2015 IEEE International Conference on Computer Vision

Learning to Track: Online Multi-Object Tracking by Decision Making

Yu Xiang[1,2], Alexandre Alahi[1], and Silvio Savarese[1]
[1]Stanford University, [2]University of Michigan at Ann Arbor
yuxiang@umich.edu, {alahi, ssilvio}@stanford.edu

Abstract

Online Multi-Object Tracking (MOT) has wide applications in time-critical video analysis scenarios, such as robot navigation and autonomous driving. In tracking-by-detection, a major challenge of online MOT is how to robustly associate noisy object detections on a new video frame with previously tracked objects. In this work, we formulate the online MOT problem as decision making in Markov Decision Processes (MDPs), where the lifetime of an object is modeled with a MDP. Learning a similarity function for data association is equivalent to learning a policy for the MDP, and the policy learning is approached in a reinforcement learning fashion which benefits from both advantages of offline-learning and online-learning for data association. Moreover, our framework can naturally handle the birth/death and appearance/disappearance of targets by treating them as state transitions in the MDP while leveraging existing online single object tracking methods. We conduct experiments on the MOT Benchmark [24] to verify the effectiveness of our method.

1. Introduction

Tracking multiple objects in videos is an important problem in computer vision which has wide applications in various video analysis scenarios, such as visual surveillance, sports analysis, robot navigation and autonomous driving. In cases where objects in a specific category are to be tracked, such as people or cars, a category detector can be utilized to facilitate tracking. Recent progress on Multi-Object Tracking (MOT) has focused on the tracking-by-detection strategy, where object detections from a category detector are linked to form trajectories of the targets. In order to resolve ambiguities in associating object detections and to overcome detection failures, most of these recent works [7, 11, 27, 23] process video sequences in a *batch mode* in which video frames from future time steps are also utilized to solve the data association problem. However, such non-causal systems are not suitable for online tracking applications like robot navigation and autonomous driving.

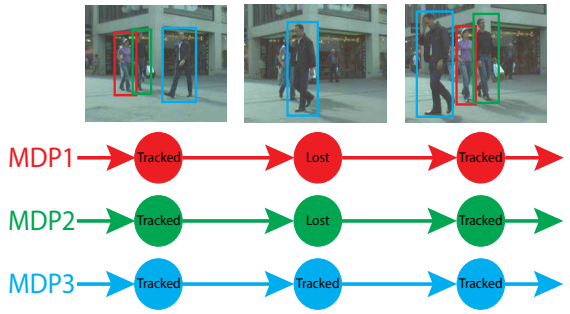

Figure 1. We formulate the online multi-object tracking problem as decision making in a Markov Decision Process (MDP) framework.

For tracking-by-detection in the *online mode*, the major challenge is how to associate noisy object detections in the current video frame with previously tracked objects. The basis for any data association algorithm is a similarity function between object detections and targets. To handle ambiguities in association, it is useful to combine different cues in computing the similarity, such as appearance, motion, and location. Most previous works rely on heuristically selected parametric models for the similarity function and tune these parameters by cross-validation, which is not scalable to the number of features and does not necessarily guarantee generalization power of the model.

Recently, there is a trend on learning to track that advocates the concept of injecting learning capabilities to MOT [38, 25, 22, 20, 4]. Based on their learning schemes, we can categorize these methods into *offline-learning* methods and *online-learning* methods. In offline-learning, learning is performed before the actual tracking takes place. For instance, [25, 20] use supervision from ground truth trajectories offline to learn a similarity function between detections and tracklets for data association. As a result, offline-learning is static: it cannot take into account the dynamic status and the history of the target in data association, which is important to resolve ambiguities, especially when it needs to re-assign missed or occluded objects when they appear again. In contrast, online-learning conducts learning during tracking. A common strategy is to construct positive and negative training examples according to the tracking results, and then to train a similarity function for data association

1550-5499/15 $31.00 © 2015 IEEE
DOI 10.1109/ICCV.2015.534

(e.g., [38, 22, 4]). Online-learning is able to utilize features based on the status and the history of the target. However, there are no ground truth annotations available for supervision. So the method is likely to learn from incorrect training examples if there are errors in the tracking results, and these errors can be accumulated and result in *tracking drift*.

