the score of branch b_n normalized over all branches within its associated tree.

$$avg_D(b_n, b_m) * (1.001 - \min(nrm(b_n), nrm(b_m))) \tag{4.3}$$

4.5 Related Works

Although uncommon, a few works in the MOT literature use the information within the ”hard” regions of tracking. Kalal et al. [51] propose to perform automatic error detection based on the forward and backward sequential key-point trackers. Rather than determine elements necessary for performing tracklet matching, [51] utilizes disagreement between the trackers to identify errors. Wang et al. [52] propose to extrapolate tracklet endpoints (i.e., extend as much as possible) using online adapted instance detectors and determine appearance affinity measures for use in the global association. Our work also utilizes what can be thought of as online instance detectors. Still, it goes further than extrapolating to the point of uncertainty by capturing possible routes the target may have taken beyond the point of uncertainty.

Milan et al. [53] propose to automatically identify regions of tracking that likely contain an error and consider trajectories from sequential trackers as a replacement to the region in question. Trajectories from sequential trackers are accepted if including the segments minimizes a global energy function. The tracklet matching approach used the ABCTracker software [16] also proposes to use the information within the ”hard” trajectory regions to perform matching. They attempt to globally track all targets in the scene, both forwards and backward, from tracklet endpoints. Predictions by the trackers, represented as cycle detection like mentioned in section-4.2, determine the matching solution over multiple iterations. The foreground corresponding to known target locations is removed during each matching iteration, and the remaining foreground is maximized among the set of targets currently being tracked using a genetic algorithm-based particle filter. Again, the goal in that work was not to capture the possible routes a target had taken. Furthermore, the method proposed in [16] relies heavily on the foreground in order to allow matching iterations to account for information gathered in previous stages of matching.

4.6 Evaluation

This chapter evaluates the proposed bi-directional hypothesis forests on their ability to determine essential elements of tracklet matching association graph — namely, the set of possible connections and their affinities. We first compare the Bi-directional hypotheses forest with the typical means of determining possible connections. The method we compare with, which is often used in tracking literature, is a maximum temporal window [29]. Here all connections within a set temporal distance are considered as possible. Table-4.3 shows the number of connections that result for each method as well as the percentage of tracklet endpoints that have their correct connection captured in the matching graph. The table shows that the Bi-forest approach can capture 98% of the correct connections while only using 18,011 possibilities in the matching graph. Compared to a maximum window approach of 768 frames, Bi-forest can gather nearly as many correct possibilities (98.22% vs. 99.1%) while only considering 2.6% of the connections (18,011 vs. 685,195).

We next evaluate the affinity estimating abilities of the proposed approach. We compare with ReID [30], a widely used feature in the tracking literature [54, 55, 47, 56, 57, 58, 59] that utilized information in the "easy-to-track" regions of a tracklet to estimate affinities. We construct an association graph using the Bi-forest approach outlined previously in this chapter to compare the methods. According to the dataset ground truth, the prevalence of correct connections in the association graph created with bi-forest is 12.07%. Affinities for connections in the graph were calculated using ReID and combinations of bi-forest affinities outlined in 4.4. The association graph, together with the method's estimated affinities, were then optimized using the Hungarian approach [60, 29]. Table-4.1 shows the performance of each method to classify connections. Note that no additional information was provided in the cost matrix to the optimization step. Finally, we evaluate the cycle detection methods for tracklet match classification. Results are reported in Table-4.2.

Table 4.1: Comparison of affinity measures for match classification on one ant video from the ant colony test dataset.

	TP Endpoint Predictions	FP Endpoint Predictions
Reid	1,667	508
HypTrees (bbld)	2,057	118
HypTrees (sbd)	2,085	90
HypTrees (bwrd)	2,065	110
HypTrees (bnrd)	2,084	91
Hyptrees (all)	2,088	87

Table 4.2: Tracklet matching classification comparison using variants of Bi-Forest cycle prediction on the ant colony test set videos. Numbers are the summed across all 5 videos in the set. The two right-most columns show the number of tracklet endpoints with a least one correct match and no correct matches available respectfully.

	Endpoint Connections Predictions			Endpoints w/ Answer	Endpoints w/o Answer
	TP	FP	Total		
Forward Predictions	5939	112	6051	7749	698
Bi-Forest Cycles	6112	65	6177	7749	698
Bi-Forest Cycles (Strict)	5179	41	5220	7749	698

4.7 Summary

This chapter proposed several ways in which bi-directional hypothesis forests can construct essential elements of the tracklet matching problem. First, we showed how bi-forests could determine a high-precision set of connections that a tracklet matching approach should consider. Then we showed how several traditional affinity measures could be calculated between pairs of tracklets using bi-forests. We then detailed how a binary form of tracklet affinity can be established based on cycle detection among independent sequential tracker predictions such that tracklet matching can be performed directly without the need for an optimization step.

Table 4.3: Comparison of approaches for determining possible connections set.

		Max window size								
Approach	Bi-Forest (ours)	8	16	32	64	128	256	512	768	1024
Possible Connections	18011	9363	18131	35131	67370	131030	250423	472980	685195	882197
Tracklet endpoints with answer	4367	1952	2722	3859	4127	4253	4353	4353	4406	4422
Percent with answer	0.982	0.439	0.612	0.760	0.868	0.928	0.956	0.979	0.991	0.994

CHAPTER 5: ESTIMATING TRACKLET MATCHING MARGINALS

This chapter proposes a probabilistic graphical model for estimating tracklet matching marginals. We first outline how we represent the tracklet matching problem as a factor graph such that matching is posed as a task of determining the best action to take on a tracklet-endpoint. Possibilities considered in the tracklet matching factor graph, as well as their favorability, are determined using the Bi-Forest approach (discussed in chapter-4). We then describe how marginal probability estimates over the set of possible connections are determined using the sum-product loopy belief propagation algorithm. The matching marginals are then used to construct a greedy matching solution together with hypothesis tree aided gap-filling. Finally, we evaluate the proposed approach against several prior works using the ant dataset detailed in section-2.4.

5.1 Constructing the tracklet matching factor graph

Let $\hat{g}$ represent the tracklet matching factor graph being constructed and g be the undirected tracklet matching graph detailed in section-4.3. To model possible connections in $\hat{g}$, we first extract the set of possible connection pairs, $\mathbf{\Psi}$, from g. For each connection pair in $\mathbf{\Psi}$, an unknown binary variable node is added to $\hat{g}$. The set of preprocessed affinity measures corresponding to a connection pair in $\mathbf{\Psi}$ define the notion of observed evidence or favorability within $\hat{g}$. In situations where only a single affinity measure is defined for each connection, a unary factor is created and linked to the connection variable in $\hat{g}$. Given the preprocessed affinity measurement m, the unary factors potentials are defined as $[m, 1-m]$ — representing the favorability of the connection being correct and incorrect respectfully.

In situations where a set of affinity measurements are available for each connection, an equality constraint factor is used to join the evidence. For each measurement $m_i \in \mathbf{m}$ a variable is created with a unary factor having potentials: $[m_i, 1 - m_i]$. Each of these measurement variables, along with the connection variable associated with the measurement set $\mathbf{m}$, is connected to an equality constraint factor. The number of matrix dimensions for the equality constraint factor's potentials will be one greater than the number measurements in $\mathbf{m}$. The value of the potentials is zero except for the configurations where all connected variables share the same label (i.e., two configurations, all true or all false). The value at these configurations is uniform, equaling 0.5. An example of this is shown for connection variable $(b, e)'$ within figure figure-5.1.

A constraint often utilized in tracklet matching is that a tracklet can connect with at most one other tracklet. Theoretically, this constraint could be modeled within the tracklet matching factor graph in one of two ways. One approach is to construct many pairwise dependencies between connections to encode the feasible associations. In this way, each pairwise dependency factor would constrain and communicate with exactly two connection variables. This results in a vast number of small loops in the factor graph. Furthermore, the number of dependency factors is defined by the number of connections being considered when using this approach. This means the number of unknown variables and the number of dependencies grow as more possibilities are considered.

During our attempts at using this approach, we experienced oscillating predictions between two significantly different solution states. Momentum was required to obtain any convergence, significantly more than what was reported in related works on oscillations in very loopy graphs [61]. Additionally, [61] reports that when momentum is applied to settle oscillating state predictions, the resulting estimates can be unreliable.

The other approach to modeling the "matches to at most one" constraint is to capture all of the dependencies related to a tracklet endpoint using a single factor. Here, the number of variables connected to the constraint factor grows as we consider more possibilities, not the number of constraint factors. One challenge with this approach is that the number of configurations modeled in the dependency factor's potentials matrix grows exponentially. Specifically, the matrix will be of size 2^n for a binary random variable formulation, where n is the number of possible connections considered for a particular tracklet endpoint. We will first describe the details of constructing the factor and then explain how this configuration size issue can be overcome by taking advantage of the constraint the factor governs.

Let $\mathbf{\Psi}^H$ be the set of unique connection heads (i.e., the earlier end of the two connection endpoints) within the set of possible connections $\mathbf{\Psi}$. For each connection head id ${c_i}^H \in \mathbf{\Psi}^H$, we first determine the set of variables ${\mathbf{v}_c}^H$ in $\hat{g}$ that have ${c_i}^H$ as their connection head id. Given ${\mathbf{v}_c}^H$, we construct a factor that dictates at most one variable in ${\mathbf{v}_c}^H$ can be correct. The values for the factor's potentials are all zero, except for configurations where only one of the variables in ${\mathbf{v}_c}^H$ is true or when all variables are false. Note that the configuration where all variables are false is included here for clarity purposes and will be replaced by a better representation in the following paragraphs. The value at these non-zero configurations is uniform. All variables ${\mathbf{v}_c}^H$ are then connected to the constraint factor. This process is repeated for unique connection tails $\mathbf{\Psi}^T$ (i.e., the latter end of the two connection endpoints).

Now, consider that a majority of the factor's configurations are defined as impossible (i.e., potentials are zero). In this way, the factor is only concerned with $n + 1$ of the 2^n possible configurations. Leveraging this is critical for achieving reasonable memory and computation performance, especially for endpoints having many connection possibilities. We modify how messages from these factors are created, ignoring all configurations other than the $n + 1$ that are possible.

