Multi-Agent Deep Reinforcement Learning for Multi-Object Tracking

Liangliang Ren[1,2,3],Zifeng Wang[1,4*], Jiwen Lu[1,2,3†], Jianjiang Feng[1,2,3], Jie Zhou[1,2,3]
[1]Department of Automation, Tsinghua University, Beijing, China
[2]State Key Lab of Intelligent Technologies and Systems, Beijing, China
[3]Tsinghua National Laboratory for Information Science and Technology (TNList), Beijing, China
[4]Department of Electronic Engineering, Tsinghua University, Beijing, China
renll16@mails.tsinghua.edu.cn; wangzf14@mails.tsinghua.edu.cn; lujiwen@tsinghua.edu.cn;
jfeng@tsinghua.edu.cn; jzhou@tsinghua.edu.cn

Abstract

In this paper, we propose a multi-agent deep reinforcement learning (M-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 M-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 interactions of different agents and environments via the decision network, so that the influences of occlusions and noisy detection results can be well alleviated. Experimental results on the challenging MOT15 and MOT16 benchmarks are presented to show the efficiency of our approach.

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 them across the video. While a variety of MOT methods have been proposed in recent years [7, 8, 14, 27, 35, 37, 41, 46–48, 53], 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.

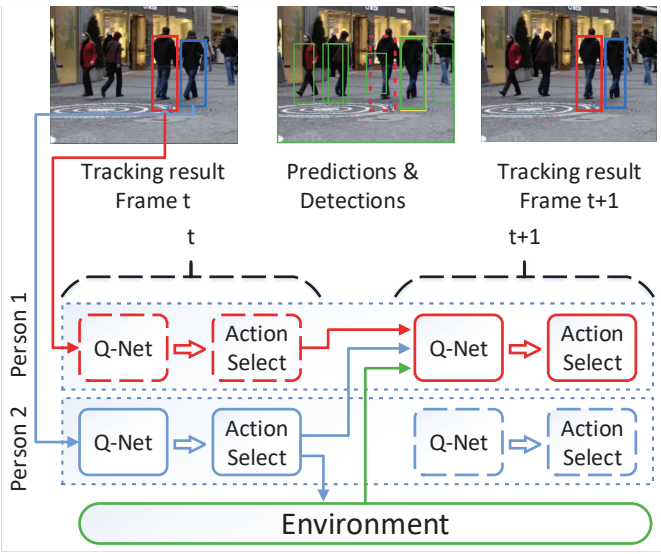

Figure 1. The key idea of our proposed M-DRL method for multi-object tracking. Given a video and the detection results of different objects for the tth frame, we model each object as an agent and predict the location of each object for the following frames, where we seek the optimal tracked results by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

Existing MOT approaches can be mainly divided into two categories, 1) offline (batch or semi-batch) [27, 41, 46, 47, 53] and 2) online [7, 8, 14, 35, 37, 48]. The key idea of offline methods is to group detections into short trajectory segments 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, 55], energy minimization [28] or generalized minimum clique

*Indicates equal contribution.
†Corresponding author.

graphs [53] 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 [33] or Markov decisions [48]. 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 multi-agent deep reinforcement learning (M-DRL) method for multi-object tracking. Figure 1 illustrates the basic idea of our proposed approach. Given a video and the detection results of different objects for the tth 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 the 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 [27, 41, 46, 47, 53] and 2) online [7, 8, 14, 35, 37, 48]. Methods 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.* [53] associated all detection results which incorporate both the appearance and motion information in a global manner by using generalized minimum clique graphs. Wen *et al.* [46] exploited dense structures from hierarchically constructed undirected affinity hypergraphs of tracklets. Tang *et al.* [41] 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. Wu *et al.* [47] proposed a multi-object tracking framework that couples object detection and data association. Methods in the second class estimate object trajectories with the detection results of the current and past frames. For example, Yang *et al.* [49, 50] 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.* [48] 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.* [35] 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.* [37] 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 [34], image super-resolution [6] and object search [20]. Current deep reinforcement learning methods can be divided into two classes: deep Q learning [12, 30, 31, 43] and policy gradient [1, 38, 51]. 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.* [34] 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.* [51] proposed a sequence generative adversarial nets with policy gradient. More recently, deep reinforcement learning [15, 16, 40, 52, 54] has also been employed in visual tracking. For example, Yun *et al.* [52] proposed an action-decision network to generate actions to seek the locations and sizes of the objects in a new coming frame. Supancic *et al.* [40] 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 multi-agent deep reinforcement learning method to exploit the interactions of different objects for multi-object tracking.