In this work, we formulate the online multi-object tracking problem (MOT in the online mode) as decision making in Markov Decision Processes (MDPs), where the lifetime of an object is modeled with a MDP, and multiple MDPs are assembled for multi-object tracking (Fig. 1). In our framework, learning a similarity function for data association is equivalent to learning a policy for the MDP. The policy learning is approached in a reinforcement learning fashion which benefits from advantages of both offline-learning and online-learning in data association. First, learning in our method is conducted offline so as to utilize supervision from ground truth trajectories. Second, learning in our method takes place while tracking objects in training sequences, so the MDP is able to make the decision based on both the current status and the history of the target. Specifically, given the ground truth trajectory of a target and an initial similarity function, the MDP attempts to track the target and collects feedback from the ground truth. According to the feedback, the MDP updates the similarity function to improve tracking. The similarity function is updated only when the MDP makes a mistake in data association, which enables us to collect hard training examples to learn the similarity function. Finally, training is finished when the MDP can successfully track the target.

In addition to the advantages of our learning strategy, our framework can naturally handle the birth/death and appearance/disappearance of targets by treating them as state transitions in the MDP. Our method also benefits from the strengths of online single object tracking methods [3, 15, 16, 5], where we learn and update an appearance model for a target online in order to handle object detection failures. We conduct experiments on the recently introduced benchmark for multi-object tracking [24]. Our extensive system analysis and comparison with the state-of-the-art tracking methods on the MOT benchmark demonstrate the superiority of our method.

2. Related Work

Multi-Object Tracking. Recent research in MOT has focused on the tracking-by-detection principal, where the main challenge is the data association problem in linking object detections. Majority of the batch methods ([43, 25, 31, 7, 36, 11, 27]) formulates MOT as a global optimization problem in a graph-based representation, while online methods solve the data association problem either probabilistically [33, 19, 32] or determinatively (e.g., Hungarian algorithm [29] in [20, 4] or greedy association [10]). A core component in any data association algorithm is a similarity function between objects. Both batch methods [25, 22] and online methods [38, 20, 4] have explored the idea of learning to track, where the goal is to learn a similarity function for data association from training data. Our main contribution in this work is a novel reinforcement learning algorithm for data association in online MOT.

Online Single Object Tracking. In single object tracking, the state-of-the-art trackers [3, 15, 16, 5, 41, 39, 34, 40] focus on how to learn a strong appearance model of the target online and use it for tracking. It is non-trivial to apply these trackers to MOT since they are not able to handle the entering/exiting of objects from the scene. The initial location of the target needs to be specified before the tracking starts, and they assume that the target exists in the whole video sequence. Additionally, online single object trackers are likely to drift if the appearance of the target changes significantly. Another contribution of our work is that by modeling the lifetime of an object with a MDP, we are able to take the advantages of existing online single object trackers to facilitate MOT, while overcoming their limitations by using object detection as additional cues.

MDP in Vision. Markov decision processes [6] have been applied to different computer vision tasks, such as feature selection for recognition [35, 17], human activity forecasting [21], video game playing [28] and human-machine collaboration [37]. MDP is suitable for dynamic environments where an agent needs to perform certain tasks by making decisions and executing actions sequentially. In our framework, we consider a single object tracker to be an agent in MDP, whose task is to track the target. Then we learn a good policy for the MDP with reinforcement learning, and employ multiple MDPs to track multiple targets.

3. Online Multi-Object Tracking Framework

In Sec. 3.1 and Sec. 3.2, we introduce our Markov decision process formulation in modeling the lifetime of a single target in object tracking, then we present our method using multiple MDPs for online multi-object tracking in Sec. 3.3.