Procedure 3 Tracklet matching factor graph construction

Input: Tracklet matching graph g, set of connection affinity scores $\mathbf{S}$
Output: Tracklet matching factor graph $\hat{g}$

1: Let $\mathbf{\Psi}$ be the set possible connection pairs, extracted from g
2: Initialize empty factor graph $\hat{g}$
3: **for** connection id pair c in the set of possible connections $\mathbf{\Psi}$ **do**
4: Create connection variable for v_c in $\hat{g}$
5: Let $\mathbf{S}_c \subset \mathbf{S}$ be the set of affinity measures associated with c
6: Create evidence factor using $\mathbf{S}_c$
7: Connect evidence factor with v_c
8: Let $\mathbf{\Psi}^H \subset \mathbf{\Psi}$ be the set of unique connection head ids
9: **for** connection head id c_H in $\mathbf{\Psi}^H$ **do**
10: Let $\mathbf{v}_c{}^H \subset \hat{g}$ be variables having c^H as their connection head
11: Create Na variable v_{Na}^H and its unary factor for c^H
12: Create a constrained factor v_f for c^H
13: Connect all variables in $\mathbf{v}_c{}^H$ with v_f
14: Connect Na variable v_{Na}^H with v_f
15: **repeat** lines 8—14 for set of unique connection tail ids $\mathbf{\Psi}^T \subset \mathbf{\Psi}$
16: **return** $\hat{g}$

Up to this point, the factor graph has been made aware of possible connections, a local measure of how favorable each connection is, and can convey the "matches to at most one" constraint to variables with constraint factors on tracklet endpoints. The last type of elements added to the factor graph represents the possibility that a tracklet endpoint has no correct connection modeled in the graph. We create what we will refer to as a Na (i.e., a No Answer available) variable for each tracklet endpoint. The Na variable is connected to two factors, a unary factor representing how likely the Na state is and the "matches to at most one" constraint factor associated with the tracklet endpoint in question. By connecting the Na variable to the constraint factor, we essentially replace the configuration where all connection variables are false, allowing us to utilize any evidence that all possible connections modeled are incorrect.

5.2 Sum-product loopy belief message passing

Given the loopy tracklet matching factor graph $\hat{g}$, we apply the sum-product message passing algorithm [62] to estimate matching marginals. We use the sum-product

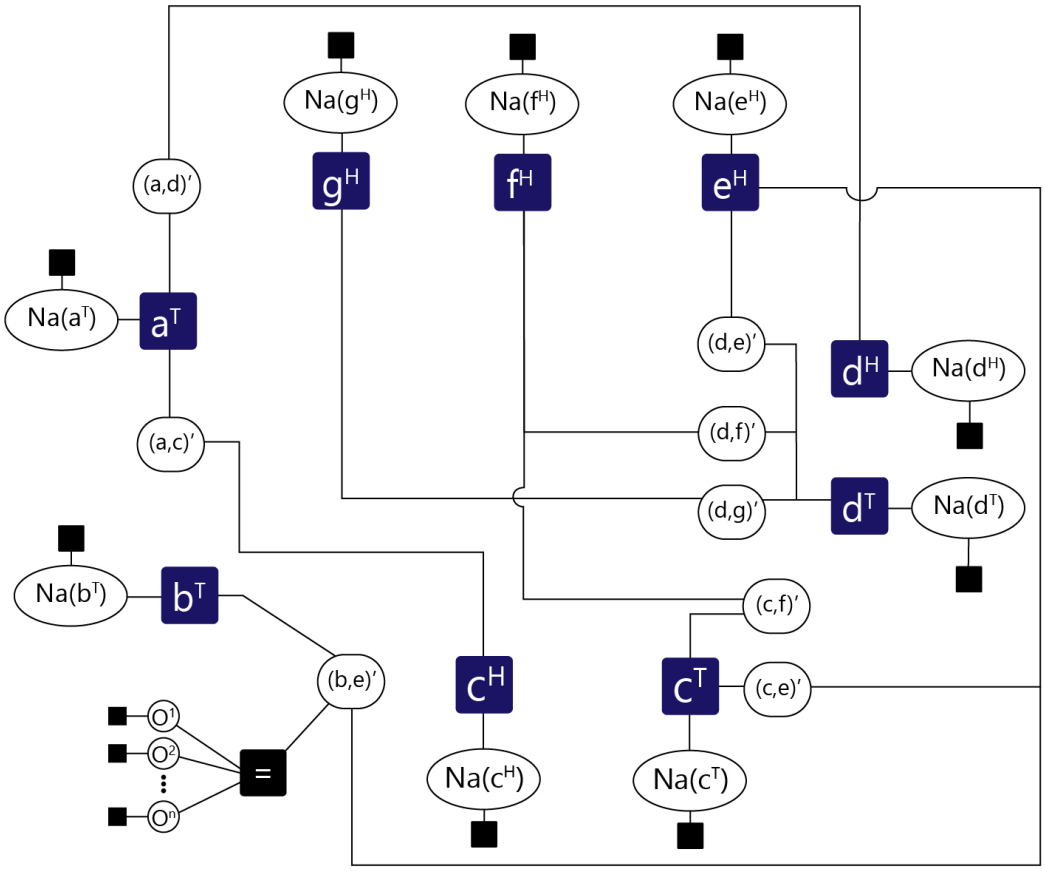

Figure 5.1: Tracklet matching factor graph example. Here illustrates the tracklet matching factor graph that would result from the toy example shown in figure-4.1 and figure-4.2. Solid black squares depict unary factors. A circle with the text "Na(x^H)" represents a Na variable on the head endpoint of tracklet x. Na variables capture the possibility that a tracklet endpoint has no correct answer represented in the association graph. Circles with the text "(x, y)" represent a connection variable between tracklets x and y. For illustration purposes, the equality constraint factor (black square with "=") is only shown for the lower-left association variable (b,e)'. A dark blue square with the text "x^T" represents a matching constraint factor node for the tail endpoint of tracklet x. The matching constraint factor nodes dictate that one and only one action (i.e., connection or no connection) explains their associated endpoint, where the possible actions are variables connected by an edge in the graph — namely, a Na variable and the possible connections.

approach instead of the max-product since we wish to estimate matching marginals. Despite the graph containing loops, we find convergence is obtained in a moderately small number of message passing cycles — often less than 30 iterations for our datasets. Convergence within loopy graphs is typically determined by monitoring the average change across all variables in the graph. When the average amount of change reaches a predefined threshold, the graph is said to have converged to an approximate state, and message passing stops. Due to the size of our graph, we also look for the maximum change in any single variable. This is because most of the variables in our graph tend to settle faster than a smaller portion of the graph. To handle this, we also monitor the maximum amount of change in the graph. When the average amount of change, as well as the maximum amount, drops below ζ_1 and ζ_2 respectfully, we terminate message passing and extract the estimated marginals for all connection variables in $\hat{g}$.

5.3 Building the matching solution

We use a greedy approach to building the matching solution given the association marginals. We maintain a list of head and tail tracklet endpoints that have been assigned a connection or marked as having no answer (i.e., the Na variable representing no connection). We iterate the following until all tracklet endpoints have a label assigned: 1.) select the variable with the highest marginal probability (among all connection and Na variables), 2) if assignment conflicts with available endpoints lists, continue. Otherwise, apply the assignment and record the endpoint(s) as no longer available for further assignment.

Gap frames between matches determined during greedy solution building are filled using the hypothesis trees corresponding to the connection endpoints. Given the pair of trees corresponding to the connection endpoints, the pair of branches with the smallest L2 distance is selected. A weighted average between the pair is determined such that the weight of a branch's location decreases linearly as you continue up the

tree (i.e., away from the root node).

5.4 Related Works

Several works have been proposed that use factor graph representations of the data association problem for sequential MOT [63, 64, 65, 66, 67, 68, 69]. In these works, the task is to sequentially associate targets with measurements (i.e., detections) frame-by-frame. Some of these works consider multiple frames at a time while processing the video sequentially [68, 70]. The target-to-detection graph constructed and solved frame-by-frame will almost always be smaller than a tracklet-to-tracklet graph would be that is constructed across the entirety of the same video.

The typical approach to tracklet matching-based MOT is to construct a graphical representation of the matching possibilities and estimate the optimal solution. To our knowledge, no works concentrate on estimating the marginal probability of the tracklet matching possibilities. Numerous works frame the tracklet matching problem using either linear programming [71, 72], network-flow [73], graph cliques [74, 75], or Hungarian optimization [29, 76, 77, 78, 28]. Some works propose statistical graphical model representations of the tracklet matching problem (e.g., mean-field, conditional random field) but only seek to find the optimal solution rather than the relative favorability of different solutions [79, 80]. Several works pose tracklet matching as an energy minimization problem [81, 80, 53, 82]. These works operate by iteratively modifying the global solution state and seeing if the resulting state has lower energy. For all of the examples listed in this paragraph, the goal of the method is to determine the optimal solution. The work presented in this dissertation first seeks to estimate the certainty of possible explanations and then pick a solution consistent with this measure of certainty.

Based on our literature search, we believe that estimating matching marginals is overlooked in the related works for a few reasons. First, the tracklet matching problem usually results in large graphical models that are difficult to solve efficiently. By

using Bi-forest (chapter-4) to determine the set of possible connections, we were able to reduce the size of the matching graph significantly without sacrificing recall. The second reason concerns tracking performance evaluation: there is no benefit to accurately modeling uncertainty when using standard evaluation metrics. An uncertain error made by a tracklet matching algorithm is treated no differently than a certain one.

5.5 Evaluation

This chapter evaluates the proposed tracklet matching method on its ability to both match tracklet and fill connection gaps (i.e., the detections between connection endpoints). For this, we use the evaluation process typically used in MOT literature, which is to sequentially map ground truth identities to predicted trajectory identities and calculate several metrics based on this mapping. One evaluation conducted in this chapter is across methods that share a common set of input (i.e., detections). Another part of the evaluation compares our work with recent tracking pipelines submitted to the 2020 MOTChallange benchmark. First, we detail the metrics used during the evaluation before discussing implementation details and results.

5.5.1 Evaluating metrics

We use an evaluation procedure and metric set commonly used MOT. The evaluation procedure and metrics follow from [29, 83]. Namely, we use the following metric set below. Note that the definition we use for fragment (Frag) and ID switch (IDS) is that of [29].

- Fragment (Frag) — Ground truth track changes its matched predicted track id.
- ID Switch (IDS) — Predicted track changes its matched ground truth id.
- Recall (Rcll) — Percent of ground truth matched to a predicted detection.
- Precision (Prcn) — Percent of predicted detections matched to a ground truth.

