# Collaborative Deep Reinforcement Learning for Multi-Object Tracking

Liangliang Ren[1], Jiwen Lu[1⋆], Zifeng Wang[1], Qi Tian[2,3],Jie Zhou[1]

[1]Tsinghua University, Beijing China; [2]Huawei Noah'S Ark Lab; [3] UNiversity of Texas at San Antonio
renll16@mails.tsinghua.edu.cn, lujiwen@tsinghua.edu.cn,
wangzf14@mails.tsinghua.edu.cn, qi.tian@utsa.edu jzhou@tsinghua.edu.cn

**Abstract.** In this paper, we propose a collaborative deep reinforcement learning (C-DRL) method for multi-object tracking. Most existing multi-object tracking methods employ the tracking-by-detection strategy which first detects objects in each frame and then associates them across different frames. However, the performance of these methods rely heavily on the detection results, which are usually unsatisfied in many real applications, especially in crowded scenes. To address this, we develop a deep prediction-decision network in our C-DRL, which simultaneously detects and predicts objects under a unified network via deep reinforcement learning. Specifically, we consider each object as an agent and track it via the prediction network, and seek the optimal tracked results by exploiting the collaborative interactions of different agents and environments via the decision network.Experimental results on the challenging MOT15 and MOT16 benchmarks are presented to show the effectiveness of our approach.

**Keywords:** Object tracking, multi-object, deep reinforcement learning

## 1 Introduction

Multi-object tracking (MOT) has attracted increasing interests in computer vision over the past few years, which has various practical applications in surveillance, human computer interface, robotics and advanced driving assistant systems. The goal of MOT is to estimate the trajectories of different objects and track those objects across the video. While a variety of MOT methods have been proposed in recent years [7,8,14,27,34,36,40,45–47,52], it remains a challenging problem to track multiple objects in many unconstrained environments, especially in crowded scenes. This is because occlusions between different objects and large intra-class variations usually occur in such scenarios.

Existing MOT approaches can be mainly divided into two categories, 1) offline (batch or semi-batch) [7,27,40,45,46,52] and 2) online [8,14,34,36,47]. The key idea of offline methods is to group detections into short trajectory segments

⋆ Corresponding author.

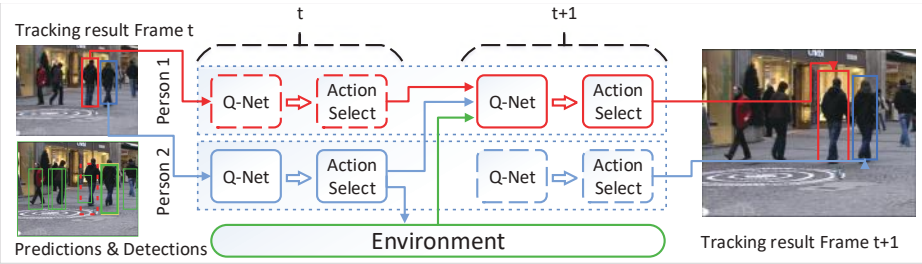


Fig. 1: The key idea of our proposed C-DRL method for multi-object tracking. Given a video and the detection results of different objects for the $t$th frame, we model each object as an agent and predict the location of each object for the following frames, where we seek the optimal tracked results by considering the interactions of different agents and environment via a collaborative deep reinforcement learning method. Lastly, we take actions to update agents at frame $t+1$ according to the outputs of the decision network

or tracklets, and then use more reliable features to connect those tracklets to full trajectories. Representative off-line methods use min-cost network flow [5, 54], energy minimization [28] or generalized minimum clique graphs [52] to address the data association problem. Online MOT methods estimate the object trajectories with the detections of the current and past frames, which can be applied to real-time applications such as advanced driving assistant systems or robotics. Conventional online methods usually employ Kalman filtering [19], Particle filtering [32] or Markov decisions [47]. However, the tracking accuracy of these methods is sensitive to the occlusions and noisy detection results, such as missing detections, false detections and the non-accurate bounding boxes, which make these methods difficult to be applied for videos of crowded scenes.

In this paper, we propose a collaborative deep reinforcement learning (C-DRL) method for multi-object tracking. Fig. 1 illustrates the basic idea of our proposed approach. Given a video and the detection results of different objects for the $t$th frame, we model each object as an agent and predict the locations of objects of the following frames by using the history trajectories and the appearance information of the $(t+1)$th image frame. We exploit the collaborative interaction of each agent between the neighboring agents and the environment, and make decisions for each agent to update, track or delete the target object via a decision network, where the influence of occlusions between objects and noisy detection results can be well alleviated by maximizing their shared utility. Experimental results on the challenging MOT15 and MOT16 benchmarks are presented to demonstrate the efficiency of our approach.

## 2 Related Work

**Multi-Object Tracking:** Most existing MOT methods can be categorized into two classes: 1) offline [7, 27, 40, 45, 46, 52] and 2) online [8, 14, 34, 36, 47]. Meth-

ods in the first class group all detection results into short trajectory segments or tracklets, and connect those tracklets into full trajectories. For example, Zamir *et al.* [52] associated all detection results which incorporate both the appearance and motion information in a global manner by using generalized minimum clique graphs. Tang *et al.* [40] introduced a graph-based method that links and clusters objects hypotheses over time by solving a subgraph multicut problem. Maksai *et al.* [27] proposed an approach to track multiple objects with non-Markovian behavioral constraints. Methods in the second class estimate object trajectories with the detection results of the current and past frames. For example, Yang *et al.* [48, 49] introduced an online learned CRF model by solving an energy minimization problem with nonlinear motion patterns and robust appearance constraints for multi-object tracking. Xiang *et al.* [47] formulated MOT as a decision-making problem via a Markov decision process. Choi *et al.* [7] presented an aggregated local flow descriptor to accurately measure the affinity between different detection results. Hong *et al.* [14] proposed a data-association method to exploit structural motion constraints in the presence of large camera motion. Sadeghian *et al.* [34] encoded dependencies across multiple cues over a temporal window and learned multi-cue representation to compute the similarity scores in a tracking framework. To overcome the influence of noisy detection, several methods have also been proposed. For example, Shu *et al.* [36] introduced a part-based representation under the tracking-by-detection framework to handle partial occlusions. Chu *et al.* [8] focused on learning a robust appearance model for each target by using a single object tracker. To address the occlusion and noisy detection problem, our approach uses a prediction-decision network to make decisions for online multi-object tracking.