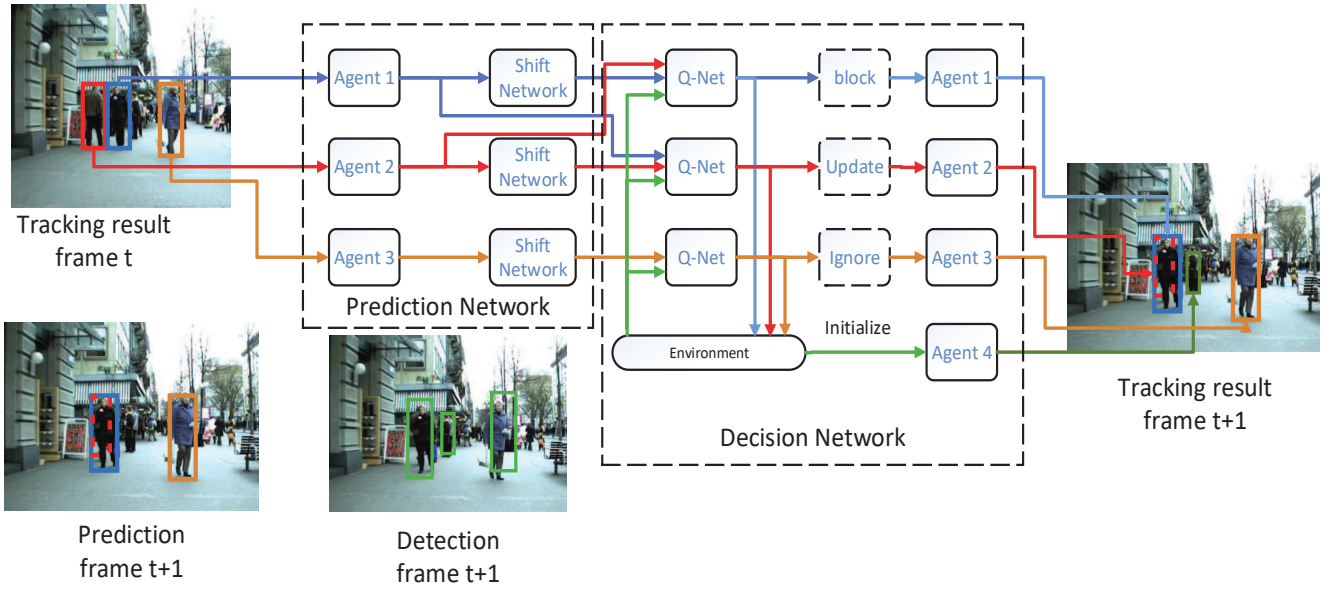

Figure 2. The framework of the proposed M-DRL for multi-object tracking. In this figure, there are three objects in the tth frame. We first predict the locations of these three objects in the $t+1$th frame. Then we use a decision network to combine the prediction and detection results and make decisions 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 in the $(t+1)$th frame.

3. Approach

Figure 2 shows the framework of the proposed M-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 in the tth frame, 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 in the $(t+1)$th frame according to decisions. In the following subsections, we will detail the prediction network and the decision network, respectively.