- Mostly tracked (MT) — Ground truth trajectory is detected more than 80%
- Partially tracked (PT) — Ground truth trajectory is detected between 20%-80%
- Mostly lost (ML) — Ground truth trajectory is detected less than 20%
- ID Precision (IDP) — Percent of predicted detections correctly identified
- ID Recall (IDR) — Percent of ground truth detections correctly identified

5.5.2 Implementation details

The set of methods sharing a common input include: GAPF [16], Cycle-BiForest (section-4.2), and the proposed method (LBP-Forest). For all of these methods, input is the set of low-level tracklets outlined in section-2.3. We also include a modified version of GAPF where the foreground is disabled during the tracklet matching stage. Specifically, all pixels of the foreground image are set to zero. The evaluation includes two versions of prediction cycle detection-based matching, strict and loose. The strict version is as described in section-4.2. The loose version is similar to the strict version, except that connections involving nodes with more than one inbound edge are not filtered. Concerning the Bi-Forest used by the proposed method, LBP-Forest, we used the same training process and training data as described in section-4.6. Namely, the appearance model was trained using the ant colony dataset training split (section-2.4, table-2.1).

Two recent methods within the MOTChallange 2020 benchmark are used while comparing tracking pipelines with the proposed approach: WOBW and MPN. The first of these works is called tracking without bells and whistles (WOBW) [54]. WOBW proposed to track targets by using bounding box regression from a trained object detector to predict subsequent positions. They also extend the method by using ReID [30]. The other method we compare against is MPN [55]. Authors of MPN [55] propose to learn message functions (i.e., multi-layered perceptron) such

that message passing can be performed towards solving the matching solution. We follow the procedures outlined by the authors during training and evaluation. According to MPN [55], we use the output trajectories from WOBW as input into MPN during evaluation. As such, we train three separate deep learning models used by WOBW and MPN, namely, a Fast recurrent neural network (FRCNN), a Siamese Re-identification network (ReId), and the learned message function used by MPN. All three models were trained using the authors' code on the ant colony training dataset (table-2.1) with only slight modification to work on our dataset.

5.6 Discussion of results

Table-5.1 shows the tracking performance comparison between methods that use a common input. The top row of Table-5.1 (low-level tracklets) shows the tracking metrics for the input. The low-level tracklets cover the ground-truth detections with 50% recall and 99% precision, which are connected into short tracklets that result in 6,826 fragment errors and one ID switch. The strict variant Cycle-BiForest performs similar to the loose version. The loose version recovered slightly more of the missing ground-truth detections (4.5 percent points) at the cost of three additional ID switches. Both variants of GAPF achieve the highest recall, with LBP-Forest recovering roughly 3 percent points less of the ground truth. While the number of fragments produced by LBP-Forest is the lowest among all the methods, it significantly increases ID switches. Note that no threshold on matching marginals was used for LBP-Forest during this evaluation. Finally, it is essential to realize that none of the approaches in table-5.1 can produce matching marginals along with the matching solution, other than LBP-Forest.

Table-5.2 shows the tracking performance comparison between methods using different inputs. The tracks produced by WOBW [54] cover 90% of the ground truth with a precision of roughly 85%. To reiterate, the output trajectories produced by WOBW are the input to MPN [55]. The table shows that MPN can increase recall and

precision on its input by 1.3 and 4.3 percent points, respectfully. Furthermore, MPN reduced the number of fragments and ID switches from its input by 824 (32%) and 132 (91%), respectfully. LBP-Forest, which used the low-level tracklets from section-2.3 as input, was able to recover 26.5 percent points more of the ground truth over its input. Neither WOBW nor MPN can estimate tracklet matching marginals because WOBW uses a sequential processing formulation, and MPN operates on detections and uses a learned message function. LBP-Forest does have the lowest number of fragments (95% reduction over input) and the highest precision but performs worse than the other methods in two ways. First, all numbers related to recall are worse. This is due to the differences in the inputs to these methods — LBP-Forest may perform more favorably if the recall was similar across the two inputs. Secondly, LBP-Forest resulted in significantly more IDS switches. The latter of these two points is discussed further in the following comparison.

Even though LBP-Forest showed comparable performance in the previous two comparisons, a significant downside of the method was the number of ID switches. Note that this is partly due to the threshold on matching marginals being set to zero for comparison. Table-5.3 shows the performance of LBP-Forest using different threshold values. At a threshold of 0.80, the number of fragment and ID switch errors is more comparable to the numbers reported for MPN in table-5.2 — albeit with lower recall, which we mentioned is likely due to differences in the input recall. Additionally, Table-5.3 also indicates that LBP-Forest is modeling the uncertainty of different tracklet matches. No ID switches occur at a threshold of 0.99. Even at the highest threshold tested in table-5.3, 0.999, more the 3,400 correct matches are made between the input tracklets.

5.7 Summary

This chapter proposed a probabilistic factor graph representation of the tracklet matching problem. We detailed how to construct a factor graph that considers the set

of possible connections and their connection affinities, both of which are determined with the Bi-Forest approach. We showed how matching marginals could be calculated with the sum-product algorithm, and with these marginals, how a greedy approach can be used to construct a tracklet matching solution. Furthermore, we show that the hypothesis trees can perform gap-filling once a matching configuration is determined. Finally, we evaluated the proposed approach against several prior works using the ant colony dataset detailed in section-2.4. The evaluation showed that the proposed method performs favorably against the previous works and can model the certainty of individual matches.

Future works could consider alternative means of calculating affinities. For example, one could use a learning method to combine individual affinity measures, similar to [29, 26]. Additionally, it is worth noting that even though Na terms were able to be used in a general manner here, they could be expanded upon to better model possibilities. For example, instead of aggregating several classes of possibilities together, as is done here, one could explicitly model terms that represent: the target exiting the scene, false-positive tracks, and even more abstract concepts like tracks that divide and merge (e.g., cell tracking were cells can split and merge, or pedestrians entering a car). Thus, making the Na concept more interpretable and likely more robust.

Furthermore, several works have been proposed on more efficiently performing message passing in loopy graphs [84, 85]. Although this work did not modify message passing scheduling, we parallelized message creation and marginal updates to help reduce computation time. It will be worth considering techniques such as residual belief propagation [84] as tracking situations become more complex.

Finally, this approach could be modified to operate on detections instead oflow-level tracklets. Doing so would further motivate the need to investigate efficiently performing message passing. It would also require a motion prior since these are determined from the low-level tracklets.

Table 5.1: Tracklet matching comparison using common set of detections as input — specifically, low-level tracklets. Cycle-BiForest is the method described in section-4.2. No marginals threshold was used when gathering LBP-Forest results (see table-5.3).

	Frag ↓	IDS ↓	FP	FN	MT ↑	PT	ML ↓	IDP	IDR	Rcll	Prcn
Low-level Tracklets (Input)	6826	1	3215	470629	37	129	43	35.8%	18.4%	50.0%	99.4%
GAPF [16]	313	5	3894	196233	138	53	18	80.2%	64.0%	79.2%	99.5%
GAPF [16] (No Foreground)	578	1	3554	255856	112	76	21	71.5%	52.5%	78.1%	99.5%
Cycle-BiForest	1848	8	5394	313033	95	87	27	57.3%	39.3%	66.7%	99.3%
Cycle-BiForest (Strict)	2879	5	4758	356238	77	102	302	46.4%	29.7%	62.2%	99.3%
LBP-Forest (Ours)	302	227	31,679	220,799	126	61	22	71.0%	57.6%	76.5%	95.6%

Table 5.2: Tracklet matching comparison using different pipelines (i.e, detection inputs are not the same). WOBW [54] uses a deep learning detection method. Tracks output by WOBW serve as input to MPN [55]. LBP-Forest does not use deep learning based detections. Input to LBP-Forest is the low-level tracklets (table-5.1). No marginals threshold was used when gathering LBP-Forest results (see table-5.3).

	Frag ↓	IDS ↓	FP	FN	MT ↑	PT	ML ↓	IDP	IDR	Rcll	Prcn
WOBW [54]	2,561	145	39,394	91,671	167	26	16	61.7%	59.2%	90.2%	85.8%
MPN [55] (Input: WOBW)	1,737	13	10,959	80,244	168	27	14	66.9%	62.9%	91.5%	90.1%
LBP-Forest (Ours) (Input: Low-level Tracklets)	302	227	31,679	220,799	126	61	22	71.0%	57.6%	76.5%	95.6%

Table 5.3: Tracking performance over different thresholds on matching marginals. The number of matches column shows the number of matches that were made by LBP-Forest at said threshold.

Thresh	Number of Matches ↑	Frag ↓	IDS ↓	FP ↓	FN ↓	MT ↑	PT	ML ↓	IDF1 ↑	IDP ↑	IDR ↑	Rcll ↑	Prcn ↑
0.00	7,869	302	227	31,679	220,799	126	61	22	63.3%	71.0%	57.6%	76.5%	95.6%
0.05	7,865	322	223	30,987	221,011	126	61	22	63.4%	71.1%	57.6%	76.5%	95.7%
0.10	7,844	344	215	29,640	223,635	125	62	22	62.9%	70.7%	57.2%	76.2%	95.9%
0.15	7,801	381	192	26,897	229,030	121	65	23	62.2%	70.3%	56.4%	75.7%	96.2%
0.20	7,733	448	167	24,058	237,824	118	68	23	62.0%	70.9%	55.9%	74.7%	96.7%
0.25	7,659	517	145	22,297	244,462	113	73	23	61.1%	70.3%	54.9%	74.0%	96.9%
0.30	7,585	586	122	19,639	250,981	112	74	23	60.7%	70.4%	54.3%	73.3%	97.4%
0.35	7,499	668	100	18,334	256,514	110	76	23	59.9%	69.8%	53.3%	72.8%	97.5%
0.40	7,417	746	84	17,515	262,704	105	81	23	59.2%	69.4%	52.5%	72.1%	97.6%
0.45	7,331	829	71	16,497	270,775	101	85	23	57.5%	67.8%	50.8%	71.2%	97.7%
0.50	7,236	917	59	14,600	279,043	98	88	23	56.7%	67.4%	49.8%	70.4%	98.0%
0.55	7,152	990	43	12,847	285,994	97	87	25	55.6%	66.6%	48.6%	69.6%	98.3%
0.60	7,061	1,066	38	10,174	293,440	95	88	26	54.3%	65.6%	47.1%	68.8%	98.6%
0.65	6,968	1,162	35	8,737	301,558	94	89	26	52.3%	63.8%	45.1%	68.0%	98.8%
0.70	6,872	1,255	25	7,791	309,161	90	93	26	51.3%	63.2%	44.0%	67.2%	98.9%
0.75	6,762	1,360	16	7,048	315,037	90	93	26	50.3%	62.4%	43.0%	66.5%	99.0%
0.80	6,645	1,472	11	5,841	322,135	89	91	29	49.8%	62.2%	42.3%	65.8%	99.2%
0.85	6,476	1,640	10	5,750	329,855	87	93	29	48.3%	60.6%	40.9%	65.0%	99.2%
0.90	6,294	1,819	8	5,123	337,726	82	98	29	46.2%	58.5%	38.9%	64.1%	99.2%
0.95	5,963	2,149	4	4,695	351,755	74	106	29	42.4%	54.5%	35.3%	62.6%	99.3%
0.99	4,988	3,118	0	4,601	379,769	67	111	31	35.3%	46.8%	28.9%	59.7%	99.3%
0.999	3,451	4,654	0	4,591	407,484	56	120	33	28.8%	39.4%	23.1%	56.7%	99.3%

CHAPTER 6: CORRECTING TRACKLET MATCHING ERRORS WITH ACTIVE SAMPLING OF THE ASSOCIATION GRAPH

This chapter details an interactive procedure that allows a user to correct tracklet matching errors more efficiently. The biggest motivator for such a procedure is that tracklet matching results typically have errors in them for all but the simplest of videos. This can present an issue to individuals who seek to use the results as part of some analysis pipeline. Faced with this issue, one can find a way to either: filter the errors if filtering would not introduce a harmful bias, correct the errors, or find an approach that does not rely so much on tracklet matching performance (e.g., manually annotating the positions all targets in the video from scratch). We will show later in this chapter that manually annotating targets from scratch is a very time-consuming process. Furthermore, we argue that time could be saved by avoiding this process if errors in tracklet matching results could be more easily addressed.