3.1. Markov Decision Process

In our framework, the lifetime of a target is modeled with a Markov Decision Process (MDP). The MDP consists of the tuple $(\mathcal{S}, \mathcal{A}, T(\cdot), R(\cdot))$:

- The target state $s \in \mathcal{S}$ encodes the status of the target.
- The action $a \in \mathcal{A}$ which can be performed to a target.
- The state transition function $T : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ describes the effect of each action in each state.
- The real-valued reward function $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ defines the immediate reward received after executing action a to state s.

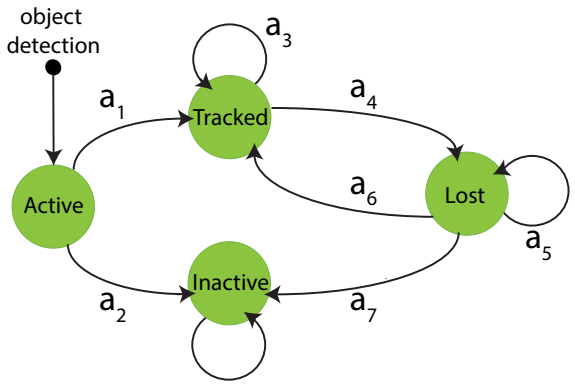

Figure 2. The target MDP in our framework.

States. We partition the state space in the target MDP into four subspaces, i.e., $\mathcal{S} = \mathcal{S}_{\text{Active}} \cup \mathcal{S}_{\text{Tracked}} \cup \mathcal{S}_{\text{Lost}} \cup \mathcal{S}_{\text{Inactive}}$, where each subspace contains infinity number of states which encode the information of the target depending on the feature representation, such as appearance, location, size and history of the target. Fig. 2 illustrates the transitions between the four subspaces. "Active" is the initial state for any target. Whenever an object is detected by the object detector, it enters an "Active" state. An active target can transition to "Tracked" or "Inactive". Ideally, a true positive from object detector should transition to a "Tracked" state, while a false alarm should enter an "Inactive" state. A tracked target can keep tracked, or transition to "Lost" if the target is lost due to some reason, such as occlusion, or disappearance from the field of view of the camera. Likewise, a lost target can stay as lost, or go back to "Tracked" if it appears again, or transition to "Inactive" if it has been lost for a sufficiently long time. Finally, "Inactive" is the terminal state for any target, i.e., an inactive target stays as inactive forever.

Actions and Transition Function. Seven possible transitions are designed between the states of a target, which correspond to seven actions in our target MDP. Fig. 2 illustrate these transitions and actions. In the MDP, all the actions are deterministic, i.e., given the current state and an action, we specify a new state for the target. For example, executing action a_4 on a tracked target would transfer the target into a lost state, i.e., $T(s_{\text{Tracked}}, a_4) = s_{\text{Lost}}$.

Reward Function. In our MDP, the reward function is not given but needs to be learned from training data, i.e., an inverse reinforcement learning problem [30], where we use ground truth trajectories of the targets as supervision.

3.2. Policy

In MDP, a policy π is a mapping from the state space $\mathcal{S}$ to the action space $\mathcal{A}$, i.e., $\pi : \mathcal{S} \mapsto \mathcal{A}$. Given the current state of the target, a policy determines which action to take. Equivalently, the decision making in MDP is performed by following a policy. The goal of policy learning is to find a policy which maximizes the total rewards obtained. In this section, we first describe our policies designed for the Active subspace and the Tracked subspace, then we present a novel reinforcement learning algorithm to learn a good policy for data association in the Lost subspace.

