Multi-Agent Reinforcement Learning for Multi-Object Tracking

Pol Rosello
Stanford University
Stanford, California
pol@cs.stanford.edu

Mykel J. Kochenderfer
Stanford University
Stanford, California
mykel@cs.stanford.edu

ABSTRACT

We present a novel, multi-agent reinforcement learning formulation of multi-object tracking that treats creating, propagating, and terminating object tracks as actions in a sequential decision-making problem. In our formulation, each agent tracks a single object at a time by updating a Bayesian filter according to a discrete set of actions. At each timestep, the reward received is dependent on the joint actions taken by all agents and the ground truth object tracks. We optimize for different tracking metrics directly while propagating covariance information about each object's state. We use trust region policy optimization (TRPO) to train a shared policy across all agents, parameterized by a multi-layer neural network. Our experiments show an improvement in tracking accuracy over similar state-of-the-art, rule-based approaches on a popular multi-object tracking dataset.

KEYWORDS

Learning and Adaptation—Multiagent learning; Engineering Multiagent Systems—Innovative agents and multiagent applications; Learning and Adaptation—Deep learning

ACM Reference Format:
Pol Rosello and Mykel J. Kochenderfer. 2018. Multi-Agent Reinforcement Learning for Multi-Object Tracking. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 8 pages.

1 INTRODUCTION

Fast, online multi-object tracking is desirable in many applications. In autonomous driving, tracking surrounding cars helps to predict their behavior. In biology, tracking fluorescently-labeled cells in vivo can help diagnose disease. In air traffic control, monitoring the state of multiple aircraft is crucial to maintaining safety.

In the multi-object tracking problem, we receive a set of noisy object detections from a sensor at every timestep, and the goal is to use these detections to estimate a set of object tracks. Multi-object tracking is more challenging than single-object tracking, where the main challenges include providing robustness to noise, false positives, false negatives, and occlusions. First, we must address data association. Since there are multiple objects, we might receive multiple detections at the same timestep. In order to update our belief about the current position of an object, we have to know what detections to use, if any. Second, the number of objects present changes over time. We have to decide when to start new tracks (without starting them for false positives) and when to terminate old tracks (without ending them for false negatives or temporarily occluded objects).

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden.

Multi-object tracking algorithms can be vision-based or vision-free. Vision-based approaches often develop a template for the appearance of objects to make them easier to track and detect over time [7, 24]. However, in certain scenarios, such as when tracking visually indistinguishable objects, visual appearance does not help. Furthermore, there are many domains where the detection sensor is not a camera, and where algorithms for extracting object appearance may not be as developed. Finally, recent work has shown that tracking performance for vision-based algorithms is highly dependent on the performance of the vision component, making it difficult to compare the tracking components of algorithms with different vision pipelines [3]. Instead, our paper focuses on vision-free algorithms like multiple-hypothesis tracking [4] or joint probability data association [9], which rely on object motion alone.

Tracking algorithms can be further divided into online and batch-based approaches. In online tracking (or filtering), the goal is to update tracks every timestep given the detections received so far. In contrast, batch-based tracking (or smoothing) considers the entire sequence of detections and optimizes object tracks with the benefit of hindsight [1, 6, 31]. Our work focuses on efficient online approaches that can run in real time.

We develop an algorithm using multi-agent reinforcement learning for multi-object tracking (MARLMOT). MARLMOT is an online, vision-free, supervised tracking algorithm that optimizes the sequential decisions involved in updating a bank of Bayesian filters. With each new set of detections, multiple agents act simultaneously. Each agent manages a separate Kalman filter, and decides when to create, propagate, or terminate object tracks. We associate detections to agents according to each filter's prior prediction about the tracked object's position. We then compute each agent's posterior hypothesis about the object's position, and compute the joint reward for all agents according to ground truth labels. The joint tracking policy is parameterized by a multi-layer neural network with its weights shared across all agents, and optimized using policy gradient methods. Our approach can tentatively track objects or estimate the states of occluded objects, including their covariances.

We evaluate MARLMOT on a simulated multi-object tracking scenario and on the MOTChallenge tracking dataset. On the simulated scenario, we show improved robustness to long-term occlusions, motion model failures, and identity switches compared to the well-performing, rule-based SORT algorithm. Our results on MOTChallenge indicate state-of-the-art performance on most popular tracking metrics for online, vision-free approaches.

2 RELATED WORK

Bayesian approaches such as the probability hypothesis density (PHD) filter and its variants have been used for multi-object tracking [26, 27]. While they can estimate a posterior distribution over the locations of objects and the number of objects present in a given

instant, these approaches do not explicitly track object identities over time. Maintaining object identities is important in settings where providing occupancy probability densities in space is not enough, such as when attempting to predict the intentions of drivers from past behavior. They also require priors over the object birth and death process, as well as over the distribution of the false positive clutter, both of which can be difficult to estimate.

Multiple-hypothesis tracking and joint probability data association are two well-understood approaches to identity-based, online tracking [4, 9]. However, their complexity increases exponentially as the number of objects is increased, making them impractical for most settings. One of the simplest approaches to multi-object tracking is to use the Hungarian algorithm to associate detections to existing object tracks and update their positions using a Kalman filter [13, 15]. In practice, this algorithm is vulnerable to false positives, false negatives, and occlusions. However, recent work has shown that by making a few modifications, this approach can actually outperform the state of the art given an appropriate detection algorithm [3]. In the simple, online, realtime tracking (SORT) algorithm, objects are tentatively tracked, and tracks are not reported until the object has been successfully associated to a detection for a specific number of timesteps. Similarly, tracks are not terminated until no detection has been associated to the track for a number of timesteps. Furthermore, detection associations above a certain cost threshold are ignored. While SORT is fast, easy to implement, and performs very well in practice, it cannot handle longer-term occlusions, re-entering objects, or a failure by the Kalman filter to account for the motion of the objects. Furthermore, if the nature of the tracking scenarios is changed, the parameters of the algorithm must be manually readjusted.