The first two of these ideas for resolving the tracking errors, filtering or correcting the errors, requires some means of identifying where errors likely are in the first place. In the case of correcting the errors, which may be required to avoid biases introduced by filtering, a process is needed to address the identified errors. These points motivate a couple of concepts. The first is that the ability to identify errors would be valuable. Secondly, the process for addressing errors is also necessary.

The following sections define an interactive procedure that allows a user to more efficiently correct tracklet matching errors by leveraging both matching marginals (Chapter-5) and information within Bi-Forests (Chapter-3 and Chapter-4). Details of the method are largely broken up into two parts: Identifying likely errors (i.e., graph sampling) and the correction procedure (i.e., review/user annotation process

that allows us to label nodes in the graph).

6.1 Sampling the association graph

Let $\mathbf{A}^\tau = \{a_i\}$ be the set of actions that could be performed on some endpoint of tracklet τ. Note that we are using a simplified notation here, which does not indicate which of the two tracklet endpoints are being examined. In the context of this chapter, $\mathbf{A}^\tau$ would be the set of possible connections that could be made with the tracklet's endpoint (specifically, the ones captured by the association graph) and the possibility that no correct connection is represented in the graph (i.e., Na term within the factor graph). So, for the endpoint of some tracklet τ that has four possible connections in the association graph, there would be five elements in the set $\mathbf{A}^\tau$.

If the goal of sampling is to identify errors efficiently with respect to the number of samples, then it would be quite naïve to sample tracklet endpoints randomly (i.e., completely disregard any indicators during sampling). For example, consider the following tracking situation: a 2.5-minute video at 30 frames per sec (about 5000 frames) containing roughly 50 objects in any given frame. A situation such as this can easily result in 1000 low-level tracklets, and, in our experience, it is often more than this if using techniques such as 2.3. Reviewing even a third of these endpoints would be unreasonable.

Another approach would be to consider the connectivity of nodes in the association graph. This would essentially result in the sampler favoring endpoints with a larger number of elements in $\mathbf{A}^\tau$. One may be led to believe this would be a good sampling heuristic given that, once the endpoint label has been determined and applied, this represents the number of neighbors that would be affected by the label (i.e., only one action in $\mathbf{A}^\tau$ can be true, all others are thus false). The shortcoming of this approach is that one tracklet endpoint could be very uncertain with only a few options, while some other tracklet endpoint could have many more options available with one clearly the answer.

We could look for endpoints that have the greatest amount of uncertainty on any single action available, essentially assigning the sampling value for an endpoint as $max_{a_i \in A^\tau}(\|p(a_i = 1) - p(a_i = 0)\|)$, where $p(a_i = 1)$ is the predicted probability of action a_i being true. A better approach would be to consider the total uncertainty at a tracklet endpoint. Here, we can calculate this as the entropy over the marginal probabilities of action set $\mathbf{A}^\tau$.

6.2 Correction procedure

We refer to a review in this chapter as a request for user input on a low-level tracklet's endpoint. The information requested from the user is target location annotations at key time steps beyond the selected endpoint. A review contains the following information once completed by the user: the tracklet endpoint, the key-frames selected for user input, and the target's location in those key-frames. Additionally, a review also contains properties associated with the occurrence of certain events (e.g., target exiting the scene) specified by the user. An illustration of this is shown in figure-6.2.

The goal of having the user review the tracklet's endpoint is to determine the best action to take on the selected endpoint. The action could be to connect the tracklet to another tracklet, remove the tracklet (i.e., false-positive tracklet), or extend and terminate the tracklet when no correct connection is available (e.g., exists the scene or reaches the end of the video sequence). The appropriate action is determined by comparing the user's key-frame annotations against known targets locations with which the selected endpoint can connect.

Before we discuss the procedure for creating tracklet endpoint reviews any further, it is important to note that we assume video playback during the user annotation process can only play forward, not in reverse. As such, we only create reviews for tracklet tail endpoints (later end of the tracklet) and do not create reviews for tracklet head endpoints (earlier end of the tracklet). This is because most video players provided with user interface APIs cannot perform playback in reverse. Future works

could consider both tracklet ends and overcome the playback limitation with additional bookkeeping in the code and maintaining two video copies, one forward and one in reverse.

6.2.1 Review Creation

Let $\hat{g}$ and $\mathbf{T}_0$ represent the tracklet matching factor graph and low-level tracklets as detailed in chapter-5 and chapter-2 respectfully. Let $\{\tau_i\}^* \in \mathbf{T}_0$ be the set of all tracklets in $\mathbf{T}_0$ which do not cover the final frame of the video. We create a review for each of these tracklets' tail endpoint. Tracklets that reach the end of the video do not need reviewing since the only possible action that could be taken (i.e., on its tail endpoint) is to remove the tracklet entirely. We create a review for the tail endpoint of every tracklet in $\mathbf{T}_0$. Let $\mathbf{R}_i = \{r_k\}$ be the set of tracklet endpoint reviews created during correction iteration i. Here, a correction iteration refers to a single cycle of sampling and applying a batch of reviews. A new set of reviews is created after each correction iteration. From the set of all possible reviews, the sampler then prioritizes and selects a batch $\mathbf{B}_i = \{r_k\} \subset \mathbf{R}_i$ of size b to present to the user during correction iteration i. Methods for prioritizing and selecting a batch of reviews are discussed in section-6.1.

6.2.2 Review annotation process

We now detail the process of annotating a single review $r_k \in \mathbf{B_i}$ within a given batch of reviews $\mathbf{B_i}$. Since reviews are created on low-level tracklets as opposed to tracklet matching tracklets, there should rarely be a conflict while applying a batch of review answers. A method that creates reviews from and continuously modifies a tracklet matching solution will need to account that multiple matches were likely made within a tracklet; thus, multiple reviews are possible on a single tracklet. Submitting multiple reviews for a single tracklet within a batch could cause conflicts, resulting in wasted reviews, and building a new solution from low-level tracklets after each

correction iteration largely avoids these issues.

Let t_0 be the frame at which the tracklet tail endpoint being reviewed occurs. For now, assume that the set of key-frames selected for annotation have been determined in advance. Section-6.2.3 discusses different approaches to scheduling key-frame selection as well as schemes that select key-frames dynamically as user annotations are gathered.

First, the user is presented with a video frame focused on the target to follow. Specifically, some frame prior to t_0 is selected as the start of video playback — roughly 30 frames before t_0 if the length of the tracklet will permit it. This lets the user see a portion of the tracklet's trajectory during playback, alleviating many minor localization issues that may confuse what target should be followed during crowded settings. At any point, the user may right-click and specify the target is a false positive or has exited the scene. Once the user clicks the video to start the review, playback begins and continues until the next key-frame is reached. While paused, the user annotates the target's location with a click and drag interaction, which defines the center of the target (i.e., the click) and the object's orientation (i.e., the drag). The length of the drag does not matter. Other interactions could be used, such as bounding box definitions used in [86], but the click and drag procedure is fast. Once the click and drags interaction is complete, playback resumes until the next key-frame. This process is repeated until some stopping condition Φ for the review is met. Said another way, when an action that should be taken is determined. Section-6.2.3 discusses stopping conditions. A sample is shown in figure-6.1, illustrating the user interaction process.

6.2.3 Key-frame selection schemes

Like poorly selecting reviews, a poorly selected set of key-frames can significantly affect user effort and tracking performance. The difference here is that poorly selecting reviews indirectly impacts tracking performance (i.e., it takes longer to achieve some

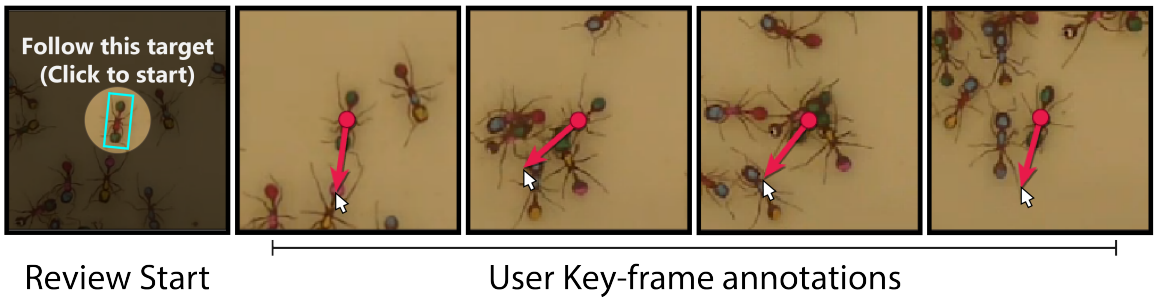

Figure 6.1: An example showing typical user annotation process.

level of matching accuracy). In contrast, a poorly selected set of key-frames can indirectly and directly harm tracking performance (i.e., lead inference to the wrong conclusion). The remainder of this section discusses ways to determine key-frames for user annotation—the final portions of the section detail the proposed key-frame selection approach.