**Deep Reinforcement Learning:** Deep reinforcement learning has gained significant successes in various vision applications in recent years, such as object detection [25], face recognition [33], image super-resolution [6] and object search [20]. Current deep reinforcement learning methods can be divided into two classes: deep Q learning [12, 29, 30, 42] and policy gradient [1, 37, 50]. For the first class, Q-values are fitted to capture the expected return for taking a particular action at a particular state. For example, Cao *et al.* [6] proposed an attention-aware face hallucination framework with deep reinforcement learning to sequentially discover attended patches and perform facial part enhancement by fully exploiting the global interdependency of the image. Rao *et al.* [33] proposed a attention-aware deep reinforcement learning method to select key frames for video face recognition. Kong *et al.* [20] presented a collaborative deep reinforcement learning method to localize objects jointly in a few iterations. For the second class, the distribution of policies is represented explicitly and the policy is increased by updating the parameters in the gradient direction. Liu *et al.* [26] applied a policy gradient method to optimize a variety of captioning metrics. Yu *et al.* [50] proposed a sequence generative adversarial nets with policy gradient. More recently, deep reinforcement learning [15, 16, 39, 51, 53] has also been employed in visual tracking. For example, Yun *et al.* [51] proposed an action-decision network to generate actions to seek the locations and sizes

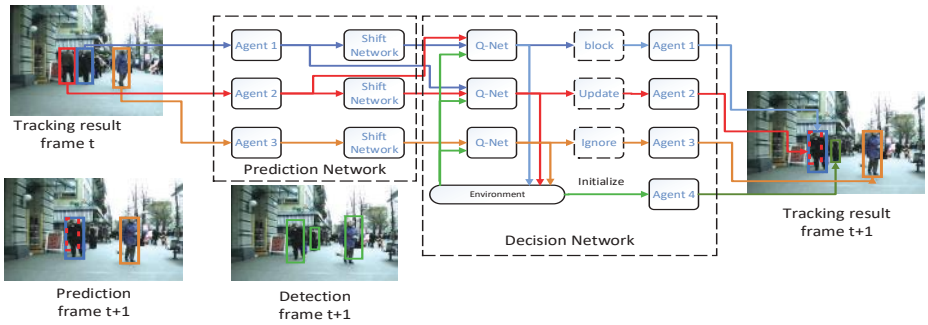


Fig. 2: The framework of the proposed C-DRL for multi-object tracking. In this figure, there are three objects at frame $t$. We first predict the locations of these three objects at frame $t+1$. Then we use a decision network to combine the prediction and detection results and make decisions for each agent to maximize their shared utility. For example, Agent 2 is blocked by its neighborhood (Agent 1). Agent 1 updates itself by using the nearest detection result, and Agent 3 ignores the noisy detection. We initialize Agent 4 by using the remaining detection result in the environment. Lastly we use the locations of each agent as the tracking results at frame $t+1$

of the objects in a new coming frame. Supancic *et al.* [39] proposed a decision policy tracker by using reinforcement learning to decide where to look in the upcoming frames, and when to re-initialize and update its appearance model for the tracked object. However, these methods can not be applied to multi-object tracking directly since they ignore the communication between different objects. In this work, we propose a collaborative deep reinforcement learning method to exploit the interactions of different objects for multi-object tracking.

## 3 Approach

Fig. 2 shows the framework of the proposed C-DRL method for multi-object tracking, which contains two parts, 1) a prediction network and 2) a decision network. Given a video and the detection results of different objects at frame $t$, we model each object as an agent and predict the locations of objects for the following frames, and seek the optimal tracked results by considering the interactions of different agents and environment via the decision network. Lastly, we take actions to update, delete or initialize agents at frame $t+1$ according to decisions. In the following subsections, we will detail the prediction network and the decision network, respectively.

### 3.1 Learning of The Prediction Network

Given initial locations of objects, the prediction network aims to learn the movement of objects to predict the locations of the target object. As shown in Fig. 3,

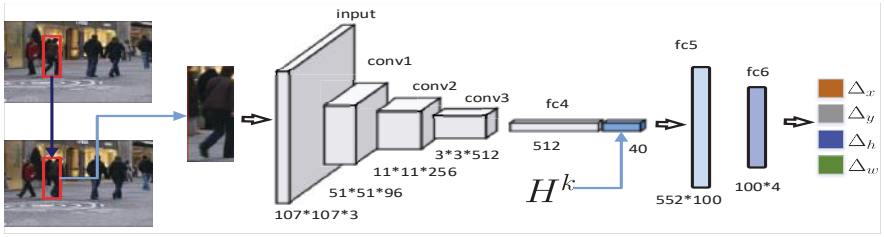


Fig. 3: The framework of the prediction network. The prediction network learns the movement of the target object given an initial location of the object, which contains three convolutional layers and three fully connected layers

the inputs to the prediction network are the raw image cropped by the initial bounding box of the next frame and history trajectories. We randomly sample bounding boxes $b \in B_{i,t}$ around the location of the object $b^*_{i,t}$ in each frame in the training videos as the training set to learn the prediction network. The prediction network takes the $(t+1)$th frame cropped by the initial location $b$ and the history trajectories $H$ of the last $K$ frames for position prediction, where $K$ is set as 10 in our work. We formulate location prediction as the following regression problem:

$$\arg\max_{\phi} J(\phi) = \sum_{i,t} \sum_{b \in B_{i,t}} g(b^*_{i,t+1}, b + \phi(I_t, b, H_t)), \tag{1}$$

where $J$ is the optimization objective function at the top layer of the prediction network, $\phi$ is the parameter set of the network, $b^*_{i,t+1}$ is the ground truth of the object $p_i$ at frame $t+1$, and $g(\cdot)$ denotes the intersection-over-union (IoU) of two bounding boxes.