Several recent papers have explored supervised tracking, where the tracking task is learned from ground truth labels [14, 17]. In a recent approach, the authors train recurrent neural networks to perform filtering and data association [19]. Another recent algorithm performed well on video sequences by modeling the lifetime of each object as a Markov decision process [29]. Deterministic actions determine the transitions between states, and a predetermined tracking policy determines how to track at each state. The reward function is learned from a labeled dataset using inverse reinforcement learning [22], and it is parameterized as a function of visual object features, including optical flow. As such, it is not directly comparable to our work. Similarly, recent work has explored using deep reinforcement learning to track objects in videos, but does not handle multiple objects or address vision-free tracking [30].

Supervised approaches to tracking are frequently forced to use proxy tracking metrics during training, instead of the target metrics used for evaluation such as MOTA and MOTP [2]. Usually, the target metric cannot be used during training either because it is not differentiable or because it depends on the joint tracking decisions made for all objects at a given point in time. For example, the MOTA metric is defined as

$$MOTA = 1 - \frac{\sum_{t=1}^{T}(m_t + fp_t + mme_t)}{\sum_{t=1}^{T} g_t} \tag{1}$$

where m_t, fp_t, and mme_t are the number of missed objects, false positives, and mismatch errors with respect to the ground truth at time t, while g_t is the total number of objects present at time t. This computation cannot be incorporated into an analytical loss function as it involves an algorithm with zero-gradient components. Our approach reformulates multi-object tracking as a multi-agent reinforcement learning problem [5], such that learning the optimal policy for an agent is equivalent to optimizing the desired tracking metric. We leverage a structure similar to the traditional Kalman filter and Hungarian algorithm approach, but rather than rely on heuristics such as SORT to deal with detector noise, we use ground truth labels to train our algorithm to take intelligent sequences of tracking decisions.

3 APPROACH

We model the dynamics of objects using linear motion and measurement models, and perform data association using the Hungarian algorithm. We create a bank of tracking agents, each of which manages a single object track at a time. Agents interact in lockstep in a multi-agent reinforcement learning environment. The actions taken by each agent determine how to update a Kalman filter, and the reward received during training is dependent on the joint tracking performance relative to ground truth object tracks.

3.1 Model

We model each object as a 7-dimensional vector changing over time:

$$\boldsymbol{x}_t^i = (p_t^x, p_t^y, v_t^x, v_t^y, v_t^s, s_t^a, s_t^r)^\top \tag{2}$$

where (p_t^x, p_t^y) is the relative position of the object, (v_t^x, v_t^y) is the relative velocity of the object, s_t^a is the area of the object bounding box, and s_t^r is the aspect ration of the bounding box. We include the velocity v_t^s of the object's changing area over time to account for the object moving closer or further away from the sensor. Additionally, an object has an associated unique ID.

A detection $\boldsymbol{d}_t^j$ is a noisy measurement of the position and size of some object:

$$\boldsymbol{d}_t^j = (\tilde{p}_t^x, \tilde{p}_t^y, \tilde{s}_t^a, \tilde{s}_t^r)^\top \tag{3}$$

There is no ID associated with a detection — it is unknown what object is responsible for it, if any. A detection may also include a scalar, real-valued score c_t^j indicating the confidence of the detection algorithm in reporting $\boldsymbol{d}_t^j$.

We maintain N simultaneous tracks. A track consists of a filter, a track mode (inactive, visible, or hidden), and a track ID. The filter provides both a prior prediction $\hat{\boldsymbol{x}}_{t|t-1}^i$ about the true state $\boldsymbol{x}_i$ of the object and a posterior hypothesis $\hat{\boldsymbol{x}}_{t|t}^i$ of the state of the object given a new detection. We use a linear Kalman filter for this purpose, with a constant velocity Euler motion model:

$$\boldsymbol{x}_t^i = \boldsymbol{F}\boldsymbol{x}_{t-1}^i + \boldsymbol{w} \tag{4}$$

Above, $\boldsymbol{F}$ is a matrix representing the linear motion model, and $\boldsymbol{w}$ represents the acceleration process noise, which is assumed to be drawn from a zero-mean multi-variate normal distribution with covariance matrix $\boldsymbol{Q}$. We initialize state estimates using the position of the detection and zero velocity, and use an initial state estimate covariance matrix $\boldsymbol{P}_{\text{init}}$ that assigns a high covariance to the velocity components. Finally, we use a linear measurement model for the

detections:

$$\boldsymbol{d}_t^i = \boldsymbol{H}\boldsymbol{x}_t^i + \boldsymbol{v} \tag{5}$$

Above, $\boldsymbol{H}$ is a matrix representing the measurement process, and $\boldsymbol{v}$ represents the measurement noise, which is assumed to be drawn from a zero-mean multi-variate normal distribution with covariance matrix $\boldsymbol{R}$. In our MOTChallenge experiments, we set $\boldsymbol{F}$, $\boldsymbol{Q}$, $\boldsymbol{P}_{\text{init}}$, $\boldsymbol{H}$, and $\boldsymbol{R}$ to the values in the publicly-available implementation of the SORT algorithm [3]. Although we use a Kalman filter to easily compare results against existing approaches, any type of motion filter can be used in practice.