3.1. The Prediction Network

Given initial locations of objects, the prediction network aims to learn the movement of objects to compute the locations of the target object. As shown in Figure 3, the input to the prediction network is the raw image cropped by the initial bounding box of the next frame. 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

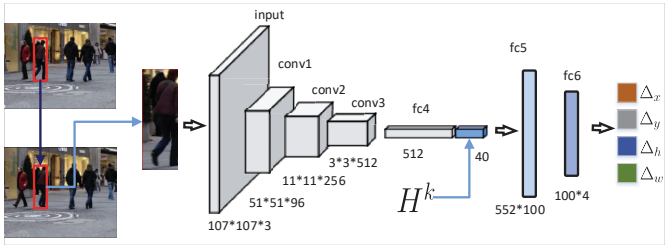

Figure 3. The framework of the prediction network. The prediction network learns the movement of the target object given an initial location of the object, which contains three convolutional layers and three fully connected layers.

location prediction as the following regression problem:

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

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

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

Algorithm 1 details the procedure of learning the prediction network.

Algorithm 1: Learning the Prediction Network

Input: Training set: $\mathbf{V} = \{V_i\}$, and convergence error ϵ.
Output: $\{W^l, b^l\}_{l=1}^{L}$

Initialize $\{W^l, b^l\}_{l=1}^{L}$;
for all $l = 1, 2, \ldots, I_t$ **do**
  Randomly select a video (V);
  Sample bounding boxes $\mathcal{B} = \{b_{i,t}\}$;
  Delete the $b_{i,t} \in \mathcal{B}$ if IoU($b_{i,t}, b^*_{i,t}$)<0.7;
  Obtain gradient of ϕ;
  Update ϕ;
  Calculate J_t according to (1);
  if $l > 1$ and $|J_t - J_{t-1}| < \epsilon$ **then** Go to **return** ϕ
  end if
end for
return f

3.2. The Decision Network

As shown in Figure 2, the decision network is a multi-agent 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 multi-agent 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 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:

$$d(p_i, p_j) = -\alpha g(p_i, p_j) + \|f_i - f_j\|_2^2 \tag{3}$$

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 pth object and the detection results can be computed as follows:

$$d(p_i, p_j^*) = -\alpha g(p_i, p_j^*) + \|f_i - f_j^*\|_2^2 \tag{4}$$

Let I_t be the tth frame of the selected video, which contains n_t objects, $\mathcal{P}_t = \{p_1, p_2, \cdots, p_{n_t}\}$. The state in the tth frame $s_t = \{\mathcal{P}_t, \mathcal{P}_t^*\}$ contains the current agents and the detection results. For the ith object p_i, we first use a prediction network to generate the position in the $(t+1)$th frame. 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 ρ 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:

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

where γ is the decaying parameter.

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

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

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

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

The gradient shows that we 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 and in some challenging cases (or at the beginning of the training stage), the Q values of all actions are negative. 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:

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

Then, the advantage value is computed as follows:

$$A(s,a) = Q(s,a) - V(s) \quad (13)$$

The final formula of the policy gradient with an entropy regularizer can be defined as:

$$L(\theta) = \mathbb{E}_{s,a} \log(\pi(a|s,\theta))A(s,a) + \alpha \mathbb{E}_s H^\pi(s) \quad (14)$$

where $H^\pi(s) = -\sum_a \pi(a|s,\theta) \log(\pi(a|s,\theta))$ is the entropy of the policy, and $\alpha > 0$ is the balance parameter.

The parameter θ can be updated as follows:

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

Algorithm 2 summarizes the detailed learning procedure of our decision network.

Algorithm 2: Learning the Decision Network

Input: Training set: $\mathbf{V} = \{V_i\}$, and convergence error ϵ_1.
Output: θ

```
Initialize θ;
for all l = 1, 2, ..., I_t do
    Randomly select a video (V);
    Initialize agents set P using the detection results in 1-st frame
    for all t = 2, 3, ..., I_t do
        for all p ∈ P do
            Take actions according to the output of multi-agent deep networks;
            Update or delete p
        end for
        Add p_i^* ∈ P^*
    end for
    Calculate L_t according to (10);
    Calculate advantage value A(s, a) for each agent;
    Update policy network θ according to (15) ;
    if l > 1 and |L_t − L_{t−1}| < ε_1 then
        Go to return
    end if
end for
return θ
```

4. Experiments

To evaluate the effectiveness of the proposed M-DRL method, we conducted experiments on the MOT15 [22] and MOT16 [28] datasets and compared it with state-of-the-art MOT methods. We detail the experimental results and analysis in the following.