$$g(b_i, b_j) = \frac{b_i \cap b_j}{b_i \cup b_j}. \tag{2}$$

### 3.2 Collaborative Deep Reinforcement Learning

As shown in Fig. 2, the decision network is a collaborative system which contains multiple agents and the environment. Each agent takes actions with the information from itself, the neighborhoods and the environment, where the interactions between agents and the environment are exploited by maximizing their shared utility. To make better use of such contextual information, we formulate multi-object tracking as a collaborative optimization problem.

We consider each object as an agent. Each agent $p$ contains the trajectory $\{(x_0, y_0), (x_1, y_1), \cdots, (x_t, y_t)\}$, the appearance feature $f$, and the current location $\{x, y, w, h\}$. Hence, the distance between two objects $p_i$ and $p_j$ can be computed as follows:

$$d(p_i, p_j) = \alpha(1 - g(p_i, p_j)) + (1 - \frac{f_i^T f_j}{\|f_i\|_2 \|f_j\|_2}), \tag{3}$$

where $g(p_i, p_j)$ is the IoU of two bounding boxes, and $\alpha \geq 0$.

The environment contains the object detection results: $\mathcal{P}_t^* = \{p_1^*, p_2^*, \cdots, p_{N_t}^*\}$. The distance between the object $p_i$ and the detection results can be computed as follows:

$$d(p_i, p_j^*) = \alpha(1 - g(p_i, p_j^*)) + (1 - \frac{f_i^T f_j}{\|f_i\|_2 \|f_j^*\|_2}). \tag{4}$$

Let $I_t$ be the $t$th frame of the selected video, which contains $n_t$ objects, $\mathcal{P}_t = \{p_1, p_2, \cdots, p_{n_t}\}$. The state at frame $t$, $s_t = \{\mathcal{P}_t, \mathcal{P}_t^*\}$ contains the current agents and the detection results. For the object $p_i$, we first use a prediction network to generate the position at frame $t+1$. Then, we select the nearest neighborhood $p_j \in \mathcal{P}_t - \{p_i\}$ and the nearest detection result $p_k^* \in \mathcal{P}_{t+1}^*$. Subsequently, we take these three images as the input to the decision network if $d(p_j, p_i) < \tau$ and $d(p_k^*, p_i) < \tau$. If $d(p_j, p_i) \geq \tau$ or $d(p_k^*, p_i) < \tau$, we replace it with a zero image.

The object has two different status in each frame: *visible* or *invisible*. If the object is *visible*, we update the agent with the prediction or the detection result. If the detection result is reliable, we use both the detection result and the prediction result. If the detection results is not reliable, we only use the prediction result. If the object is *invisible*, the object may be blocked by other objects or disappears. If the object is blocked, we keep the appearance feature and only use the movement model to predict the location of the object for the next frame. If the object disappears, we delete the object directly. Hence, for each agent, the action set is defined as $\mathcal{A} = \{update, ignore, block, delete\}$.

For the action *update*, we use both the prediction and detection results to update the position of $p_i$ and the appearance feature,described as below:

$$f_i = (1 - \rho) f_i + \rho f_i^*, \tag{5}$$

where $\rho$ is the learning rate of appearance features.

We delete the detection results which are used to update agents features. For remaining detection results in the environment, we initialize an agent for each remaining result. For a false detection, the agent is also initialized, but the reward of the action $\{update, ignore, block\}$ is set to -1 while the reward of the action *delete* is set to 1. Then, the agent is deleted in the next iteration.

For the action *ignore*, the detection result is not reliable or missing, while the prediction result is more reliable. We use the prediction result to update the position of $p_i$.

For the action *block*, we keep the feature of $p_i$ as the object has been blocked by other objects, and the location is updated according to the prediction result.

For the action *delete*, the object disappears, and we delete the object $p_i$ directly.

Therefore, the rewards $r_{i,t}^*$ of each action contains two terms: $r_{i,t}$ and $r_{j,t+1}$, where $r_{i,t}$ describes its own state in the next frame, and $r_{j,t+1}$ refers to its nearest neighborhood state in the next frame. The final rewards can be computed as follows:

$$r_{i,t}^* = r_{i,t} + \beta r_{j,t+1}, \tag{6}$$

where $\beta \geq 0$ is the balance parameter.

The $r_{i,t}$ of actions $\{update,\ ignore,\ block\}$ is defined by the IoU of the prediction location with the ground truth in the next frame. If the value of IoU is too small or the object disappears, $r_{i,t}$ is set to $-1$.

$$r_{i,t} = \begin{cases} 1 & \text{if } IoU \geq 0.7 \\ 0 & \text{if } 0.5 \leq IoU \leq 0.7 \\ -1 & else \end{cases}. \tag{7}$$

The $r_{i,t}$ of the action *delete* is defined by the states of objects. If the object disappears in the next frame, $r_{i,t}$ is 1, and otherwise -1.

$$r_{\text{delete}} = \begin{cases} 1 & \text{if object disappeared} \\ -1 & \text{else} \end{cases}. \tag{8}$$

We compute the Q value of $\{s_{i,t}, a_{i,t}\}$ as follows:

$$Q(s_{i,t}, a_{i,t}) = r^*_{i,t} + \gamma r^*_{i,t+1} + \gamma^2 r^*_{i,t+2} + \cdots, \tag{9}$$

where $\gamma$ is the decaying parameter.