3.2 Data Association

Our approach requires a cost function $C(\boldsymbol{d}_t^i, \hat{\boldsymbol{x}}_{t|t-1}^j)$ that returns the cost of associating a detection $\boldsymbol{d}_t^i$ to an object prediction $\hat{\boldsymbol{x}}_{t|t-1}^j$. The cost function can be $1 - P(\boldsymbol{d}_t^i \mid \hat{\boldsymbol{x}}_{t|t-1}^j)$ if there is a suitable estimate for it. In our case, we use negative intersection over union between the bounding box of the detection and the bounding box of the prediction. Therefore, if a detection mostly overlaps with the predicted location of an existing object track, the cost of associating the two will be low.

At each timestep t, we receive a set of M detections $\boldsymbol{d}_t^1, \ldots, \boldsymbol{d}_t^M$. We use the filter for each track to predict the current position of its associated object. We use the Hungarian algorithm to efficiently find the bijection between detections and predictions that minimizes the sum of the costs [15]. Note that N and M may not be equal; some detections may not have an associated prediction, and vice-versa. Once data association is complete, we formulate the creation, propagation, and termination of object tracks as a multi-agent sequential decision-making problem.

3.3 Multi-Object Tracking Environment

At any point in time, the internal mode of each track can be either visible, hidden, or inactive. Visible and hidden tracks are associated to a unique object in the scene, but only visible tracks are exposed to the downstream system. Initially, all tracks are inactive. Once a track becomes visible, it receives a new, unique object ID. The hidden track state allows MARLMOT to tentatively track objects until there are enough detections to report an object present with high confidence, or to continue tracking objects if they temporarily leave the scene but later return. A visible track becoming inactive signifies an object leaving the scene.

Figure 1 outlines the tracking environment. Episodes begin by randomly selecting one of the tracking scenarios provided by our dataset. At each timestep t in the episode:

(1) For all i, compute $\hat{\boldsymbol{x}}_{t|t-1}^i$, the filter's prior prediction of the object's state. If the filter is inactive, set $\hat{\boldsymbol{x}}_{t|t-1}^i$ to the zero vector.
(2) Perform data association between the prior predictions and the detections at the current timestep as described in Section 3.2. If $M > N$, detections that are not associated to a prediction are discarded.
(3) For all i, let $\boldsymbol{d}_t^i$ be the track's associated detection. If $M < N$, a track may not have been associated with a detection. In that case, set $\boldsymbol{d}_t^i$ to the zero vector. Observe $\boldsymbol{o}_t^i$, as defined below. Take an action a_i^t that determines how to estimate $\hat{\boldsymbol{x}}_{t|t}^i$ and the new internal mode of the track (see Table 1).
(4) During training, compute the joint reward r_t using the posterior hypothesis $\hat{\boldsymbol{x}}_{t|t}^i$ and the true object positions $\boldsymbol{x}_{t|t}^i$ according to the desired tracking metric using only the estimates of visible trackers.

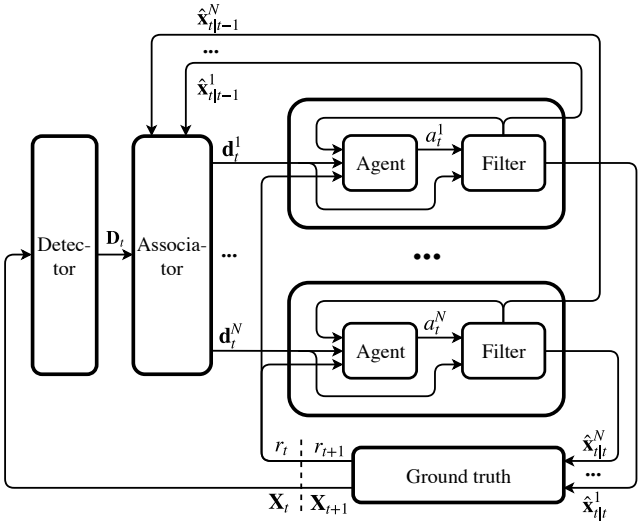

Figure 1: The MARLMOT multi-agent environment architecture. At each timestep, we associate detections with existing tracks. Each agent computes a new object hypothesis by observing its associated detection and taking an action to update a filter. The agents then receive a reward based on the ground truth.

The real-valued observations $\boldsymbol{o}_t^i \in \mathbb{R}^{18}$ of each agent include the prior prediction $\hat{\boldsymbol{x}}_{t|t-1}^i \in \mathbb{R}^7$ of the agent, the associated detection $\boldsymbol{d}_t^i \in \mathbb{R}^4$, the confidence of the detection $c_t^i \in \mathbb{R}$, the cost of the association $C \in \mathbb{R}$, and the internal mode of the track as a one-hot encoding $\boldsymbol{v}_i^t \in \mathbb{R}^3$. Also, we include the number of timesteps since the last association to the agent while the agent is in the active or hidden state ($n_{\text{unassoc}} \in \mathbb{N}$), and the number of consecutive timesteps that an association has been received by the agent while it is in the active or hidden state ($n_{\text{streak}} \in \mathbb{N}$). Agent observations therefore include information about the past, and the vectors $\hat{\boldsymbol{x}}_{t|t}^i$ alone do not fully represent the underlying, unobservable state of the environment.

The actions available to each tracker agent are designed to address situations encountered during tracking:

- Action a_1 terminates an object track, and should be taken when an object leaves the scene.
- Action a_2 restarts an object track, and should be taken when an object enters the scene. Action a_2 can also be helpful to rectify a drifting track during a failure of the motion model to account for an object's trajectory, such as during a period of sharp acceleration.

Table 1: Action space of each track agent.