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 [32] which was pretrained 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.

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 M-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 [44] toolbox.

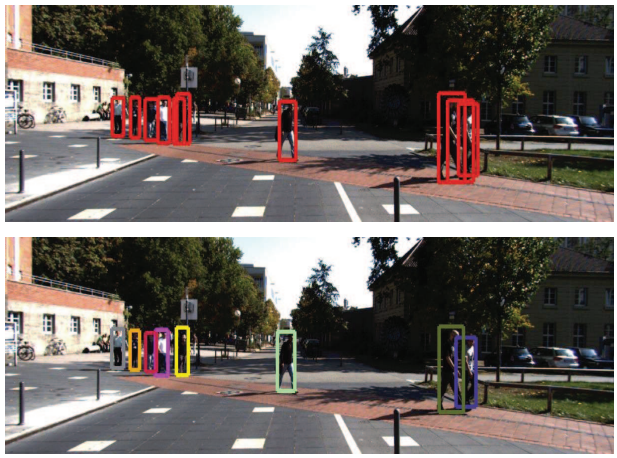

Figure 5. The image on the top is detection result and the one on the bottom is our tracking result. Our tracking system successfully eliminate the false positives.

4.4. Evalutaions on MOT15

Comparison with State-of-the-arts: For a fair comparison, we used the public detection results on MOT15 and MOT16. As shown in Table 2, 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 [35]. 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

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

THRESH	Rcll	Prcn	GT	MT	ML	FP	FN	IDs	MOTA	MOTP	MOTAL
1	83.1	81.7	458	321	31	7300	6598	481	63.3	85.0	64.5
2	83.0	83.3	458	316	34	6520	6673	440	65.2	85.0	66.3
3	82.8	83.9	458	313	36	6214	6742	411	65.9	85.0	66.9
4	82.5	84.6	458	313	41	5861	6837	380	66.6	85.1	67.6
5	82.3	85.2	458	311	45	5588	6929	359	67.1	85.1	68.0
6	82.1	85.6	458	308	51	5392	7003	348	67.4	85.1	68.3
7	81.8	86.1	458	302	60	5181	7142	329	67.7	85.2	68.5
8	81.4	86.4	458	293	66	4995	7292	307	67.8	85.2	68.6
9	81.0	86.7	458	287	73	4854	7448	293	67.8	85.2	68.6

Table 2. The performance of different methods on MOT15.

Mode	Method	MOTA↑	MOTP↑	FAF↓	MT(%)↑	ML(%)↓	FP↓	FN↓	ID Sw↓	Frag ↓
Offline	LINF1 [10]	24.5	71.3	1.0	5.5	64.6	5864	40207	**298**	**744**
	LP_SSVM [45]	25.2	71.7	1.4	5.8	53.0	8369	36932	646	849
	MHT_DAM [18]	32.4	71.8	1.6	16.0	43.8	9064	32060	435	826
	NMOT [7]	33.7	**71.9**	1.3	12.2	44.0	7762	32547	442	832
	QuadMOT [39]	33.8	73.4	1.4	12.9	36.9	7898	32061	703	1430
	JointMC [17]	35.6	**71.9**	1.8	**23.2**	39.3	10580	**28508**	457	969
Online	SCEA [14]	29.1	71.1	1.0	8.9	47.3	6060	36912	604	1182
	MDP [48]	30.3	71.3	1.7	13.0	38.4	9717	32422	680	1500
	CDA_DDALpb [2]	32.8	70.7	**0.9**	9.7	42.2	**4983**	35690	614	1583
	AMIR15 [35]	**37.6**	71.7	1.4	15.8	**26.8**	7933	29397	1026	2024
	Ours	37.1	71.0	1.2	14.0	31.3	7036	30440	1115	2223