A naïve approach to key-frame scheduling is to employ a static step size over a predefined number of key-frame annotations. Such an approach is used in [16]. Here, no inference is used to determine when to stop requesting user annotations within a single review. As a result, a review can end up falling short of the correct connection tail (a connection tail is a tracklet head) and needs to be re-queued. In other words, no action on the review endpoint gets taken at that time, causing the user to see the review later again and continue key-frame annotations where they left off. The more obvious downside of re-queuing reviews is time wasted transitioning reviews. A less obvious downside is that it can cause some confusion to the user. For example, if a target is not moving much and a review gets re-queued several time for it, it may appear to the user as though they are annotating the same set of frames over and over (i.e., it looks like a bug in the software), even though they are continuing where they left off. Another disadvantage of this approach is that it can result in a review overextending, which means more key-frame annotations were requested than necessary to match with the correct connection tail. Lastly, a small static step size

tends to be the best option to handle the wide variety of situations that corrections will encounter. Although this approach has been shown to work [16], it ignores several sources of information that, if utilized, would allow the key-frame scheduler to determine step size dynamically as well when to stop requesting annotations for a review.

A slight improvement to the previous approach is to continue using a static key-frames step size but automatically decide when to stop requesting annotations. Another way to describe this modification is that the action to take on the review endpoint is determined online, or rather as annotations are being gathered, not as a post-processing step that hopes enough information was requested. The benefit of this is that reviews rarely overextend the correct connection tail, and an action is always determined, so re-queuing reviews are never necessary.

The better alternative to a static key-frame step size is to figure out what frames to annotate dynamically. The simplest method for dynamically determining key-frame step size is to consider the points in time where connection tails occur. Here, the key-frame scheduler will take the set of connection tails and define the next key-frame as the minimum frame within the connection tails beyond the last annotation. The downside of this approach is that playback will often make many unnecessary stops depending on how far away the correct connection tail (temporally) is.

One improvement on the previous approach is to dynamically determine key-frame step sizes based on displacement probability to connection tails. The basic idea is that instead of simply stopping once playback reaches a connection tail, we can consider the proximity between the last annotation and the connection tail to establish if the match is reasonable, thus preventing unnecessary stoppage playback to some extent.

The key-frame selection approach proposed in this dissertation is a dynamic step size method based on proximity to connection tails and their favorability. Let r be the review for the tail (later) endpoint of some tracklet $\tau_i \in \mathbf{T_0}$ and let t_0 be the

Table 6.1: Manual annotation, simulated using ground truth for a single video within the ant colony dataset (section-2.4). The length of the video used is approximately 2.5 minutes and contains 48 targets which cannot enter or exit the scene. Annotations are specified every StepSz frames with linear interpolation. Note that this table is identical to the calculation presented in [16].

StepSz	Num-Annotations	Playback Time	Est. Time	Recall	MT	PT	ML	IDS	FAF
20	12,048	2h 12m	7h 16m	0.99	48	0	0	0	0.16
30	8,064	2h 12m	5h 36m	0.97	48	0	0	2	1.22
40	6,048	2h 12m	4h 46m	0.92	47	1	0	2	3.41
60	4,080	2h 12m	3h 56m	0.82	25	23	0	12	8.61
80	3,072	2h 12m	3h 31m	0.73	16	32	0	45	12.95
100	2,448	2h 12m	3h 16m	0.66	12	36	0	86	16.11
300	864	2h 12m	2h 36m	0.39	7	23	18	313	28.92
500	528	2h 12m	2h 28m	0.30	4	20	24	395	33.41
1000	288	2h 12m	2h 22m	0.21	2	14	32	350	37.44

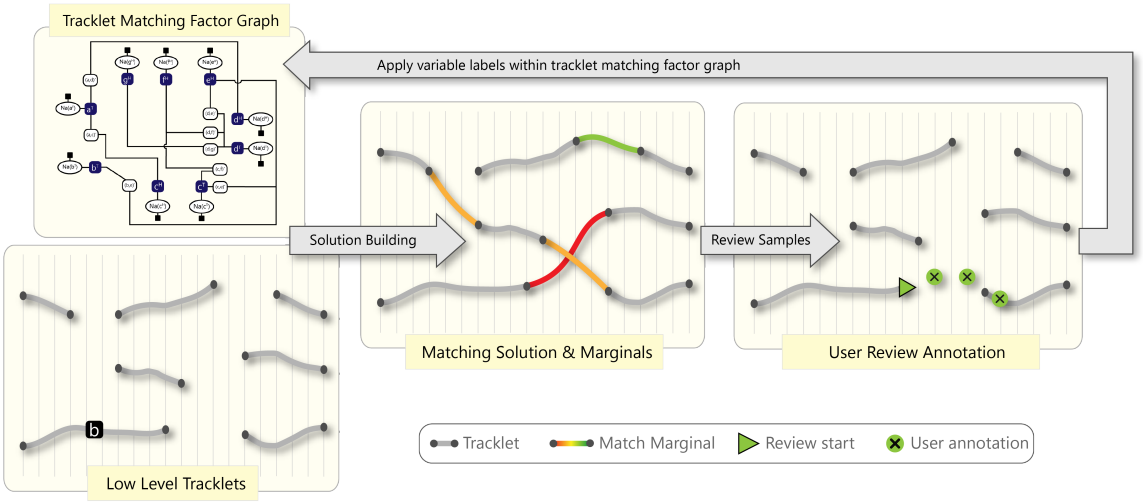

Figure 6.2: Proposed process for gathering and applying user annotations.

frame at which this endpoint occurs. Let $\Gamma^{\hat{\tau}} = \{\hat{\tau}_k\}$ represent the head endpoints of all tracklet that temporally extend past t_0 (thus, these are a set of connection tails). Here, the upward hat on $\hat{\tau}_k$ denotes the head endpoint of the tracklet τ_k — a downward hat $\check{\tau}_k$ would then represent the tail endpoint of the tracklet. Note that this is not the set of tracklets that begin at some point later than t_0, therefore allowing the system to consider tracklet merges (i.e., temporally overlapping trajectories) in addition to connections. We will refer to the set $\Gamma^{\hat{\tau}}$ and the set of possible connection tail endpoints to tracklet τ_i, or, simply, as the connection tails for this review.

Given the review r and its possible connection tails $\Gamma^{\hat{\tau}}$, we repeat the following until an action for the review endpoint is determined: the key-frame scheduler selects the next frame to annotate given the favorability of the connection tails, the annotation is gathered from the user in the selected frame, we update the favorability of the connection tails $\Gamma^{\hat{\tau}}$ given the updated set of annotations, and determine if the annotations have identified an action to take. Once an action has been determined, this endpoint's review process is complete, and labels resulting from the action are applied to the tracklet matching factor graph.

The key-frame scheduler relies primarily on three elements: a measure of favora-

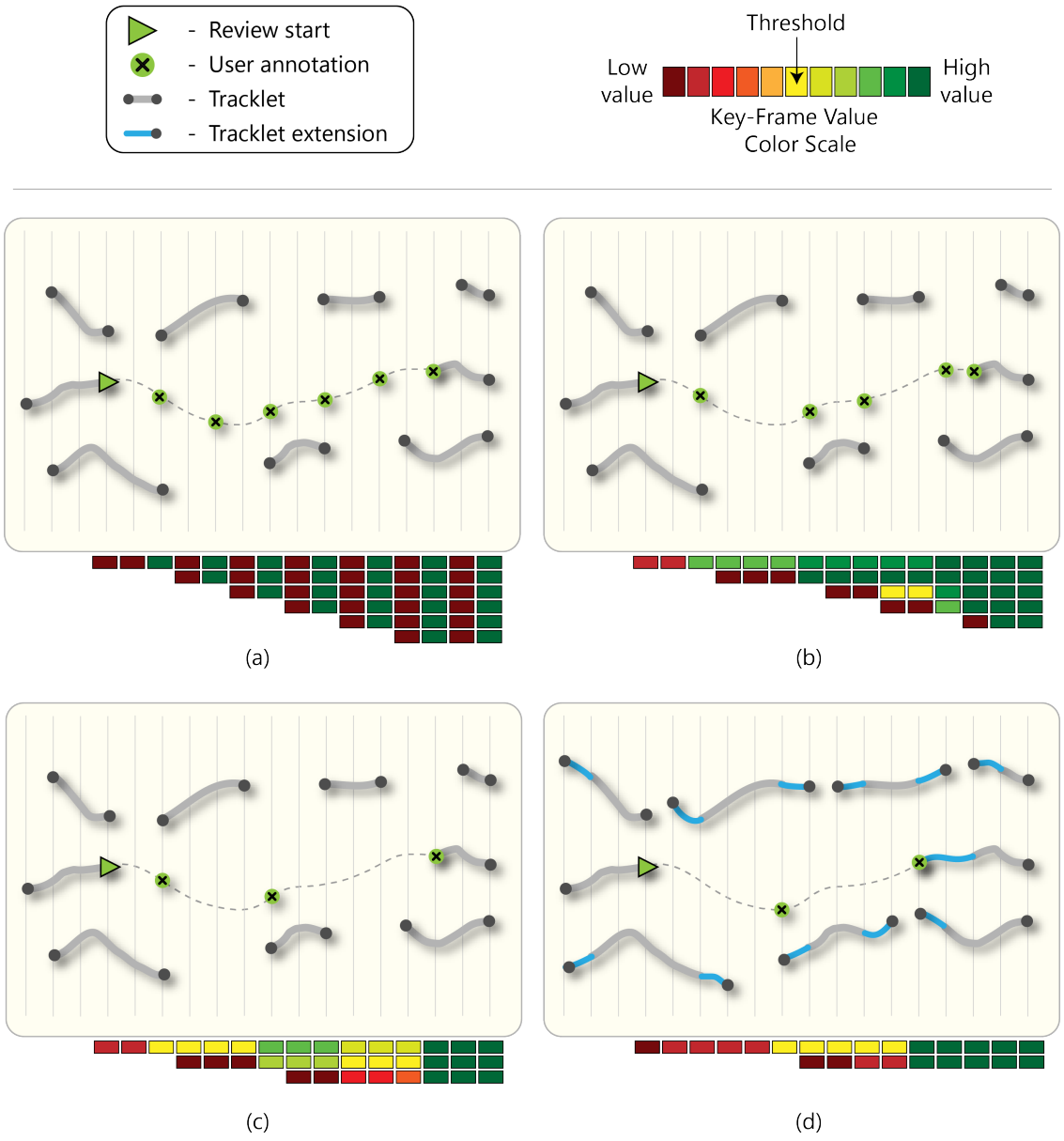

Figure 6.3: Illustration comparing different approaches to key-frame scheduling. (a) Static step size key-frame scheme, (b) dynamic step size based on the beginning frame of connection heads, (c) dynamic step size that that also uses the marginals to determine if a connection head should be stopped at, and (d) is the same as the previous, but with tracklets extended (cyan segments) using confident regions of hypothesis trees.