Action	$\hat{\boldsymbol{x}}^i_{t\|t-1}$	$\boldsymbol{d}^i_t$	New object state estimate	New track mode
a_1	Ignore	Ignore	$\hat{\boldsymbol{x}}^i_{t\|t}, \boldsymbol{P}^i_{t\|t} \leftarrow \varnothing$	Inactive
a_2	Ignore	Use	$\hat{\boldsymbol{x}}^i_{t\|t}, \boldsymbol{P}^i_{t\|t} \leftarrow \boldsymbol{d}^i_t, \boldsymbol{P}_{\text{init}}$	Visible
a_3	Use	Use	$\hat{\boldsymbol{x}}^i_{t\|t}, \boldsymbol{P}^i_{t\|t} \leftarrow \textsc{Update}(\hat{\boldsymbol{x}}^i_{t\|t-1}, \boldsymbol{P}^i_{t\|t-1}, \boldsymbol{d}^i_t)$	Visible
a_4	Use	Ignore	$\hat{\boldsymbol{x}}^i_{t\|t}, \boldsymbol{P}^i_{t\|t} \leftarrow \hat{\boldsymbol{x}}^i_{t\|t-1}, \boldsymbol{P}^i_{t\|t-1}$	Visible
a_5	Use	Ignore	$\hat{\boldsymbol{x}}^i_{t\|t}, \boldsymbol{P}^i_{t\|t} \leftarrow \hat{\boldsymbol{x}}^i_{t\|t-1}, \boldsymbol{P}^i_{t\|t-1}$	Hidden

- Action a_3 performs a filter update using the provided detection, which helps remove detection noise as in the single-object tracking paradigm.
- Action a_4 ignores the associated detection and uses the filter prediction to update the estimated object state. This action can be helpful during false negatives (no associated detection), false positives (when the associated detection does not correspond to an actual object) or data association failures (when the associated detection belongs to a different object than the one currently tracked).
- Action a_5 is similar to a_4, but it puts the tracker in a hidden state. This action is helpful during periods of occlusion, and allows the agent to tentatively track an object without revealing it to the downstream system until it is certain the object exists.

When interacting in scenario i, we set the joint reward at time t for all agents to

$$r^i_t = \frac{1}{T_i} - \frac{m_t + fp_t + mme_t}{\sum_{t'=1}^{T_i} g^i_{t'}} \tag{6}$$

where m_t, fp_t, and mme_t are the number of missed objects, false positives, and mismatch errors with respect to the ground truth at time t, while g^i_t is the total number of objects present at time t in the ground truth scenario i. We compute missed objects, false positives, and mismatch errors as defined in the popular multi-object tracking accuracy (MOTA) metric [2]. The MOTA metric is defined as

$$MOTA_i = 1 - \frac{\sum_{t=1}^{T_i}(m_t + fp_t + mme_t)}{\sum_{t=1}^{T_i} g^i_t} \tag{7}$$

Therefore, the total return $\sum_{t=1}^{T_i} r^i_t$ obtained in scenario i is equal to the MOTA obtained for the scenario. Episodes terminate once there are no more frames. In practice, we could set $r^i_{T_i} = MOTA_i$, and $r^i_t = 0$ for all other t, but we find that providing reward feedback throughout the trajectory helps guide the optimization. In the MOTChallenge, the overall MOTA metric is not equivalent to the average MOTA across all scenarios; rather, it is computed with respect to the total number of ground truth objects across all scenarios. In that case, we simply set the reward to $r^i_t = -(m_t + fp_t + mme_t)$.

3.4 Model Architecture

We want to learn a single policy π that we can use for all N tracks simultaneously to decide how to act at every timestep. It is intractable to optimize policies that consider all joint combinations of actions by the N agents, since the dimensionality of the action space becomes exponential with respect to N [25]. We instead solve a multi-agent RL problem with N agents sharing the same policy.

Since the observation space is continuous, neural networks are a good choice to optimize the policy of the agents. We parameterize π by a 3-layer, fully-connected neural network, with hidden layers of 128, 64, and 32 hidden units, in that order. Hidden units use the ReLU activation function [21], while the output layer applies a softmax function over the 5 possible actions.

We use the gradient-based trust-region policy optimization (TRPO) [23] to optimize π using detections and ground truth labels provided by the MOTChallenge dataset. Unlike other policy optimization algorithms, TRPO guarantees monotonic improvement in the expected discounted cost of the policy. Although we could have chosen a more straightforward reinforcement learning algorithm such as DQN [20], Gupta et al. have shown TRPO to perform better in cooperative environments than other deep reinforcement learning algorithms such as DQN and DDPG [18]. In practice, we found that training was more stable when we used TRPO to optimize the policy network. We use OpenAI's `rllab` implementation of TRPO [8], modified for the multi-agent setting [11]. We use a learning rate of 10^{-4}. Although a baseline function is sometimes used alongside TRPO to reduce the variance of the gradient estimate [10], we do not use one in our implementation. We use a discount factor $\gamma = 0.95$ to stabilize training.

To handle the multi-agent nature of the problem, we perform joint policy rollouts for all agents in lockstep. During the rollout, we accumulate the joint reward for all agents based on the ground truth. The rest of the training procedure follows the TRPO algorithm as if all rollouts in a batch came from a single agent. At the end of the rollout (i.e., once there are no more detections), we update the policy network's weights according to the gradient of the TRPO loss for the joint rewards. For conveninence during training, we bound N to some fixed number of agents. However, since the policy network weights are shared across agents, MARLMOT can track any number of objects at test time. We dynamically create as many agents as needed to associate every detection, and forward each agent's observation through the trained policy network to obtain the agent's action at that timestep.

Tracking environments such as the MOTChallenge pose additional hurdles. In the MOTChallenge, tracking scenarios vary widely with respect to their frame size and frame rate. In some scenarios, the camera is fixed high above a building, while in others, it is moved along a sidewalk. We found that normalizing observations by the width of the frame size and further normalizing them to be in the range $[0, 1]$ was important to stabilize training. To prevent

exploding observations, the n_{unassoc} and n_{streak} observations are normalized by applying the sigmoid function to their values. Furthermore, we set the batch size to the number of tracking scenarios in the training set, such that each batch includes a rollout for each of the scenarios for all agents. Each batch is therefore representative of the scenario distribution on the training set, which we expect to be similar to the distribution on the test set.