3.2.1 Policy in an Active State

In an Active state s, the MDP makes the decision between transferring an object detection into a tracked or inactive target to deal with noisy detections. This decision making can be considered to be a preprocessing step before tracking. Strategies such as non-maximum suppression or thresholding detection scores are usually used. In our implementation, we train a binary Support Vector Machine (SVM) [8] offline to classify a detection into tracked or inactive using a normalized 5D feature vector $\phi_{\text{Active}}(s)$, i.e., 2D coordinates, width, height and score of the detection, where training examples are collected from training video sequences. This is equivalent to learning the reward function in Active:

$$R_{\text{Active}}(s, a) = y(a)\big(\mathbf{w}_{\text{Active}}^T \phi_{\text{Active}}(s) + b_{\text{Active}}\big), \quad (1)$$

where $(\mathbf{w}_{\text{Active}}, b_{\text{Active}})$ defines the hyperplane in SVM, $y(a) = +1$ if action $a = a_1$, and $y(a) = -1$ if $a = a_2$ in Fig. 2. Note that a false alarm from object detector can still be miss-classified and transfered to a tracked state, which will be handled by the MDP in the tracked and lost states.

3.2.2 Policy in a Tracked State

In a Tracked state, the MDP needs to decide whether to keep tracking the target or to transfer it into a lost state. As long as the target is not occluded and is in the camera's field of view, we should keep tracking it. Otherwise, it should be marked as lost. This decision making is related to the goal of single object tracking in the literature [3, 15, 16, 5]. Inspired by these works, we build an appearance model for the target online and use it to track the target. If the appearance model is able to successfully track the target in the next video frame, the MDP leaves the target in a tracked state. Otherwise, the target is transferred to a lost state. Our framework is general to utilize different approaches in building the appearance model. We describe our implementation based on the TLD tracker [16] in this work.

Template Representation. The appearance of the target is simply represented by a template that is an image patch of the target in a video frame. Whenever an object detection is transferred to a tracked target, we initialize the target template with the detection bounding box. Fig. 3(a) illustrates a template for a pedestrian. When the target is being tracked, the MDP collects its templates in the tracked frames to represent the history of the target, which will be used in the lost state for decision making.

(a) target template (b) stable prediction (c) unstable prediction

Figure 3. The appearance of the target is represented by a template in a video frame (a). We compute optical flow from densely sampled points inside the target template to a new frame. The quality of the flow is used as a cue to make the decision: (b) an example of stable prediction; (c) an example of unstable prediction due to partial occlusion, where we show both the cropped frames and the origin frames. The yellow box is the predicted location of the target.

Template Tracking. In order to use the target template for tracking, we compute an optical flow from densely and uniformly sampled points inside the template to a new video frame. Specifically, given a point $\mathbf{u} = (u_x, u_y)$ on the target template I, we find its corresponding location $\mathbf{v} = \mathbf{u}+\mathbf{d} = (u_x+d_x, u_y+d_y)$ in the new frame J using the iterative Lucas-Kanade method with pyramids [9], where $\mathbf{d} = (d_x, d_y)$ is the optical flow at $\mathbf{u}$. After computing the optical flow of all the sampled points, we use the Forward-Backward (FB) error defined in [16] to measure how stable the predict is. Given the prediction $\mathbf{v}$ of point $\mathbf{u}$ on the target template, we can compute the backward flow of point $\mathbf{v}$ to the target template and obtain a new prediction $\mathbf{u}'$. If the optical flow is stable, $\mathbf{u}$ and $\mathbf{u}'$ should be close to each other. So FB error of a point is defined as the Euclidean distance between the original point and the forward-backward prediction: $e(\mathbf{u}) = \|\mathbf{u} - \mathbf{u}'\|^2$, and the stability of the tracking is measured using the median of the FB errors of all sampled points: $e_{\text{medFB}} = \text{median}(\{e(\mathbf{u}_i)\}_{i=1}^n)$, where n is the number of points. If e_{medFB} is larger than some threshold, the tracking is considered to be unstable. Moreover, after filtering out unstable matches whose FB error is larger than the threshold, we can predict a bounding box for the target using the remaining matches, which is treated as the new location of the target. Fig. 3 (b) and (c) illustrate the optical flow in a stable case and an unstable case respectively. As we can see, the quality of the optical flow is an important cue to decide whether to keep tracking the target or not.