bility for the connection tails $\Gamma^{\hat{\tau}}$, a favorability threshold θ_{kf} for selecting the frame, and a maximum step-size m. The favorability of the connection tails is calculated in one of two ways depending on whether the connection tail overlaps temporally with the annotations. When a connection tail does not overlap with the annotations, the favorability is defined using an L2 displacement probability measure. Specifically, we consider the average displacement between the last annotation a^{tn} and the location of the connection tail endpoint $\hat{\tau_k} \in \Gamma^{\hat{\tau}}$. Note that at the very start of a review, no annotations are available — thus, we use the location of the review endpoint as a^{tn} instead. The favorability of a non-overlapping connection tail is then:

$$F_n(\hat{\tau_k}, a^{tn}) = P(\frac{\Delta(\hat{\tau_k}, a^{tn})}{Q}) \tag{6.1}$$

where $\Delta(.)$ is the L2 distance between the endpoint location and the last annotation, Q is the number of frames separating the endpoint and the annotation, and $P(.)$ is a half-normal distribution with 0 mean and standard deviation σ^2. The standard deviation is determined automatically based on motion statistics gathered from the low-level tracklets. The favorability of connection tails that have any amount of overlap temporally with the annotations defined by:

$$F_O(\hat{\tau_k}, \{a^t\}^*) = \frac{1}{\|\{a^t\}\|} * \sum_{a^n \in \{a^t\}^*} P(\Delta(\hat{\tau_k}, a^{tn})) \tag{6.2}$$

Where τ_k is the tracklet associated with connection tail $\hat{\tau_k}$, $\{a^t\}^*$ is the subset of annotations overlapping with τ_k, $\Delta(.)$ is the L2 distance between an annotation and the corresponding location in tracklet τ_k, and $P(.)$ is a half-normal distribution with 0 mean and standard deviation σ^2. The standard deviation is determined automatically based on the average size of the targets in the scene.

Once the favorability of all connection tails has been determined, we construct an array v such that each index $v[i]$ in the array represents the maximum favorability

among all connection tails within a particular frame. Specifically, for a connection tail $\hat{\tau_k}$, we say that all frames covered by the tracklet associated with the connection tail, τ_k, have equal favorability, calculated with one of the two functions outlined above. The clarification being made here is that an annotation has value anywhere within a connection tail's tracklet τ_k, not just at the frame in which the connection tail endpoint occurs $\hat{\tau_k}$. The resulting array v allows us to quantify the value of annotating a particular frame. Again, each index $v[i]$ represents the maximum connection tail favorability in a frame, not the sum of favorabilities across the set of connection tails in that frame. Summing the values across the options would allow for many poor options to collectively trigger an annotation request. With the frame annotation value array v, we select the next annotation frame by finding the first element in v that is both $\geq \theta_{kf}$ and less than the maximum step size m. If no elements in v meet these criteria, then the maximum step is used to define the next frame requested for annotation.

After each new user annotation, the favorability of the connection tails is updated, and connection actions are considered for connection tails temporally overlapping with the annotations. A connection action is accepted for a particular connection tail $\hat{\tau_k}$ if the overlapping favorability score $F_O(\hat{\tau_k}, \{a^t\}^*)$ is greater than a predefined threshold θ_c. Note that this means there is a chance that a connection tail will require multiple annotations before it is accepted as the connection action. Because of this, we include an ideal minimum frame step size to prevent annotating consecutive frames, if possible. We set the ideal minimum to be 10, meaning that the next key-frame will be at least ten frames from the last key-frame. The exceptions to this are when the connection tail requesting the annotation does not extend ten frames or when the end of the video has been reached.

6.2.4 Implementation details

As observations are made on nodes in the tracklet matching factor graph, we rebuild a new matching solution using the factor graph and greedy matching on marginal values (after optional sum-product message updates). An illustration is shown in figure-6.2.

As pointed out in chapter-4, Na variables within the factor graph rely on unary terms determined during tree growth. The Na variables can adjust their marginal beliefs as labels are added to the graph, but their unary terms can significantly hinder this. For example, imagine a tree strongly believes an answer is present in the graph because one branch dominates the others in terms of score and has "landed." Even if the answer this branch believes receives a negative confirmation label, the unary term will still push the endpoint to make a connection. This can cause unusual connections to be established. We update the unary term for all Na variables after each batch of reviews is answered to handle this. Branches that landed on endpoints with negative confirmation labels are not counted as such and are ignored during the recalculation of the Na unary terms.

6.3 Evaluation

This chapter compares the various alternatives for correcting tracklet matching errors outlined in this chapter. We first compare review sampling approaches by simulating user corrections and observing the decrease in fragment and ID switch errors. From there, we compare key-frame selection schemes in terms of user time spent. We use one video from the ant colony dataset testing split for all comparisons made in this chapter. Input to the correction simulations is the LBP-Forest results generated according to chapter-5. Correction simulations are performed using ground truth. When calculating estimates of user time spent during simulations, we define: the amount of time a user spends making a single click-and-drag annotation as 1.5

seconds, the number of frames viewed at the start of a review as 30 (i.e., the segment of the trajectory a user sees before the track is lost), the amount of time it takes to transition between reviews as 2 seconds, and the playback speed is set to 30 frames per second.

While comparing review sampling approaches, we ignore estimates of user time spent and concentrate on the reduction of errors in the tracking results. We set the key-frame scheme to use static step sizes with a step size value of 1. We set the key-frame selection scheme as such to eliminate it as a factor towards correcting tracking errors. We compare three review sampling approaches, including random endpoints, density favoring endpoints (i.e., number of possible connections), and an endpoint entropy-based sampler. Tables 6.2-6.4 show the results of the samplers over 8 batches of 50 endpoint reviews. The random sampler resulted in the worst performance, overall only addressing 20 ID switches and one fragment error. The endpoint density sampler (i.e., number of possible connections for an endpoint) showed better performance over random sampling, resulting in one ID switch (57 addressed) and six fragments (31 addressed). Also, recall improved by 13.4 percent points using density sampling. The endpoint entropy sampling approach achieved the overall best performance in terms of errors addressed and the number of reviews needed to get there. At 200 reviews, the endpoint entropy sampler resulted in 5 fragments and one ID switch. Furthermore, recall is 98% at 200 reviews which is two percentage points better than density-based sampling and more than 12 percentage points higher than random sampling.

To compare key-frame selection schemes, we lock in the review sampling approach and observe the reduction in tracking errors in addition to estimates of user time spent. We compare three key-frame sampling schemes: static step size, dynamic step size based on connection endpoints proximity, and dynamic step size based on connection endpoint proximity with marginal dampening. We will refer to these

as static, simple-dynamic, and dampened-dynamic, respectfully. We also include a variant we will refer to as damped-dynamic with extensions for this comparison. For damped-dynamic with extensions, we extend the temporal extent of tracklets with the confident segments of the hypothesis trees grown at its endpoints (figure-6.3, (d)). Tables 6.5-6.8 show the results of the different key-frame selection schemes over 8 batches of 50 endpoint reviews. The static method had the overall worst performance among the methods compared — requiring an estimated user time of 3 hours to perform 400 reviews. While ID switches were mostly corrected using the static approach, the number of fragments was significantly higher than the other methods. This is because the static step size would overshoot the correct answer's endpoint. In terms of fragment and ID switch errors, both the simple-dynamic and the dampened-dynamic approaches achieved the same amount of error reduction at 400 reviews — with dampened-dynamic performing 400 reviews with 27% less time over simple-dynamic (152 minutes vs 110 minutes). Note that the amount of video playback between dampened-dynamic and simple-dynamic is nearly identical. The reduction in time between the two is due to dampened-dynamic requesting fewer annotations. The last approach in this comparison, damped-dynamic with extensions, performed similar to dampened-dynamic but with a slight reduction in overall time — primarily due to slightly less video playback because of the tracklet extensions.

In several of the comparisons, the result comes to 5 fragments and one ID switch. We reviewed the tracking results and found that all 6 of these errors are identical. Upon review, all 5 of the fragment errors were due to tracks of length one. In these cases, an earlier track was being reviewed for which the length-one track would be the correct answer for connection. The connection is not made due to an edge case scenario. Specifically, the short track in these scenarios is within the threshold for mapping to ground truth but far enough away from the ground truth that one annotation does not meet the threshold for making the connection. The one ID switch

that occurs is a track that briefly (less than 30 frames) switches to a target that has not been detected at all since the beginning of the video.

6.4 Summary

In this chapter, we detailed an interactive procedure that allows a user to more efficiently correct tracklet matching errors. We discussed several approaches that could be taken for sampling reviews and selecting key-frames for the user to annotate. Evaluation on the ant colony video containing 50 objects over 5000 frames showed two things. First is the entropy sampling approach based on estimated marginals gathered with LBP-Forest (chapter-5) outperforms both random sampling and a graph density sampling approach. This indicates that the marginals calculated by LBP-Forest help correct tracking errors. Second, the evaluation showed that marginals could also be used to determine key-frames for the user to annotate. The proposed dynamic step size with the marginals dampening approach reduced user time spent over the dynamic step size approach by 27% (42 minutes).

Table 6.2: Random endpoint graph sampler performance over 400 reviews (average of 6 runs, rounded down).

Num. Reviews	Frag	IDS	Rcll	Prcn	FP
0	37	58	84.60%	98.1%	3958
50	35	53	84.7%	98.0%	4129
100	37	51	85.2%	98.2%	3805
150	36	52	85.6%	97.9%	4358
200	39	47	85.8%	98.1%	3922
250	42	41	86.2%	98.6%	2857
300	41	41	85.9%	98.5%	3210
350	37	40	86.6%	98.6%	2939
400	36	38	87.6%	98.8%	2476

Table 6.3: Endpoint Density endpoint graph sampler performance over 400 reviews (8 batches of 50 reviews) .

Num. Reviews	Frag	IDS	Rcll	Prcn	FP
0	37	58	84.60%	98.1%	3958
50	24	31	89.2%	98.8%	2655
100	22	16	92.1%	99.0%	2240
150	16	10	94.7%	99.0%	2328
200	12	5	96.0%	99.3%	1596
250	12	4	96.1%	99.4%	1507
300	10	3	97.3%	99.4%	1484
350	8	4	97.9%	99.4%	1459
400	6	1	98.0%	99.4%	1440

Table 6.4: Endpoint Entropy graph sampler performance over 400 reviews.