4 EXPERIMENTAL RESULTS

We evaluate our approach on both a simulated multi-object tracking scenario and on the 2015 MOTChallenge tracking dataset [16]. The simulated scenario allows us to easily compare the qualitative performance of our algorithm against rule-based approaches like SORT. By controlling the tracking scenario, we can understand the decisions of the learned policy network in challenging situations. The 2015 MOTChallenge compares tracking algorithms on the task of pedestrian tracking in videos. The dataset provides detections and ground truth tracks in the form of bounding boxes, allowing us to quantitatively assess the performance of MARLMOT in real-world tracking scenarios.

4.1 Simulated Scenario

A challenging synthetic scenario illustrates the differences between our supervised tracking algorithm and the rule-based SORT algorithm. An instance of this tracking scenario is shown in Figure 2. In each instance of the scenario, objects spawn at $p_y = 0$ at random times, initially traveling upwards at a constant velocity. The starting p_x position of an object is chosen uniformly at random between 0.1 and 0.5. Objects that spawn at $p_x < 0.3$ travel straight, while objects that spawn at $p_x > 0.3$ travel straight and then turn sharply to the right. We simulate a detection sensor by applying Gaussian noise to the ground truth bounding boxes according to the measurement noise covariance matrix $\boldsymbol{R}$, stripping object identities, and randomly omitting detections to simulate false negatives. Objects with $0.7 < p_y < 0.9$ are occluded and do not generate detections. For clarity, we limit the scenario to at most two objects in the scene at a time.

We train our multi-agent policy as described in Section 3.4, obtaining detections and ground truth tracks from the generative process described above until training converges. We evaluate the policy against the rule-based SORT algorithm using the same Kalman filter parameters. Figure 2 illustrates three challenges present in the simulated scenario that the SORT algorithm is unable to address. First, the long term occlusion of objects causes SORT to terminate object tracks that enter the occlusion area and to spawn new object tracks once they exit. Under the SORT policy, object tracks that have not been associated with a detection for a fixed number of frames (in this case, three) are terminated. Second, the constant velocity motion model fails to account for the sharp turning behavior of the objects. SORT fragments their tracks under the assumption that a new object has entered the scene. Finally, objects commonly travel close to each other, leading to frequent data association failures. It is common for SORT to switch the identities of objects despite no tracks ever crossing in the ground truth.

As a rule-based algorithm, SORT does not learn information about the structure of the environment. Although the Kalman filter parameters or the algorithm's thresholds can be manually tuned, adjustments are unlikely to be optimal in every scenario. For example, in our simulated environment, it is advantageous to rely on the constant-velocity motion model for objects traveling straight, but to rely more heavily on detections during turns. In contrast, our algorithm can exploit the structure of the tracking scenario to overcome issues present in SORT's rule-based approach, as seen in Figure 2. Since each of these issues results in a lower MOTA, our training procedure's reward incentivizes policies that avoid them. Figure 3 shows the trained policy network's response to the three challenges in the scenario in Figure 2.

The leftmost scenario in Figure 3 illustrates improved robustness through periods of occlusion. While SORT terminates all object tracks entering the occluded area because there have been no data associations for too many timesteps, our trained policy learns to propagate object tracks through this area using the Kalman filter predictions until detections are visible again. In the scenario, both agents begin by taking actions a_1 and a_4. Since the agents start in the idle state, no object tracks are created. Once the object enters the scene, the agent associated to the object responds to incoming detections by increasing the probability of a_3, updating the Kalman filter with the detection every timestep. Once the object enters the occluded area, the probability of action a_4 peaks. The position of the object is estimated during occlusion by using the Kalman filter's predictions, and the object's identity is preserved. The probability of a_3 increases again under the policy once the object reappears. Finally, once the object exits the scene, the probability of a_1 peaks for a single frame, terminating the track. No detections are received by the agent during the occlusion and once the object leaves the scene, but the policy behaves differently depending on the track's current position. The probability of a_4 increases in the occlusion area, but the probability of a_1 increases once the object leaves the scene, as desired.

The second scenario in Figure 3 shows MARLMOT's response against inaccurate motion models. The constant velocity motion model cannot account for the sharp path of the turning object, resulting in SORT incorrectly concluding that a new object traveling rightwards entered the scene. Although our algorithm uses the same motion model, it is able to exploit structure in the scenario to correctly predict the object's path. Since objects never spawn in the center of the scene, the policy network learns to maintain the identity of the object through the turn, rather than conclude that it vanished midway through. Under the agent's policy, the probability of a_2 increases once the object begins to turn, resetting the Kalman filter to the detection position and maintaining the identity of the agent through the turn. Once the turn is over, the probability of a_2 decays and the agent resumes a_3 to update its Kalman filter as usual, until the agent terminates the track once the object leaves by taking action a_1.