The optimization problem of the decision network is formulated as follows:

$$\arg\max_{\theta} L(\theta) = \mathbb{E}_{s,a} \log(\pi(a|s,\theta)) Q(s,a), \tag{10}$$

where $\theta$ is the parameter set of the decision network, and the policy gradient can be computed as follows:

$$\begin{aligned} \Delta_\theta L(\theta) &= \mathbb{E}_{s,a} \Delta_\theta \log(\pi(a|s,\theta)) Q(s,a) \\ &= \mathbb{E}_{s,a} \frac{Q(s,a)}{\pi(a|s,\theta)} \Delta_\theta \pi(a|s,\theta). \end{aligned} \tag{11}$$

The gradient shows that we can increase the probability of actions which have positive Q values, and decrease the probability of actions which have negative Q values. However, in some easy scenes, the Q values of most actions are positive, while the Q values of all actions are negative in some challenging cases or at the beginning of the training stage. Hence, the policy gradient network is difficult to converge. Therefore, we use the advantage value of actions to replace the Q value, where we first compute the value of the state $s$ as follows:

$$V(s) = \frac{\sum_a p(a|s) Q(s,a)}{\sum_a p(a|s)}. \tag{12}$$

Then, the advantage value is computed as follows:

$$A(s,a) = Q(s,a) - V(s). \tag{13}$$

The final formulation of the policy gradient is defined as:

$$L(\theta) = \mathbb{E}_{s,a} \log(\pi(a|s,\theta)) A(s,a). \tag{14}$$

**Algorithm 1 :** Learning the Decision Network

**Input:** Training set: $\mathbf{V} = \{V_i\}$, and convergence error $\epsilon_1$ maximal iterations number $M$.
**Output:** $\theta$

Initialize $\theta$;
**for all** $l = 1, 2, \ldots, M$ **do**
  Randomly select a video ($V$);
  Initialize agents set $\mathcal{P}$ using the detection results in 1-st frame
  **for all** $t = 2, 3, \ldots, I_t$ **do**
    **for all** $p \in \mathcal{P}$ **do**
      Take actions according to the output of decision networks;
      Update or delete $p$ according to actions;
    **end for**
    Add $p_i^* \in \mathcal{P}^*$
  **end for**
  Calculate $L_t$ according to (10);
  Calculate advantage value $A(s, a)$ for each agent;
  Update policy network $\theta$ according to (15) ;
  **if** $l > 1$ and $|L_t - L_{t-1}| < \epsilon_1$ **then**
    Go to **return**
  **end if**
**end for**
**return** $\theta$

The parameter $\theta$ can be updated as follows:

$$\begin{aligned}\theta &= \theta + \rho \frac{L(\theta)}{\partial \theta} \\ &= \theta + \rho \mathbb{E}_{s,a} \frac{A(s,a)}{\pi(a|s,\theta)} \frac{\partial \pi(a|s,\theta)}{\partial \theta}.\end{aligned} \tag{15}$$

**Algorithm 1** summarizes the detailed learning procedure of our decision network.

## 4 Experiments

### 4.1 Datasets

**MOT15**: It contains 11 training sequences and 11 testing sequences. For each testing sequence, we have a training set of similar conditions so that we can learn our model parameters accordingly. The most challenging sequence in MOT15 is the AVG-TownCentre in the testing sequences because its frame rate is very low, and there is no corresponding training sequence.
**MOT16**: It contains 7 training sequences and 7 testing sequences. Generally, MOT16 is more challenging than MOT15 because the ground truth annotations are more accurate (some hard examples are taken into account), the background

settings are more complex (e.g. with moving cars or captured with a fast moving camera), and the pedestrians are more crowded so that the occlusion possibility is increased. The camera motion, camera angle and the imaging conditions vary largely among different sequences in both datasets.

### 4.2 Evaluation Metrics

We adopted the widely used CLEAR MOT metrics [4] including multiple object tracking precision (MOTP) and multiple object tracking accuracy (MOTA) which combine false positives (FP), false negatives (FN) and the identity switches (ID Sw) to evaluate the effectiveness of different MOT methods. We also used the metrics defined in [24] which contains the percentage of mostly tracked targets (MT, the ratio of ground-truth trajectories that is covered by a track hypothesis for at least 80% of their respective life span), the percentage of mostly lost targets (ML, the ratio of ground-truth trajectories that are covered by a track hypothesis for at most 20% of their respective life span), and the time of a trajectory is fragmented (Frag, interrupted during tracking).

### 4.3 Implementation Details

**Decision Network:** Our decision network consists of a feature extraction part and a decision-making part. We used the part of MDNet [31] which was pre-trained on ImageNet [9] to extract the feature of each object. The input size of the network is 3×107×107. It consists of three consecutive convolution layer (7×7×96, 5×5×256, 3×3×512) and max pooling layer combos (including batch normalization layers), and finally a fully connected layer to flatten the feature to a column vector $D$ of size 512×1. We then calculate the position feature $P$ (of size 4×1) and concatenate $D$ and $P$ to a mixed feature vector $W$. Having predicted $agent_1$ with feature $W_1$, the agent which is closest to $agent_1$ in the predicted model is called $agent_2$ with feature $W_2$, and the counterpart $agent_1^{det}$ with feature $W_1^{det}$ for $agent_1$ in the detection of the next frame, and finally $agent_1$ in the previous frame with feature $D_1^{pre}$. Having concatenated all features, we obtain the input to the decision-making network (input size: 2060×1). The structure of the network is relatively simple, we just utilized 3 fully connected layers (with dropout when training) to reduce the dimension to 4×1, which is corresponding to these four strategies.