However, it is risky to make the decision based on optical flow only. Because the tracked target can be a false alarm from the object detector (see Sec. 3.2.1), whose appearance may not change, such as a detection on the background of the scene. In this case, the optical flow tracker will keep tracking the false alarm. To handle this case, we resort to the object detector. The intuition is that a false alarm cannot be consistently detected. If a tracked target does not encounter object detections for a while, it is likely to be a false alarm. So we examine the history of the target, and compute the bounding box overlap $o(t_k, \mathcal{D}_k)$ between the target t_k in k frames before and the corresponding detections $\mathcal{D}_k$. Then we compute the mean bounding box overlap for the past K tracked frames $o_{\text{mean}} = \text{mean}\big(\{o(t_k, \mathcal{D}_k)\}_{k=1}^K\big)$ as another metric to make the decision. Finally, we define the reward function in a tracked state s with feature representation $\phi_{\text{Tracked}}(s) = (e_{\text{medFB}}, o_{\text{mean}})$ as

$$R_{\text{Tracked}}(s,a) = \begin{cases} y(a), & \text{if } e_{\text{medFB}} < e_0 \text{ and } o_{\text{mean}} > o_0 \\ -y(a), & \text{otherwise,} \end{cases} \tag{2}$$

where e_0 and o_0 are specified thresholds, $y(a) = +1$ if action $a = a_3$, and $y(a) = -1$ if $a = a_4$ in Fig. 2. So the MDP keeps the target in a tracked state if e_{medFB} is smaller but o_{mean} is larger than certain thresholds respectively. Otherwise, the target is transfered to a lost state.

Template Updating. The appearance model of the target needs to be updated in order to accommodate the appearance change. Online tracking methods [3, 15, 16, 5] update the appearance model whenever the tracker tracks the target. As a result, they are likely to accumulate tracking errors during the update, and drift from the target. In our MDP, we adopt a "lazy" updating rule and resort to the object detector in preventing tracking drift. Specifically, the template used in tracking remains unchanged if it is able to track the target. Whenever the template fails to track the target due to appearance change, the MDP transfers the target into a lost state. The "tracking" template is replaced by the associated detection when the target transitions from lost to tracked (Sec. 3.2.3). Meanwhile, we store K templates as the history of the target being tracked. The "tracking" template is one of the K templates, but may not be the latest one due to our "lazy" updating rule. These K templates are used for data association in lost states. So we do not accumulate tracking errors, but reply on the data association to handle the appearance change and continue the tracking.

3.2.3 Policy in a Lost State

In a Lost state, the MDP needs to decide whether to keep the target as lost, transition it to a tracked state, or mark it as inactive. We simply mark a lost target as inactive and terminate the tracking if the target has been lost for more than T_{Lost} frames. The challenging case is to make the decision between tracking the target and keeping it as lost. We treat it as a data association problem: in order to transfer a

lost target into a tracked state, the target needs to be associated with one of the detections from the object detector, otherwise, the target is kept as lost.

Data Association. Let t denote a lost target, and d be an object detection. Our goal is to predict the label $y \in \{+1, -1\}$ of the pair (t, d) indicating that the target is linked ($y = +1$) or not linked ($y = -1$) to the detection. We perform the binary classification using a real-valued linear function $f(t, d) = \mathbf{w}^T \phi(t, d) + b$, where $(\mathbf{w}, b)$ are the parameters that control the function, and $\phi(t, d)$ is the feature vector which captures the similarity between the target and the detection. The decision rule is given by $y = +1$ if $f(t, d) \geq 0$, otherwise $y = -1$. Consequently, the reward function for data association in a lost state s with feature representation $\phi_{\text{Lost}}(s) = \{\phi(t, d_k)\}_{k=1}^{M}$ is defined as

$$R_{\text{Lost}}(s, a) = y(a)\Big(\max_{k=1}^{M} \big(\mathbf{w}^T \phi(t, d_k) + b\big)\Big), \quad (3)$$

where $y(a) = +1$ if action $a = a_6$, $y(a) = -1$ if $a = a_5$ in Fig. 2, and k indexes M potential detections for association. The task of policy learning in the lost state reduces to learning the parameters $(\mathbf{w}, b)$ in the decision function.