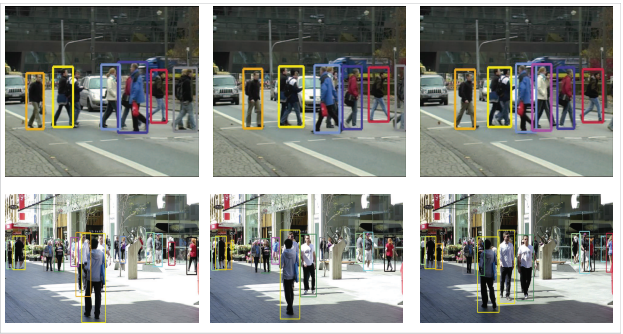

Figure 6. Our method can recognize the sources of ID switches from the sample frames above. For the first row, we could see that when people walk by each other, it is easy for them to switch their ids. For example the woman in white is initially in blue box, however the blue box moves to the man in blue 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.

framework.

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 M-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 and conducted experiments on the SubCNN detection of MOT15 training set which was provided in [48]. 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 ≥ 8, setting THRESH to be 8 is a good choice for optimizing MOTA. Our method achieved relatively lower MT because MOT15 public detection is a noisy dataset.

4.5. Evaluations on MOT16

Comparison with the state-of-the-art: As shown in Table 3, our method achieved the best MOTA result among all online MOT methods and is comparable to the best offline methods such as LMP [42] and FWT [13]. In terms of MT and ML, our method also achieved 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

Table 3. The performance of different methods on MOT16.

Mode	Method	MOTA↑	MOTP↑	FAF↓	MT(%)↑	ML(%)↓	FP↓	FN↓	ID Sw↓	Frag ↓
Offline	CEM [29]	33.2	75.8	1.2	7.8	54.4	6837	114322	642	731
	TBD [11]	33.7	76.5	1.0	7.2	54.2	5804	112587	2418	2252
	LTTSC-CRF [21]	37.6	75.9	2.0	9.6	55.2	11969	101343	481	1012
	LINF1 [10]	41.0	74.8	1.3	11.6	51.3	7896	99224	430	963
	MHT_DAM_16 [18]	45.8	76.3	1.1	16.2	43.2	6412	91758	590	781
	NOMT [7]	46.4	76.7	1.6	**18.3**	41.4	9753	87565	**359**	**504**
	NLLMPa [23]	47.6	78.5	1.0	17.0	40.4	5844	89093	629	768
	LMP [42]	**48.8**	**79.0**	1.1	18.2	40.1	6654	**86245**	481	595
Online	OVBT [3]	38.4	75.4	1.9	7.5	47.3	11517	99463	1321	2140
	EAMTT_pub [36]	38.8	75.1	1.4	7.9	49.1	8114	102452	965	1657
	CDA_DDALv2 [2]	43.9	74.7	1.1	10.7	44.4	6450	95175	676	1795
	AMIR [35]	47.2	75.8	**0.5**	14.0	41.6	**2681**	92856	774	1675
	Ours	47.3	74.6	1.1	17.4	**39.9**	6375	88543	1215	1906

Figure 4. Some detection 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.

our method obtained 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 address this decently. We can also see that our method outperforms these offline methods by a large margin, which shows that the effectiveness of the decision network where multi-agent interaction maximizing the utilization of contextual information is effectively exploited to enhance the generalization ability of our network.

Failure Cases: Figure 4 shows some failure examples of our method. Our method has a relatively high ID switch and Frag (actually these two metrics are closely correlated) on both 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 multi-agent 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 a deep multi-agent reinforcement learning network 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.