In order to show that our network can learn to make decisions under various scenarios, we trained the decision network on all training sequences (both from MOT15 and MOT16) and then evaluate it on all the testing sequences without further processing. Here we trained the decision network on the training sequence of MOT15 and MOT16 for 10 epochs (1 epoch loops through all training sets including both MOT15 and MOT16). We optimized the network with stochastic gradient descent with weight decay at the rate of 0.0005 and momentum at the rate of 0.9. We set the learning rate 0.0002 at first 5 epochs and changed it into 0.0001 for the next 5 epochs. We applied a dynamic batch size strategy, which means that we obtain each frame and feed all objects in this frame to the

network as a batch. This process best mimics the real tracking process thus is good for our network to be utilized to real tracking scenarios.

As for the reinforcement learning hyper-parameters, we firstly set balance parameter $\beta$ and discount rate parameter $\gamma$ to zero to simplify the training phase and let the decision network converge to a certain reward. Here the reward is 0.637. We then did grid search based on fine tuning the network. As shown in Fig. 4 that when $\gamma = 0.8$ and $\beta = 0.4$, we get maximized normalized reward (we normalize it to $[0, 1]$), so we set the hyper-parameters as above.

**Prediction Network:** We extracted all positive examples from the all training sequences from the datasets. In order to simulate noisy situations, we merged the information of detections and ground truth annotations, and computed the $IoU$ of the detection bounding boxes and ground truth bounding boxes. If $IoU > 0.5$, the detection is valid and we put the detection into our dataset; otherwise, we treated the detection as a false positive and discard it. Therefore, we combined the detection and ground truth information when training the shift network. Our prediction network shares the same feature extraction part with the C-DRL network. Having obtained the feature vector $D$, we concatenated it with $H^{10}(x, y, h, w)$, which is the trajectory of the past 10 frames of the target. We trained the network for 20 epochs with a batch size of 20. We selected stochastic gradient descend with a learning rate of 0.002 and weight decay at the rate of 0.0005 and the momentum at the rate of 0.95. We halved the learning rate every 5 epochs. Our tracking system was implemented under the MATLAB 2015b platform with the MatConvNet [43] toolbox.

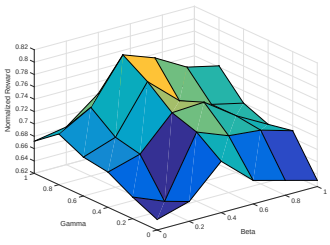


Fig. 4: The average normalized rewards versus different $\beta$ and $\gamma$ on the MOT15 training dataset

| THRESH | Rcll | FP | FN | IDs | MOTA | MOTAL |
|---|---|---|---|---|---|---|
| 1 | 83.1 | 81.7 | 6598 | 481 | 63 | 64 |
| 2 | 83.0 | 83.3 | 6673 | 440 | 65 | 66 |
| 3 | 82.8 | 83.9 | 6742 | 411 | 65 | 66 |
| 4 | 82.5 | 84.6 | 6837 | 380 | 66 | 67 |
| 5 | 82.3 | 85.2 | 6929 | 359 | 67 | 68 |
| 6 | 82.1 | 85.6 | 7003 | 348 | 67 | 68 |
| 7 | 81.8 | 86.1 | 7142 | 329 | 67 | 68 |
| 8 | 81.4 | 86.4 | 7292 | 307 | 67 | 68 |
| 9 | 81.0 | 86.7 | 7448 | 293 | 67 | 68 |

Table 1: Performance of our method under different inter-frame relation thresholds

### 4.4 Ablation Studies

We conducted ablation studies on MOT the SubCNN detection of MOT15 training set which was provided in [47].

Table 2: Ablation studies of different settings

| Method | Rcll | Prcn | GT | MT | ML | FP | FN | IDs | MOTA | MOTP | MOTAL |
|---|---|---|---|---|---|---|---|---|---|---|---|
| OB | 81.4 | 86.4 | 458 | 293 | 66 | 4995 | 7292 | 307 | 67.8 | 85.2 | 68.6 |
| DN → HA | 83.2 | 78.3 | 458 | 317 | 31 | 9042 | 6562 | 2048 | 54.9 | 84.4 | 60.1 |
| PN → VM | 83.1 | 81.7 | 458 | 317 | 31 | 7296 | 6620 | 453 | 63.3 | 84.7 | 64.4 |
| MD → HIST | 81.8 | 84.7 | 458 | 304 | 35 | 5772 | 7121 | 463 | 65.9 | 85.2 | 67.1 |

**Influences of the Inter-frame Relation:** We changed the consecutive frame information in our network to investigate how it affects the performance. Our method automatically wipes out the agents of relatively short continuous appearing time, which has been utilized in the training stage of our C-DRL network (e.g. when an agent is lost for a certain number of frames, our method gives the command to pop it out and renews the weights in that direction). We set the threshold from 1 to 9. From Table 1, we see that when more inter-frame information is utilized, more constraints to our agents can be included, so that the noisy detection results can be well eliminated in our model. We also notice that as our FP goes up, our FN falls as well, which is a trade-off between the precision and recall. Since MOTA seems to be saturated after THRESH $\geq 8$, setting THRESH to be 8 is a good choice for optimizing MOTA.

**Influences of decision network:** We set inter-frame threshold to 8 from the conclusion of previous part. Our original baseline (OB) is our full pipeline without modification. We replaced our decision network with vanilla Hungarian algorithm and fixed all other parameters (DN → HA). We find that the overall performance of the whole system falls drastically according to 2. Especially, the FP almost doubles and the IDs increases by an order of magnitude. Our decision network effectively wipes out false positives and id switches by conducting appropriate actions.

**Influences of prediction network**: We replaced our prediction network with velocity model method (PN → VM). We predict the position of each agent by using the trace of them. In other words, we model the instant velocity of agents by using their previous movement. According to our experiment result showed in Table 2, the performance gets worse as well. As the movement of pedestrians in MOT15 training set is relatively smooth and slow, there are rarely edge cases like turning or running. As a result, the performance is not bad. However, our original pipeline is still able to give more precise position prediction.