The rightmost scenario in Figure 3 illustrates the identity swap caused by a failure in the data association step in Figure 2. In this scenario, two objects are traveling very close to each other, resulting in an increased likelihood of an identity switch. Since the learned policy is shared across all agents, cooperative decision making is not possible, so this is a challenging situation for our algorithm. The learned policy partially overcomes this obstacle by increasing the probability of a_4. By probabilistically ignoring the

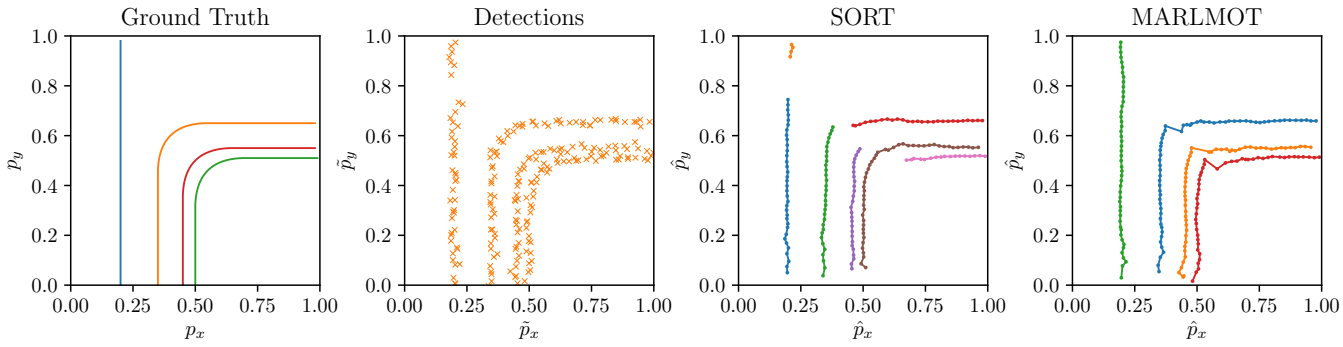

Figure 2: An instance of the synthetic multi-object tracking scenario. Object identities are distinguished by color.

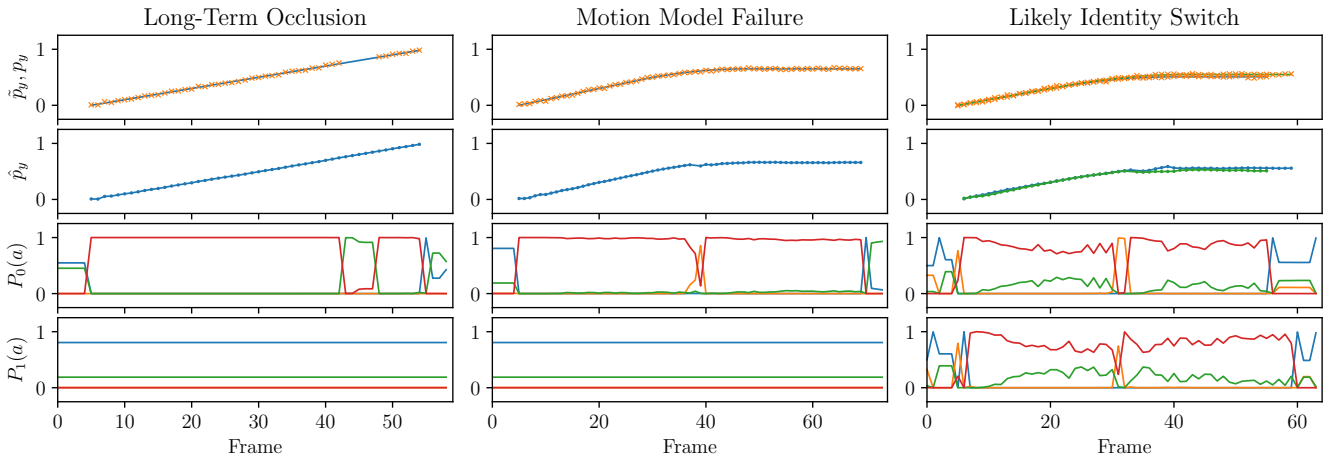

Figure 3: A trained policy's response to three tracking challenges present in the scenario from Figure 2. We plot (top to bottom) the ground truth paths and simulated detections, our algorithm's generated tracks, the first agent's probability of taking each action, and the second agent's probability of taking each action. Actions are colored as follows: a_1 (blue), a_2 (orange), a_3 (red), and a_4 (green). The probability of a_5 was 0 for all t.

associated detection and using the Kalman filter's prediction instead, the policy reduces the chance of updating the filter with an incorrect detection and swapping the identities of the two objects. This option is afforded by the simulated scenario's structure: objects travel straight at a constant velocity, so updating the filter every timestep is unnecessary and increases the chances of an identity switch. Once objects approach the turn, a_2 increases again to maintain the objects' identities through the turn, as explained in the previous paragraph.

In the simulated scenario, the agent never takes action a_5. This action places tracked objects in the occluded state, such that the output tracks are hidden from the downstream system. However, in our simulated scenario, ground truth objects that are occluded are considered when computing MOTA. In other words, a well-performing algorithm should report the hypothesized positions of occluded objects, so action a_5 is not beneficial. In contrast, MOTChallenge considers only visible objects when computing MOTA. In that case, action a_5 is necessary to maintain object identities through occlusions while not reporting them as visible.

We designed the simulated scenario as a generative model so that the tracking decisions of MARLMOT could be easily interpreted and reproduced. In practice, more general tracking algorithms such as MHT may be better choices if knowledge about the underlying dynamics of the generative model is available. Unlike MARLMOT, traditional appearance-free tracking algorithms typically require a-priori knowledge about the environment in the form of a generative model or an explicit prior over the object birth, death, and noise processes. However, in many cases, these dynamics are unavailable or hard to approximate. While access to the ground truth during the MARLMOT training procedure constitutes a form of a-priori knowledge, the structure in the environment is automatically learned. Our results on the MOTChallenge in Section 4.2 showcase the advantages of our approach in cases where the structure in the environment is not obviously exploitable.