References

[1] H. B. Ammar, E. Eaton, P. Ruvolo, and M. Taylor. Online multi-task learning for policy gradient methods. In *ICML*, pages 1206–1214, 2014. 2

[2] S.-H. Bae and K.-J. Yoon. Confidence-based data association and discriminative deep appearance learning for robust online multi-object tracking. *TPAMI*, 2017. 7, 8

[3] Y. Ban, S. Ba, X. Alameda-Pineda, and R. Horaud. Tracking multiple persons based on a variational bayesian model. In *ECCV*, pages 52–67, 2016. 8

[4] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP*, 2008(1):246309, 2008. 5

[5] A. A. Butt and R. T. Collins. Multi-target tracking by lagrangian relaxation to min-cost network flow. In *CVPR*, pages 1846–1853, 2013. 1

[6] Q. Cao, L. Lin, Y. Shi, X. Liang, and G. Li. Attention-aware face hallucination via deep reinforcement learning. In *CVPR*, pages 690–698, 2017. 2

[7] W. Choi. Near-online multi-target tracking with aggregated local flow descriptor. In *ICCV*, pages 3029–3037, 2015. 1, 2, 7, 8

[8] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu. Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism. In *ICCV*, pages 4836–4845, 2017. 1, 2

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 6

[10] L. Fagot-Bouquet, R. Audigier, Y. Dhome, and F. Lerasle. Improving multi-frame data association with sparse representations for robust near-online multi-object tracking. In *ECCV*, pages 774–790, 2016. 6, 7, 8

[11] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun. 3d traffic scene understanding from movable platforms. *TPAMI*, 36(5):1012–1025, 2014. 8

[12] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine. Continuous deep q-learning with model-based acceleration. In *ICML*, pages 2829–2838, 2016. 2

[13] R. Henschel, L. Leal-Taixé, D. Cremers, and B. Rosenhahn. Improvements to frank-wolfe optimization for multi-detector multi-object tracking. *arXiv preprint arXiv:1705.08314*, 2017. 7

[14] J. Hong Yoon, C.-R. Lee, M.-H. Yang, and K.-J. Yoon. Online multi-object tracking via structural constraint event aggregation. In *CVPR*, pages 1392–1400, 2016. 1, 2, 7

[15] C. Huang, S. Lucey, and D. Ramanan. Learning policies for adaptive tracking with deep feature cascades. In *ICCV*, pages 105–114, 2017. 2

[16] R. Kamalapurkar, L. Andrews, P. Walters, and W. E. Dixon. Model-based reinforcement learning for infinite-horizon approximate optimal tracking. *TNNLS*, 28(3):753–758, 2017. 2

[17] M. Keuper, S. Tang, Y. Zhongjie, B. Andres, T. Brox, and B. Schiele. A multi-cut formulation for joint segmentation and tracking of multiple objects. *arXiv preprint arXiv:1607.06317*, 2016. 7

[18] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg. Multiple hypothesis tracking revisited. In *ICCV*, pages 4696–4704, 2015. 7, 8

[19] D. Y. Kim and M. Jeon. Data fusion of radar and image measurements for multi-object tracking via kalman filtering. *Information Sciences*, 278:641–652, 2014. 2

[20] X. Kong, B. Xin, Y. Wang, and G. Hua. Collaborative deep reinforcement learning for joint object search. In *CVPR*, pages 1695–1704, 2017. 2

[21] N. Le, A. Heili, and J.-M. Odobez. Long-term time-sensitive costs for crf-based tracking by detection. In *ECCV*, pages 43–51, 2016. 8

[22] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*, 2015. 5

[23] E. Levinkov, J. Uhrig, S. Tang, M. Omran, E. Insafutdinov, A. Kirillov, C. Rother, T. Brox, B. Schiele, and B. Andres. Joint graph decomposition & node labeling: Problem, algorithms, applications. In *CVPR*, pages 6012–6020, 2017. 8

[24] Y. Li, C. Huang, and R. Nevatia. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *CVPR*, pages 2953–2960, 2009. 5

[25] X. Liang, L. Lee, and E. P. Xing. Deep variation-structured reinforcement learning for visual relationship and attribute detection. *arXiv preprint arXiv:1703.03054*, 2017. 2

[26] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy. Optimization of image description metrics using policy gradient methods. *arXiv preprint arXiv:1612.00370*, 2016. 2

[27] A. Maksai, X. Wang, F. Fleuret, and P. Fua. Non-markovian globally consistent multi-object tracking. In *ICCV*, pages 2544–2554, 2017. 1, 2