**Influences of MDNet feature**: We replaced the MDNet part of our decision and the prediction networks with simple color histogram feature (PN → VM) and then feed them to the fully-connected layers. This time, the performance downgrade is slight, which means our reinforcement learning method is robust to different feature representations. However, more delicate and informative feature is a boost.

We could easily see the advantage of our decision network and the effectiveness of prediction network. As our decision network apparently enhances the performance by a large margin, thats the core part of our whole system.

Fig. 5: Some tracking results on the MOT15 and MOT16 public detections, where the trajectory of each object has been painted from the first frame in the same color as its bounding box

Table 3: The performance of different methods on MOT15

| Mode | Method | MOTA↑ | MOTP↑ | FAF↓ | MT(%)↑ | ML(%)↓ | FP↓ | FN↓ |
|---|---|---|---|---|---|---|---|---|
| Offline | LINF1 [10] | 24.5 | 71.3 | 1.0 | 5.5 | 64.6 | 5864 | 40207 |
| | LP_SSVM [44] | 25.2 | 71.7 | 1.4 | 5.8 | 53.0 | 8369 | 36932 |
| | MHT_DAM [18] | 32.4 | 71.8 | 1.6 | 16.0 | 43.8 | 9064 | 32060 |
| | NMOT [7] | 33.7 | **71.9** | 1.3 | 12.2 | 44.0 | 7762 | 32547 |
| | QuadMOT [38] | 33.8 | 73.4 | 1.4 | 12.9 | 36.9 | 7898 | 32061 |
| | JointMC [17] | 35.6 | **71.9** | 1.8 | **23.2** | 39.3 | 10580 | **28508** |
| Online | SCEA [14] | 29.1 | 71.1 | 1.0 | 8.9 | 47.3 | 6060 | 36912 |
| | MDP [47] | 30.3 | 71.3 | 1.7 | 13.0 | 38.4 | 9717 | 32422 |
| | CDA_DDALpb [2] | 32.8 | 70.7 | **0.9** | 9.7 | 42.2 | **4983** | 35690 |
| | AMIR15 [34] | **37.6** | 71.7 | 1.4 | 15.8 | **26.8** | 7933 | 29397 |
| | Ours | 37.1 | 71.0 | 1.2 | 14.0 | 31.3 | 7036 | 30440 |

### 4.5 Evaluations on MOT15

**Comparison with State-of-the-arts:** For a fair comparison, we used the public detection results on MOT15 and MOT16. Sampled results are showed in Fig. 5. As shown in Table 3, our method outperforms most state-of-the-art trackers on MOT15 under the MOTA metric, which is one of the most important and persuasive metrics in multi-object tracking. Our method is also comparable with AMIR15 [34]. Moreover, we obtained the best FN among all online methods, which indicates that our method is able to recover detections effectively. We noticed that some methods such as LINF1 [10] can obtain relatively high performance on FP and ID Sw. However, it sacrifices lots of hard examples, which leads to a bad FN performance. Our method also outperforms all offline methods (e.g. they have access to all frames regardless of the time order so that they get far more information than an online one), which indicates that our network can well learn contextual information via the deep reinforcement learning framework.

### 4.6 Evaluations on MOT16

**Comparison with State-of-the-arts:** As shown in Table 4, our method achieved the best MOTA result among all online MOT methods and is comparable to the

Table 4: The performance of different methods on MOT16

| Mode | Method | MOTA↑ | MOTP↑ | FAF↓ | MT(%)↑ | ML(%)↓ | FP↓ | FN↓ |
|---|---|---|---|---|---|---|---|---|
| Offline | TBD [11] | 33.7 | 76.5 | 1.0 | 7.2 | 54.2 | 5804 | 112587 |
| | LTTSC-CRF [21] | 37.6 | 75.9 | 2.0 | 9.6 | 55.2 | 11969 | 101343 |
| | LINF1 [10] | 41.0 | 74.8 | 1.3 | 11.6 | 51.3 | 7896 | 99224 |
| | MHT_DAM_16 [18] | 45.8 | 76.3 | 1.1 | 16.2 | 43.2 | 6412 | 91758 |
| | NOMT [7] | 46.4 | 76.7 | 1.6 | **18.3** | 41.4 | 9753 | 87565 |
| | NLLMPa [23] | 47.6 | 78.5 | 1.0 | 17.0 | 40.4 | 5844 | 89093 |
| | LMP [41] | **48.8** | **79.0** | 1.1 | 18.2 | 40.1 | 6654 | **86245** |
| Online | OVBT [3] | 38.4 | 75.4 | 1.9 | 7.5 | 47.3 | 11517 | 99463 |
| | EAMTT_pub [35] | 38.8 | 75.1 | 1.4 | 7.9 | 49.1 | 8114 | 102452 |
| | CDA_DDALv2 [2] | 43.9 | 74.7 | 1.1 | 10.7 | 44.4 | 6450 | 95175 |
| | AMIR [34] | 47.2 | 75.8 | **0.5** | 14.0 | 41.6 | **2681** | 92856 |
| | Ours | 47.3 | 74.6 | 1.1 | 17.4 | **39.9** | 6375 | 88543 |

best offline methods such as LMP [41] and FWT [13]. In terms of MT and ML, our method also achieves the best performance among all online methods, which indicates that our method can keep track of relatively more objects than other methods under complex environments. Since the detection results of MOT16 are more accurate, our decision network and prediction network can learn more right behaviors and decrease the possibility of losing objects. Another observation is that our method obtains the best FN performance among all online methods, which is because our method recovers some missing objects that were missed in the detector via the decision network. Since the public detector does not cover all positive samples in MOT16, the rates of FN are naturally high for all methods. However, our method addresses this decently. We see that our method outperforms these offline methods by a large margin, which shows that the effectiveness of the decision network where collaborative interaction maximizing the utilization of contextual information is effectively exploited to enhance the generalization ability of our network. Also, our FP gets the second place among both online and offline methods, which means our method has strong ability to eliminate false positives exist in the detection results. In Fig. 6 (a) the top image shows the provided public detection results which contain multiple detections of same people. However, in the tracking result below, our method successfully eliminate those redundant detections.