4.2 Pedestrian Tracking in Pixel Coordinates

To quantitatively compare the performance of our algorithm to existing work, we focus on the 2015 MOTChallenge dataset. We

Table 2: Quantitative tracker results on the 2015 MOTChallenge training set for vision-free algorithms using public detections. Metrics are (in order): MOTA and MOTP [2]; false positives, false negatives, ID switches and fragmented tracks; mostly-tracked and mostly-lost tracks [17]; and recall and precision. SORT-Public refers to the results of the publicly-available implementation of the SORT algorithm when run on the public detections, while remaining results are quoted from [19]. Algorithms that use offline post-processing are denoted with an asterisk (*), and algorithms that use the Hungarian algorithm are labeled with HA. Higher values for metrics labeled ↑ are better, while lower values for metrics labeled ↓ are better.

Model	MOTA ↑	MOTP ↑	FP ↓	FN ↓	IDs ↓	FM ↓	MT ↑	ML ↓	Rcll ↑	Prcn ↑
Kalman-HA	19.2	69.9	3031	28520	685	837	32	334	28.5	79.0
Kalman-HA2*	22.4	69.4	**2245**	28626	**105**	**342**	39	354	28.3	**83.4**
JPDA-m*	23.5	69.0	2728	27707	109	380	38	348	30.6	81.7
RNN-HA	24.0	68.7	4984	24832	518	963	50	267	37.8	75.2
RNN-LSTM	22.3	69.0	5327	25094	572	983	50	260	37.1	73.5
SORT-Public	26.0	**72.5**	6767	21988	780	1174	64	237	**44.9**	72.6
MARLMOT (Train on 1)	24.0	**72.5**	5631	23923	784	974	60	265	40.1	73.9
MARLMOT (Train on all)	**27.7**	**72.5**	6092	**21976**	767	1164	**75**	**236**	**44.9**	74.6

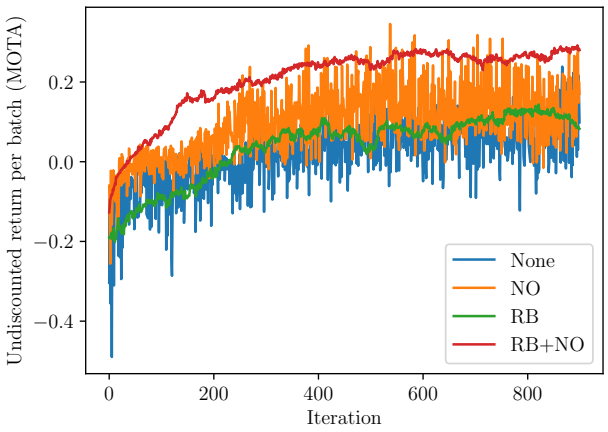

Figure 4: Average joint return (MOTA) while training on the 2015 MOTChallenge benchmark. NO refers to using normalized observations, and RB refers to using representative batches.

choose this dataset due to the unavailability of appearance-free tracking datasets, and due to several comparable works in appearance-free tracking using this dataset as well. We avoid appearance features to focus on the tracking aspect of tracking-by-detection. Previous work has shown that optimizing vision features can have a great effect on tracking performance, making it difficult to compare tracking algorithms. For example, when paired with a good detection algorithm, SORT performs near the state-of-the-art for vision-based trackers, despite being much simpler than the MDP-based tracker from Xiang et al [3, 29]. In practice, enhancing the vision component of a tracking-by-detection algorithm is frequently the easiest way to improve performance.

Since SORT is the state-of-the-art on this dataset for online, appearance-free tracking, we use the same Kalman filter parameters used in their publicly-available implementation to provide a fair comparison. Figure 4 shows the average return obtained by all tracking agents (equivalently, MOTA) for the first 900 iterations of training. Normalizing observations improved performance, and using batches representative of all possible scenarios stabilized training. Since we are able to optimize MOTA directly, the eventual performance of the algorithm during training is more interpretable than when using proxy training metrics.

Table 2 shows our results on this dataset compared to other published approaches according to the official evaluation procedure. On most metrics, we outperform all other published online, appearance-free algorithms that use publicly-available detections. Despite only using MOTA as the environment's reward, our algorithm performs well on other metrics such as MOTP [2], mostly-tracked (MT), and mostly-lost (ML) [17]. We speculate that this is due to MOTA being a good global tracking metric, as claimed by Bernardin and Stiefelhagen. Our algorithm tends to report more false positives than other approaches; however, it is more robust to false negatives, illustrating a necessary trade-off in the presence of detection noise. MARLMOT is also prone to fragmenting tracks, which may be a consequence of the probabilistic nature of the tracking policy.

Even when training on only one of the 11 scenarios in the dataset, we are able to outperform all supervised RNN approaches from Milan et al. [19], demonstrating the ability of our algorithm to generalize to unseen scenarios. This generalization is only possible by the use of observation normalization as described in Section 3.4, given that scenarios in MOTChallenge vary widely in their frame size and camera positioning. We found that only by normalizing observations and training with representative batches was it possible to outperform SORT. Training proceeds in a much more stable fashion if observations are similar across scenarios and batches are representative of all possible scenarios. In most applications, including autonomous driving, tracking is performed using the same set of sensors in the same configuration, so this is usually not as much of an issue. Although the version of MARLMOT trained on only one scenario does not outperform SORT, the parameters of the SORT algorithm were manually tuned on the entire training data. In practice, the performance of SORT is highly dependent on the choice of parameters. In this regard, MARLMOT provides an advantage by defining a training algorithm to adjust the algorithm parameters automatically.

In our experiments, the implementation of our algorithm runs with 10 simultaneous agents at a frequency of 240 Hz on a laptop CPU, demonstrating that it is possible to use for autonomous driving or real-time video purposes. Since the observations for all agents are forwarded through the same network and can be provided simultaneously in a single batch, determining each agent's action every timestep is efficient. Data association is also efficient thanks to the Hungarian algorithm, which has a cubic runtime. Furthermore, the policy network is not especially deep or complex. We expect that optimizing the algorithm or using a GPU alongside a deep learning library would further improve runtime performance.