[28] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016. 1, 5

[29] A. Milan, S. Roth, and K. Schindler. Continuous energy minimization for multitarget tracking. *TPAMI*, 36(1):58–72, 2014. 8

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 2

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. 2

[32] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, pages 4293–4302, 2016. 6

[33] K. Okuma, A. Taleghani, N. De Freitas, J. J. Little, and D. G. Lowe. A boosted particle filter: Multitarget detection and tracking. In *ECCV*, pages 28–39, 2004. 2

[34] Y. Rao, J. Lu, and J. Zhou. Attention-aware deep reinforcement learning for video face recognition. In *ICCV*, pages 3931–3940, 2017. 2

[35] A. Sadeghian, A. Alahi, and S. Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. *arXiv preprint arXiv:1701.01909*, 2017. 1, 2, 6, 7, 8

[36] R. Sanchez-Matilla, F. Poiesi, and A. Cavallaro. Multi-target tracking with strong and weak detections. In *ECCVW*, volume 5, page 18, 2016. 8

[37] G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah. Part-based multiple-person tracking with partial occlusion handling. In *CVPR*, pages 1815–1821, 2012. 1, 2

[38] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, pages 387–395, 2014. 2

[39] J. Son, M. Baek, M. Cho, and B. Han. Multi-object tracking with quadruplet convolutional neural networks. In *ICCV*, pages 5620–5629, 2017. 7

[40] J. Supancic, III and D. Ramanan. Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning. In *ICCV*, pages 322–331, 2017. 2

[41] S. Tang, B. Andres, M. Andriluka, and B. Schiele. Multi-person tracking by multicut and deep matching. In *ECCV*, pages 100–111, 2016. 1, 2

[42] S. Tang, M. Andriluka, B. Andres, and B. Schiele. Multiple people tracking by lifted multicut and person reidentification. In *ICCV*, pages 3539–3548, 2017. 7, 8

[43] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *AAAI*, pages 2094–2100, 2016. 2

[44] A. Vedaldi and K. Lenc. Matconvnet: Convolutional neural networks for matlab. In *ACMMM*, pages 689–692, 2015. 6

[45] S. Wang and C. C. Fowlkes. Learning optimal parameters for multi-target tracking with contextual interactions. *IJCV*, 122(3):484–501, 2017. 7

[46] L. Wen, Z. Lei, S. Lyu, S. Z. Li, and M.-H. Yang. Exploiting hierarchical dense structures on hypergraphs for multi-object tracking. *TPAMI*, 38(10):1983–1996, 2016. 1, 2

[47] Z. Wu, A. Thangali, S. Sclaroff, and M. Betke. Coupling detection and data association for multiple object tracking. In *CVPR*, pages 1948–1955, 2012. 1, 2

[48] Y. Xiang, A. Alahi, and S. Savarese. Learning to track: Online multi-object tracking by decision making. In *ICCV*, pages 4705–4713, 2015. 1, 2, 7

[49] B. Yang and R. Nevatia. Multi-target tracking by online learning of non-linear motion patterns and robust appearance models. In *CVPR*, pages 1918–1925, 2012. 2

[50] B. Yang and R. Nevatia. An online learned crf model for multi-target tracking. In *CVPR*, pages 2034–2041, 2012. 2

[51] L. Yu, W. Zhang, J. Wang, and Y. Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858, 2017. 2

[52] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Young Choi. Action-decision networks for visual tracking with deep reinforcement learning. In *CVPR*, pages 2711–2720, 2017. 2

[53] A. R. Zamir, A. Dehghan, and M. Shah. Gmcp-tracker: Global multi-object tracking using generalized minimum clique graphs. In *ECCV*, pages 343–356. 2012. 1, 2

[54] D. Zhang, H. Maei, X. Wang, and Y.-F. Wang. Deep reinforcement learning for visual object tracking in videos. *arXiv preprint arXiv:1701.08936*, 2017. 2

[55] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *CVPR*, pages 1–8, 2008. 1