**Failure Cases:** Fig. 6 (b) shows some failure examples of our method . For the first row, we could see that when people walk by each other, it is easy for them to switch their ids. For example the woman in white is initially in blue box, however the blue box moves to the man in blue in the next frames. For the second row, we could see that when occlusion lasts for long time, the reappeared person would be assigned with a new id (i.e. A bounding box with a new color in our picture). For instance, the man in white is initially in yellow box and he is hidden by another one in the second frame. When he reappears in the third frame, he is in a newly assigned yellow box. Our method has a relatively high ID switch and Frag (actually these two metrics are closely correlated) on both

(a) (b)

Fig. 6: (a) False positives eliminating (b) ID switch problems

the MOT15 and MOT16 datasets, which indicates that our decision network is sometimes over-cautious when there are some changes on conditions. In such scenarios, our method will assign the object a new ID label. For the memory optimization, we keep the object in our model for several frames (where we set 2 in our experiments) if it got lost at a certain frame. For some videos of high sampling rates, the object lost for relatively more frames due to occlusion and this also caused ID switch as well. However, this can be relieved by saving the feature of possibly disappeared objects in our model for longer frames and training the network with more similar sequences so that the network can better utilize dynamic information. Another reason is that when two or more objects move to each other, both their position and appearance feature are extremely similar, which poses large challenges for MOT trackers.

## 5 Conclusion

In this paper, we have proposed a collaborative deep reinforcement learning method for multi-object tracking. Specifically, we have employed a prediction-network to estimate the location of objects in the next frame, and used deep reinforcement learning to combine the prediction results and detection results and make decisions of state updates to overcome the occlusion and the missed or false detection. Experimental results on both the challenging MOT15 and MOT16 benchmarks are presented to show the effectiveness of our approach. How to apply our method to camera-network multi-object tracking seems to be an interesting future work.

## Acknowledgement

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFA0700802, in part by the National Natural Science Foundation of China under Grant 61672306, Grant U1713214, Grant 61572271, and in part by supported by NSFC under Grant No.61429201, in part to Dr. Qi Tian by ARO grant W911NF-15-1-0290 and Faculty Research Gift Awards by NEC Laboratories of America and Blippar.

## References


1. Ammar, H.B., Eaton, E., Ruvolo, P., Taylor, M.: Online multi-task learning for policy gradient methods. In: ICML. pp. 1206–1214 (2014)
2. Bae, S.H., Yoon, K.J.: Confidence-based data association and discriminative deep appearance learning for robust online multi-object tracking. TPAMI (2017)
3. Ban, Y., Ba, S., Alameda-Pineda, X., Horaud, R.: Tracking multiple persons based on a variational bayesian model. In: ECCV. pp. 52–67 (2016)
4. Bernardin, K., Stiefelhagen, R.: Evaluating multiple object tracking performance: the clear mot metrics. EURASIP **2008**(1), 246309 (2008)
5. Butt, A.A., Collins, R.T.: Multi-target tracking by lagrangian relaxation to min-cost network flow. In: CVPR. pp. 1846–1853 (2013)
6. Cao, Q., Lin, L., Shi, Y., Liang, X., Li, G.: Attention-aware face hallucination via deep reinforcement learning. In: CVPR. pp. 690–698 (2017)
7. Choi, W.: Near-online multi-target tracking with aggregated local flow descriptor. In: ICCV. pp. 3029–3037 (2015)
8. Chu, Q., Ouyang, W., Li, H., Wang, X., Liu, B., Yu, N.: Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism. In: ICCV. pp. 4836–4845 (2017)
9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. pp. 248–255 (2009)
10. Fagot-Bouquet, L., Audigier, R., Dhome, Y., Lerasle, F.: Improving multi-frame data association with sparse representations for robust near-online multi-object tracking. In: ECCV. pp. 774–790 (2016)
11. Geiger, A., Lauer, M., Wojek, C., Stiller, C., Urtasun, R.: 3d traffic scene understanding from movable platforms. TPAMI **36**(5), 1012–1025 (2014)
12. Gu, S., Lillicrap, T., Sutskever, I., Levine, S.: Continuous deep q-learning with model-based acceleration. In: ICML. pp. 2829–2838 (2016)
13. Henschel, R., Leal-Taixé, L., Cremers, D., Rosenhahn, B.: Improvements to frank-wolfe optimization for multi-detector multi-object tracking. arXiv preprint arXiv:1705.08314 (2017)
14. Hong Yoon, J., Lee, C.R., Yang, M.H., Yoon, K.J.: Online multi-object tracking via structural constraint event aggregation. In: CVPR. pp. 1392–1400 (2016)
15. Huang, C., Lucey, S., Ramanan, D.: Learning policies for adaptive tracking with deep feature cascades. In: ICCV. pp. 105–114 (2017)
16. Kamalapurkar, R., Andrews, L., Walters, P., Dixon, W.E.: Model-based reinforcement learning for infinite-horizon approximate optimal tracking. TNNLS **28**(3), 753–758 (2017)
17. Keuper, M., Tang, S., Zhongjie, Y., Andres, B., Brox, T., Schiele, B.: A multi-cut formulation for joint segmentation and tracking of multiple objects. arXiv preprint arXiv:1607.06317 (2016)
18. Kim, C., Li, F., Ciptadi, A., Rehg, J.M.: Multiple hypothesis tracking revisited. In: ICCV. pp. 4696–4704 (2015)
19. Kim, D.Y., Jeon, M.: Data fusion of radar and image measurements for multi-object tracking via kalman filtering. Information Sciences **278**, 641–652 (2014)
20. Kong, X., Xin, B., Wang, Y., Hua, G.: Collaborative deep reinforcement learning for joint object search. In: CVPR. pp. 1695–1704 (2017)
21. Le, N., Heili, A., Odobez, J.M.: Long-term time-sensitive costs for crf-based tracking by detection. In: ECCV. pp. 43–51 (2016)