5 CONCLUSION

We presented a multi-agent reinforcement learning formulation of multi-object tracking, where agents leverage sensor and motion models to filter noise but learn a policy from data to start, propagate, and end object tracks. Our algorithm outperforms state-of-the-art, rule-based approaches using similar ideas as well as supervised approaches involving RNNs. Although algorithms that use appearance models of objects outperform our multi-agent formulation on the MOTChallenge dataset, in tracking applications not involving video it may not be possible to use them. In these cases, a hybrid model-based/supervised approach such as ours may perform well, especially if there is a known tracking metric to optimize and tracking scenarios are similar to each other.

Future work could explore recurrent policies. In particular, it is likely that the optimal decision for an agent depends on the history of observations rather than only the current observation. In other words, the underlying dynamics are not truly Markovian. While we include information about past states and actions in our observations, it is likely that additional information about the past would help agents make better decisions. Using an RNN as the tracking policy may help learn these longer-term dependencies between observations and optimal actions [12, 28]. Experimenting with more sophisticated motion filters such as the extended Kalman filter or a supervised filter is also likely to improve performance. As another avenue of future work, incorporating visual information into the agent observations would allow comparisons against high-performing object trackers in videos. Finally, exploring supervised approaches to data association such as those presented by Milan et al. [19] may allow for end-to-end multi-object tracking.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is sponsored by SAIC Innovation Center, a subsidiary of SAIC Motor.

REFERENCES

[1] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. 2011. Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 9 (Sept 2011), 1806–1819.

[2] Keni Bernardin and Rainer Stiefelhagen. 2008. Evaluating multiple object tracking performance: the CLEAR MOT metrics. *EURASIP Journal on Image and Video Processing* 2008, 1 (2008), 1–10.

[3] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. 2016. Simple online and realtime tracking. In *IEEE International Conference on Image Processing (ICIP)*.

[4] Samuel S. Blackman. 2004. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine* 19, 1 (2004), 5–18.

[5] Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2006. Multi-agent reinforcement learning: A survey. *IEEE International Conference on Control, Automation, Robotics and Vision* 527 (2006), 1–6.

[6] Asad A. Butt and Robert T. Collins. 2013. Multi-target tracking by Lagrangian relaxation to min-cost network flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[7] Wongun Choi. 2015. Near-online multi-target tracking with aggregated local flow descriptor. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[8] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. 2016. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning (ICML)*.

[9] Thomas Fortmann, Yaakov Bar-Shalom, and Molly Scheffe. 1983. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering* 8, 3 (1983), 173–184.

[10] Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. 2004. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research* 5, Nov (2004), 1471–1530.

[11] Jayesh K. Gupta, Maxim Egorov, and Mykel J. Kochenderfer. 2017. Cooperative multi-agent control using deep reinforcement learning. In *Adaptive Learning Agents Workshop*.

[12] Matthew Hausknecht and Peter Stone. 2015. Deep recurrent Q-Learning for partially observable MDPs. In *AAAI Fall Symposium Series*.

[13] Rudolph Emil Kalman. 1960. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering* 82, 1 (1960), 35–45.

[14] Suna Kim, Suha Kwak, Jan Feyereisl, and Bohyung Han. 2012. Online multi-target tracking by large margin structured learning. In *Asian Conference on Computer Vision*.

[15] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly* 2, 1-2 (1955), 83–97.

[16] Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. 2015. MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking. *arXiv:1504.01942 [cs]* (April 2015). http://arxiv.org/abs/1504.01942 arXiv: 1504.01942.

[17] Yuan Li, Chang Huang, and Ram Nevatia. 2009. Learning to associate: Hybrid-boosted multi-target tracker for crowded scene. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[18] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[19] Anton Milan, S. Hamid Rezatofighi, Anthony Dick, Ian Reid, and Konrad Schindler. 2017. Online multi-target tracking using recurrent neural networks. In *AAAI Conference on Artificial Intelligence*.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[21] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *International Conference on Machine Learning (ICML)*.

[22] Andrew Y Ng and Stuart J Russell. 2000. Algorithms for Inverse Reinforcement Learning. In *International Conference on Machine Learning (ICML)*.

[23] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*.

[24] Xuan Song, Jinshi Cui, Hongbin Zha, and Huijing Zhao. 2008. Vision-based multiple interacting targets tracking via on-line supervised learning. In *European Conference on Computer Vision (ECCV)*.

[25] Ming Tan. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents.. In *International Conference on Machine Learning (ICML)*.

[26] Martin Ulmke, Ozgur Erdinc, and Peter Willett. 2007. Gaussian mixture cardinalized PHD filter for ground moving target tracking. In *IEEE International Conference on Information Fusion*.

[27] Ba-Ngu Vo and Wing-Kin Ma. 2006. The Gaussian mixture probability hypothesis density filter. *IEEE Transactions on Signal Processing* 54, 11 (2006), 4091–4104.

[28] Daan Wierstra, Alexander Foerster, Jan Peters, and Juergen Schmidhuber. 2007. Solving deep memory POMDPs with recurrent policy gradients. In *International Conference on Artificial Neural Networks*.

[29] Yu Xiang, Alexandre Alahi, and Silvio Savarese. 2015. Learning to track: Online multi-object tracking by decision making. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[30] Sangdoo Yun, Jongwon Choi, Youngjoon Yoo, Kimin Yun, and Jin Young Choi. 2017. Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[31] Li Zhang, Yuan Li, and Ramakant Nevatia. 2008. Global data association for multi-object tracking using network flows. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.