Num. Reviews	Frag	IDS	Rcll	Prcn	FP
0	37	58	84.60%	98.1%	3958
50	22	26	91.8%	98.4%	3681
100	13	12	96.4%	98.8%	2698
150	8	2	97.8%	99.4%	1511
200	5	1	98.0%	99.3%	1544
250	5	1	98.2%	99.6%	1030
300	5	1	98.2%	99.4%	1363
350	5	1	98.2%	99.4%	1363
400	5	1	98.2%	99.5%	1303

Table 6.5: Correction performance over 400 reviews (8 batches of 50 reviews) using static step size key-frame selection scheme (figure-6.3, (a)). Step size is set to 30. Review sampling was performed using endpoint entropy.

Num Reviews	Frag	IDS	Rcll	Prcn	FP	Time (Total)	Time (Annot.)	Time (Playback)
0	37	58	84.6%	98.1%	3958	0	0	0
50	21	23	94.6%	95.0%	12039	1:01:25	0:34:51	0:26:34
100	20	12	96.6%	94.9%	12354	1:39:02	0:55:22	0:43:40
150	17	6	97.9%	95.0%	12465	1:57:31	1:04:24	0:53:07
200	14	4	98.3%	95.0%	12416	2:10:50	1:09:52	1:00:58
250	19	2	98.4%	94.0%	14996	2:27:38	1:17:43	1:09:55
300	22	3	98.4%	93.7%	15788	2:41:08	1:23:42	1:17:26
350	27	3	98.6%	93.5%	16562	2:55:42	1:29:48	1:25:54
400	31	3	98.6%	93.1%	17434	3:04:29	1:33:00	1:31:29

Table 6.6: Correction performance over 400 reviews (8 batches of 50 reviews) using a dynamic step size key-frame selection scheme based on proximity to connection endpoints (figure-6.3, (c)). Step size is set to 30. Review sampling was performed using endpoint entropy.

Num Reviews	Frag	IDS	Rcll	Prcn	FP	Time (Total)	Time (Annot.)	Time (Playback)
0	37	58	84.6%	98.1%	3958	0	0	0
50	22	26	91.8%	98.3%	3857	0:47:51	0:30:25	0:17:26
100	13	12	96.3%	98.8%	2923	1:20:15	0:49:30	0:30:45
150	8	2	97.7%	99.3%	1601	1:39:33	0:59:16	0:40:17
200	6	2	97.9%	99.3%	1702	1:50:59	1:04:28	0:46:31
250	6	2	98.0%	99.5%	1249	2:02:54	1:08:18	0:54:36
300	6	2	98.0%	99.5%	1255	2:13:04	1:12:51	1:00:13
350	6	2	98.0%	99.5%	1284	2:21:31	1:15:22	1:06:09
400	5	1	98.1%	99.5%	1227	2:32:29	1:20:03	1:12:26

Table 6.7: Correction performance over 400 reviews (8 batches of 50 reviews) using dynamic step size key-frame selection scheme based on proximity to connection endpoints with negative-belief dampening (figure-6.3, (c)). Review sampling was performed using endpoint entropy.

Num Reviews	Frag	IDS	Rcll	Prcn	FP	Time (Total)	Time (Annot.)	Time (Playback)
0	37	58	84.6%	98.1%	3958	0	0	0
50	24	26	91.8%	98.2%	3953	0:30:28	0:12:52	0:17:35
100	15	13	96.3%	98.6%	3383	0:52:20	0:21:27	0:30:53
150	6	2	97.7%	99.2%	1807	1:06:59	0:26:22	0:40:36
200	5	1	97.8%	99.2%	1856	1:16:57	0:29:22	0:47:35
250	5	1	98.0%	99.4%	1390	1:27:23	0:31:57	0:55:26
300	5	1	98.0%	99.4%	1394	1:35:33	0:34:12	1:01:21
350	5	1	98.0%	99.4%	1448	1:43:45	0:36:06	1:07:39
400	5	1	98.0%	99.4%	1395	1:50:55	0:37:52	1:13:03

Table 6.8: Correction performance over 400 reviews (8 batches of 50 reviews) using dynamic step size key-frame selection scheme based on proximity to connection endpoints with negative-belief dampening and extended tracklets (figure-6.3, (d)). Review sampling was performed using endpoint entropy.

Num Reviews	Frag	IDS	Rcll	Prcn	FP	Time (Total)	Time (Annot.)	Time (Playback)
0	37	58	84.6%	98.1%	3958	0	0	0
50	24	26	91.8%	98.4%	3648	0:28:49	0:12:19	0:16:30
100	15	13	96.0%	98.7%	3017	0:48:38	0:20:03	0:28:35
150	6	2	97.6%	99.3%	1618	1:04:26	0:25:25	0:39:01
200	5	1	97.8%	99.3%	1673	1:12:45	0:27:51	0:44:54
250	5	1	98.0%	99.5%	1207	1:22:28	0:30:34	0:51:54
300	5	1	98.0%	99.5%	1213	1:31:13	0:32:46	0:58:26
350	5	1	98.0%	99.5%	1202	1:39:42	0:34:57	1:04:45
400	5	1	98.0%	99.5%	1278	1:46:31	0:36:45	1:09:46

CHAPTER 7: CONCLUSION

This dissertation described an offline multi-object tracking procedure that presents a means for estimating matching uncertainty. This work ultimately outlines four contributions to multi-object tracking, particularly in videos containing objects with similar appearance and unpredictable motion:

- We presented a SOT procedure for growing a tree-like data structure that models possible paths a target may have taken within "hard-to-track" regions of a video recording.
- We presented bi-directional hypothesis forests for determining two essential elements of the tracklet matching association graph, the set of possible connections and their affinities.
- We outlined how a probabilistic factor graph model is constructed for estimating tracklet matching marginals using loopy belief propagation.
- We presented an interactive procedure that allows a user to correct tracklet matching errors more efficiently.

Several areas could be explored as future works. Concerning hypothesis trees, we've mentioned that several promising areas of improvement are happening in the SOT appearance modeling community, such as [48, 49]. Our hypothesis tree approach should fit well with anything that can provide a heat map-like output. Furthermore, parts-based appearance modeling methods could be explored [50]. Another area that could be further explored pertains to the proposed Bi-Forest method. For one, using some global context when growing the forest, similar to [16], could significantly

improve tracking performance and possibly even reduce computation. Furthermore, calculating affinity measures could using Bi-Forest could be improved. For example, one could determine the most likely path given a pair of trees and calculate affinity based on the generated path's appropriateness.

Concerning LBP-Forest, future works could consider alternative means of calculating affinities and expand upon Na terms to better model possibilities. For example, instead of aggregating several classes of possibilities with the Na term, one could explicitly model terms that represent: the target exiting the scene, false-positive tracks, and even more abstract concepts like tracks that divide and merge. Moreover, our LBP-Forest approach could be modified to operate from detections instead of low-level tracklets.

Finally, there are several ways in which interactively correcting tracking errors could be explored further. Generally speaking, these areas fall into one or more of a few categories. The first category concerns speeding up time spent performing corrections. This could include better approaches for review sampling and key-frame selection schemes. The other category allows the user to define what errors are a higher priority. This would be a practical improvement, as different video analysis situations have different needs. One analysis setting may need to address ID switches over fragments or vice versa. Furthermore, analysis settings may only need a subset of the targets to be corrected. Finally, another practical improvement would be to present some indicator that conveys when to stop answering reviews. Throughout the document, we have made several points relating to the large number of tracklet endpoints that either tracking or corrections must address. Providing a reliable indicator of expected change to the tracking results would allow for either the system to automatically stop requesting reviews or the user to determine on their own when to stop.

REFERENCES

[1] X. Liu, W. Song, L. Fu, W. Lv, and Z. Fang, "Typical features of pedestrian spatial distribution in the inflow process," *Physics Letters A*, vol. 380, no. 17, pp. 1526 – 1534, 2016.

[2] D. Surie, B. Baydan, and H. Lindgren, "Proxemics awareness in kitchen as-a-pal: Tracking objects and human in perspective," in *Intelligent Environments (IE), 2013 9th International Conference on*, pp. 157–164, July 2013.

[3] D. Surie, S. Partonia, and H. Lindgren, "Human sensing using computer vision for personalized smart spaces," in *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*, pp. 487–494, IEEE, 2013.

[4] P. U. Lima, A. Ahmad, A. Dias, A. G. Conceição, A. P. Moreira, E. Silva, L. Almeida, L. Oliveira, and T. P. Nascimento, "Formation control driven by cooperative object tracking," *Robotics and Autonomous Systems*, vol. 63, pp. 68–79, 2015.

[5] V. A. Laurense, J. Y. Goh, and J. C. Gerdes, "Path-tracking for autonomous vehicles at the limit of friction," in *American Control Conference (ACC), 2017*, pp. 5586–5591, IEEE, 2017.

[6] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773–1795, 2013.

[7] M. Brown, J. Funke, S. Erlien, and J. C. Gerdes, "Safe driving envelopes for path tracking in autonomous vehicles," *Control Engineering Practice*, vol. 61, pp. 307 – 316, 2017.

[8] S. Aslani and H. Mahdavi-Nasab, "Optical flow based moving object detection and tracking for traffic surveillance," *International Journal of Electrical, Electronics, Communication, Energy Science and Engineering*, vol. 7, no. 9, pp. 789–793, 2013.

[9] B. Tian, Q. Yao, Y. Gu, K. Wang, and Y. Li, "Video processing techniques for traffic flow monitoring: A survey," in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pp. 1103–1108, IEEE, 2011.

[10] S. Ojha and S. Sakhare, "Image processing techniques for object tracking in video surveillance-a survey," in *Pervasive Computing (ICPC), 2015 International Conference on*, pp. 1–6, IEEE, 2015.

[11] S. Sivanantham, N. N. Paul, and R. S. Iyer, "Object tracking algorithm implementation for security applications," *Far East Journal of Electronics and Communications*, vol. 16, no. 1, p. 1, 2016.

[12] J. E. Franco-Restrepo, D. A. Forero, and R. A. Vargas, "A review of freely available, open-source software for the automated analysis of the behavior of adult zebrafish," *Zebrafish*, vol. 16, no. 3, pp. 223–232, 2019.

[13] V. Panadeiro, A. Rodriguez, J. Henry, D. Wlodkowic, and M. Andersson, "A review of 28 free animal-tracking software applications: Current features and limitations," *Lab animal*, pp. 1–9, 2021.

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

[15] H. Nguyen, T. Fasciano, D. Charbonneau, A. Dornhaus, and M. C. Shin, "Data association based ant tracking with interactive error correction," in *IEEE Winter Conference on Applications of Computer Vision*, pp. 941–946, IEEE, 2014.