22. Leal-Taixé, L., Milan, A., Reid, I., Roth, S., Schindler, K.: Motchallenge 2015: Towards a benchmark for multi-target tracking. arXiv preprint arXiv:1504.01942 (2015)
23. Levinkov, E., Uhrig, J., Tang, S., Omran, M., Insafutdinov, E., Kirillov, A., Rother, C., Brox, T., Schiele, B., Andres, B.: Joint graph decomposition & node labeling: Problem, algorithms, applications. In: CVPR. pp. 6012–6020 (2017)
24. Li, Y., Huang, C., Nevatia, R.: Learning to associate: Hybridboosted multi-target tracker for crowded scene. In: CVPR. pp. 2953–2960 (2009)
25. Liang, X., Lee, L., Xing, E.P.: Deep variation-structured reinforcement learning for visual relationship and attribute detection. arXiv preprint arXiv:1703.03054 (2017)
26. Liu, S., Zhu, Z., Ye, N., Guadarrama, S., Murphy, K.: Optimization of image description metrics using policy gradient methods. arXiv preprint arXiv:1612.00370 (2016)
27. Maksai, A., Wang, X., Fleuret, F., Fua, P.: Non-markovian globally consistent multi-object tracking. In: ICCV. pp. 2544–2554 (2017)
28. Milan, A., Leal-Taixé, L., Reid, I., Roth, S., Schindler, K.: Mot16: A benchmark for multi-object tracking. arXiv preprint arXiv:1603.00831 (2016)
29. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
30. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
31. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: CVPR. pp. 4293–4302 (2016)
32. Okuma, K., Taleghani, A., De Freitas, N., Little, J.J., Lowe, D.G.: A boosted particle filter: Multitarget detection and tracking. In: ECCV. pp. 28–39 (2004)
33. Rao, Y., Lu, J., Zhou, J.: Attention-aware deep reinforcement learning for video face recognition. In: ICCV. pp. 3931–3940 (2017)
34. Sadeghian, A., Alahi, A., Savarese, S.: Tracking the untrackable: Learning to track multiple cues with long-term dependencies. arXiv preprint arXiv:1701.01909 (2017)
35. Sanchez-Matilla, R., Poiesi, F., Cavallaro, A.: Multi-target tracking with strong and weak detections. In: ECCVW. vol. 5, p. 18 (2016)
36. Shu, G., Dehghan, A., Oreifej, O., Hand, E., Shah, M.: Part-based multiple-person tracking with partial occlusion handling. In: CVPR. pp. 1815–1821 (2012)
37. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. In: ICML. pp. 387–395 (2014)
38. Son, J., Baek, M., Cho, M., Han, B.: Multi-object tracking with quadruplet convolutional neural networks. In: ICCV. pp. 5620–5629 (2017)
39. Supancic, III, J., Ramanan, D.: Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning. In: ICCV. pp. 322–331 (2017)
40. Tang, S., Andres, B., Andriluka, M., Schiele, B.: Multi-person tracking by multicut and deep matching. In: ECCV. pp. 100–111 (2016)
41. Tang, S., Andriluka, M., Andres, B., Schiele, B.: Multiple people tracking by lifted multicut and person reidentification. In: ICCV. pp. 3539–3548 (2017)
42. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: AAAI. pp. 2094–2100 (2016)
43. Vedaldi, A., Lenc, K.: Matconvnet: Convolutional neural networks for matlab. In: ACMMM. pp. 689–692 (2015)

44. Wang, S., Fowlkes, C.C.: Learning optimal parameters for multi-target tracking with contextual interactions. IJCV **122**(3), 484–501 (2017)
45. Wen, L., Lei, Z., Lyu, S., Li, S.Z., Yang, M.H.: Exploiting hierarchical dense structures on hypergraphs for multi-object tracking. TPAMI **38**(10), 1983–1996 (2016)
46. Wu, Z., Thangali, A., Sclaroff, S., Betke, M.: Coupling detection and data association for multiple object tracking. In: CVPR. pp. 1948–1955 (2012)
47. Xiang, Y., Alahi, A., Savarese, S.: Learning to track: Online multi-object tracking by decision making. In: ICCV. pp. 4705–4713 (2015)
48. Yang, B., Nevatia, R.: Multi-target tracking by online learning of non-linear motion patterns and robust appearance models. In: CVPR. pp. 1918–1925 (2012)
49. Yang, B., Nevatia, R.: An online learned crf model for multi-target tracking. In: CVPR. pp. 2034–2041 (2012)
50. Yu, L., Zhang, W., Wang, J., Yu, Y.: Seqgan: Sequence generative adversarial nets with policy gradient. In: AAAI. pp. 2852–2858 (2017)
51. Yun, S., Choi, J., Yoo, Y., Yun, K., Young Choi, J.: Action-decision networks for visual tracking with deep reinforcement learning. In: CVPR. pp. 2711–2720 (2017)
52. Zamir, A.R., Dehghan, A., Shah, M.: Gmcp-tracker: Global multi-object tracking using generalized minimum clique graphs. In: ECCV, pp. 343–356 (2012)
53. Zhang, D., Maei, H., Wang, X., Wang, Y.F.: Deep reinforcement learning for visual object tracking in videos. arXiv preprint arXiv:1701.08936 (2017)
54. Zhang, L., Li, Y., Nevatia, R.: Global data association for multi-object tracking using network flows. In: CVPR. pp. 1–8 (2008)