[16] L. Rice, S. Tate, D. Farynyk, J. Sun, G. Chism, D. Charbonneau, T. Fasciano, A. Dornhaus, and M. C. Shin, "Abctracker: an easy-to-use, cloud-based application for tracking multiple objects," *arXiv preprint arXiv:2001.10072*, 2020.

[17] N. Razin, J.-P. Eckmann, and O. Feinerman, "Desert ants achieve reliable recruitment across noisy interactions," *Journal of The Royal Society Interface*, vol. 10, no. 82, 2013.

[18] D. P. Mersch, A. Crespi, and L. Keller, "Tracking individuals shows spatial fidelity is a key regulator of ant social organization," *Science*, vol. 340, no. 6136, pp. 1090–1093, 2013.

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

[20] 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*, vol. 29, no. 7, pp. 417–428, 2014.

[21] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 7, pp. 1442–1468, 2013.

[22] A. Dutta, A. Mondal, N. Dey, S. Sen, L. Moraru, and A. E. Hassanien, "Vision tracking: A survey of the state-of-the-art," *SN Computer Science*, vol. 1, no. 1, pp. 1–19, 2020.

[23] H. C. Lu, P. Li, and D. Wang, "Visual object tracking: a survey," *Pattern Recognition and Artificial Intelligence*, vol. 31, no. 1, pp. 61–76, 2018.

[24] S. M. Marvasti-Zadeh, L. Cheng, H. Ghanei-Yakhdan, and S. Kasaei, "Deep learning for visual tracking: A comprehensive survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[25] A. Pérez-Escudero, J. Vicente-Page, R. C. Hinz, S. Arganda, and G. G. De Polavieja, "idtracker: tracking individuals in a group by automatic identification of unmarked animals," *Nature methods*, vol. 11, no. 7, pp. 743–748, 2014.

[26] T. Fasciano, A. Dornhaus, and M. C. Shin, "Ant tracking with occlusion tunnels," in *IEEE Winter Conference on Applications of Computer Vision*, pp. 947–952, IEEE, 2014.

[27] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A survey of deep learning-based object detection," *IEEE access*, vol. 7, pp. 128837–128868, 2019.

[28] J. Peng, T. Wang, W. Lin, J. Wang, J. See, S. Wen, and E. Ding, "Tpm: Multiple object tracking with tracklet-plane matching," *Pattern Recognition*, vol. 107, p. 107480, 2020.

[29] Y. Li, C. Huang, and R. Nevatia, "Learning to associate: Hybridboosted multi-target tracker for crowded scene," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 2953–2960, IEEE, 2009.

[30] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[31] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 300–311, 2017.

[32] C. Dicle, O. I. Camps, and M. Sznaier, "The way they move: Tracking multiple targets with similar appearance," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2304–2311, 2013.

[33] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "MOTChallenge 2015: Towards a benchmark for multi-target tracking," *arXiv:1504.01942 [cs]*, Apr. 2015. arXiv: 1504.01942.

[34] D. Chicco, "Siamese neural networks: An overview," *Artificial Neural Networks*, pp. 73–94, 2021.

[35] R. Pflugfelder, "An in-depth analysis of visual tracking with siamese neural networks," *arXiv preprint arXiv:1707.00569*, 2017.

[36] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8971–8980, 2018.

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

[38] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *European conference on computer vision*, pp. 850–865, Springer, 2016.

[39] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1619–1632, 2011.

[40] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, "Struck: Structured output tracking with kernels," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2096–2109, 2016.

[41] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, J.-K. Kämäräinen, M. Danelljan, L. Č. Zajc, A. Lukežič, O. Drbohlav, *et al.*, "The eighth visual object tracking vot2020 challenge results," in *European Conference on Computer Vision*, pp. 547–601, Springer, 2020.

[42] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "Siamrpn++: Evolution of siamese visual tracking with very deep networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4282–4291, 2019.

[43] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, "Fast online object tracking and segmentation: A unifying approach," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1328–1338, 2019.

[44] Z. Zhu, W. Wu, W. Zou, and J. Yan, "End-to-end flow correlation tracking with spatial-temporal attention," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 548–557, 2018.

[45] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 101–117, 2018.

[46] D. Reid, "An algorithm for tracking multiple targets," *IEEE transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[47] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, "Multiple hypothesis tracking revisited," in *Proceedings of the IEEE international conference on computer vision*, pp. 4696–4704, 2015.

[48] Y. Zhang, L. Wang, J. Qi, D. Wang, M. Feng, and H. Lu, "Structured siamese network for real-time visual tracking," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 351–366, 2018.

[49] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. Lau, and M.-H. Yang, "Vital: Visual tracking via adversarial learning," in *Proceedings of*

the IEEE conference on computer vision and pattern recognition, pp. 8990–8999, 2018.

[50] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge, "Deeplabcut: markerless pose estimation of user-defined body parts with deep learning," *Nature neuroscience*, vol. 21, no. 9, pp. 1281–1289, 2018.

[51] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-backward error: Automatic detection of tracking failures," in *Pattern recognition (ICPR), 2010 20th international conference on*, pp. 2756–2759, IEEE, 2010.

[52] W. Wang, R. Nevatia, and B. Yang, "Beyond pedestrians: A hybrid approach of tracking multiple articulating humans," in *2015 IEEE Winter Conference on Applications of Computer Vision*, pp. 132–139, 2015.

[53] A. Milan, R. Gade, A. Dick, T. B. Moeslund, and I. Reid, "Improving global multi-target tracking with local updates," in *European Conference on Computer Vision*, pp. 174–190, Springer, 2014.

[54] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking without bells and whistles," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 941–951, 2019.

[55] G. Brasó and L. Leal-Taixé, "Learning a neural solver for multiple object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6247–6257, 2020.

[56] T. Zhou, J. Li, X. Li, and L. Shao, "Target-aware object discovery and association for unsupervised video multi-object segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6985–6994, 2021.

[57] J. Luiten, I. E. Zulfikar, and B. Leibe, "Unovost: Unsupervised offline video object segmentation and tracking," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2000–2009, 2020.

[58] M. Li, X. Zhu, and S. Gong, "Unsupervised noisy tracklet person re-identification," *arXiv preprint arXiv:2101.06391*, 2021.

[59] M. Ye, J. Shen, G. Lin, T. Xiang, L. Shao, and S. C. Hoi, "Deep learning for person re-identification: A survey and outlook," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

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

[61] K. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," *arXiv preprint arXiv:1301.6725*, 2013.

[62] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[63] J. L. Williams and R. A. Lau, "Data association by loopy belief propagation," in *2010 13th International Conference on Information Fusion*, pp. 1–8, IEEE, 2010.

[64] J. L. Williams and R. A. Lau, "Convergence of loopy belief propagation for data association," in *2010 Sixth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pp. 175–180, IEEE, 2010.

[65] M. Chertkov, L. Kroc, F. Krzakala, M. Vergassola, and L. Zdeborová, "Inference in particle tracking experiments by passing messages between images," *Proceedings of the National Academy of Sciences*, vol. 107, no. 17, pp. 7663–7668, 2010.

[66] F. Meyer, T. Kropfreiter, J. L. Williams, R. Lau, F. Hlawatsch, P. Braca, and M. Z. Win, "Message passing algorithms for scalable multitarget tracking," *Proceedings of the IEEE*, vol. 106, no. 2, pp. 221–259, 2018.

[67] R. A. Lau and J. L. Williams, "A structured mean field approach for existence-based multiple target tracking," in *2016 19th International Conference on Information Fusion (FUSION)*, pp. 1111–1118, IEEE, 2016.

[68] J. L. Williams and R. A. Lau, "Multiple scan data association by convex variational inference," *IEEE Transactions on Signal Processing*, vol. 66, no. 8, pp. 2112–2127, 2018.

[69] W. Choi and S. Savarese, "A unified framework for multi-target tracking and collective activity recognition," in *European Conference on Computer Vision*, pp. 215–230, Springer, 2012.

[70] C.-Y. Chong, "Graph approaches for data association," in *2012 15th International Conference on Information Fusion*, pp. 1578–1585, IEEE, 2012.

[71] H. Jiang, S. Fels, and J. J. Little, "A linear programming approach for multiple object tracking," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2007.

[72] A. V. Segal and I. Reid, "Latent data association: Bayesian model selection for multi-target tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2904–2911, 2013.

[73] A. Dehghan, Y. Tian, P. H. Torr, and M. Shah, "Target identity-aware network flow for online multiple target tracking," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1146–1154, IEEE, 2015.

[74] A. Dehghan, S. Modiri Assari, and M. Shah, "Gmmcp tracker: Globally optimal generalized maximum multi clique problem for multiple object tracking," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[75] A. R. Zamir, A. Dehghan, and M. Shah, "Gmcp-tracker: Global multi-object tracking using generalized minimum clique graphs," in *Computer Vision–ECCV 2012*, pp. 343–356, Springer, 2012.

[76] S. Tang, B. Andres, M. Andriluka, and B. Schiele, "Multi-person tracking by multicut and deep matching," in *European Conference on Computer Vision*, pp. 100–111, Springer, 2016.

[77] Z. Wu, T. H. Kunz, and M. Betke, "Efficient track linking methods for track graphs using network-flow and set-cover techniques," in *CVPR 2011*, pp. 1185–1192, IEEE, 2011.

[78] Z. Wu, T. H. Kunz, and M. Betke, "Efficient track linking methods for track graphs using network-flow and set-cover techniques," in *CVPR 2011*, pp. 1185–1192, 2011.

[79] B. Yang and R. Nevatia, "An online learned crf model for multi-target tracking," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2034–2041, IEEE, 2012.

[80] A. Milan, K. Schindler, and S. Roth, "Multi-target tracking by discrete-continuous energy minimization," *IEEE TPAMI*, vol. 38, pp. 2054–2068, Oct 2016.

[81] A. Milan, S. Roth, and K. Schindler, "Continuous energy minimization for multi-target tracking," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 1, pp. 58–72, 2013.

[82] A. Milan, K. Schindler, and S. Roth, "Multi-target tracking by discrete-continuous energy minimization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 2054–2068, 2015.

[83] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European conference on computer vision*, pp. 17–35, Springer, 2016.

[84] G. Elidan, I. McGraw, and D. Koller, "Residual belief propagation: Informed scheduling for asynchronous message passing," *arXiv preprint arXiv:1206.6837*, 2012.

[85] S. Jo, J. Yoo, and U. Kang, "Fast and scalable distributed loopy belief propagation on real-world graphs," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 297–305, 2018.

[86] C. Vondrick, D. Patterson, and D. Ramanan, "Efficiently scaling up crowdsourced video annotation," *International journal of computer vision*, vol. 101, no. 1, pp. 184–204, 2013.