Reinforcement Learning. We train the binary classifier with reinforcement learning in our MDP. Let $\mathcal{V} = \{v_i\}_{i=1}^{N}$ denote a set of video sequences for training, where N is the number of sequences. Suppose there are N_i ground truth targets $\mathcal{T}_i = \{t_{ij}\}_{j=1}^{N_i}$ in video v_i. Our goal is training the MDP to successfully track all these targets. We start training with an initial weights $(\mathbf{w}_0, b_0)$ and an empty training set $\mathcal{S}_0 = \emptyset$ for the binary classifier. Note that whenever the weights of the binary classifier are specified, we have a complete policy for the MDP which takes the action maximizing the reward in a given state. So the training algorithm loops over all the videos and all the targets, follows the current policy of the MDP to track the targets. The binary classifier or the policy is updated only when the MDP makes a mistake in data association. In this case, the MDP takes a different action as indicated by the ground truth trajectory. Suppose the MDP is tracking the jth target t_{ij} in video v_i, and on the lth frame of the video, the MDP is in a lost state. Let's consider two types of mistakes that can happen. i) The MDP associates the target t_{ij}^l to an object detection d_k which is wrong according to the ground truth, i.e., the target is incorrectly associated to a detection. Then $\phi(t_{ij}^l, d_k)$ is added to the training set $\mathcal{S}$ of the binary classifier as a negative example. ii) The MDP decides to not associate the target to any detection, but the target is visible and correctly detected by a detection d_k according to the ground truth, i.e., the MDP missed the correct association. Then $\phi(t_{ij}^l, d_k)$ is added to the training set as a positive example. After the training set has been augmented, we update the binary classifier by re-training it on the new training set. Specifically, given the current training set $\mathcal{S} = \{(\phi(t_k, d_k), y_k)\}_{k=1}^{M}$, we solve the following soft-margin optimization problem to obtain a max-margin classifier for data association:

$$\min_{\mathbf{w}, b, \boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{k=1}^{M} \xi_k$$
$$\text{s.t. } y_k\big(\mathbf{w}^T \phi(t_k, d_k) + b\big) \geq 1 - \xi_k, \xi_k \geq 0, \forall k, \quad (4)$$

where $\xi_k, k = 1, \ldots, M$ are the slack variables, and C is a regularization parameter. Once the classifier has been updated, we obtain a new policy which is used in the next iteration of the training process. We keep iterating and updating the policy until all the targets are successfully tracked. Algorithm 1 summarizes the policy learning algorithm.

input : Video sequences $\mathcal{V} = \{v_i\}_{i=1}^{N}$, ground truth trajectories $\mathcal{T}_i = \{t_{ij}\}_{j=1}^{N_i}$ and object detection $\mathcal{D}_i = \{d_{ij}\}_{j=1}^{N_i'}$ for video $v_i, i = 1, \ldots, N$
output: Binary classifier $(\mathbf{w}, b)$ for data association

Initialization: $\mathbf{w} \leftarrow \mathbf{w}_0$, $b \leftarrow b_0$, $\mathcal{S} \leftarrow \emptyset$
repeat
  foreach *video* v_i *in* $\mathcal{V}$ **do**
    foreach *target* t_{ij} *in* v_i **do**
      Initialize the MDP in Active ;
      $l \leftarrow$ index of the 1st frame t_{ij} correctly detected ;
      Transfer the MDP to Tracked, and initial the target template ;
      while $l \leq$ *index of last frame of* t_{ij} **do**
        Follow the current policy and choose an action a ;
        Compute the action a_{gt} indicated by the ground truth ;
        if *Current state is Lost and* $a \neq a_{gt}$ **then**
          Decide the label y_k of the pair (t_{ij}^l, d_k) ;
          $\mathcal{S} \leftarrow \mathcal{S} \cup \{(\phi(t_{ij}^l, d_k), y_k)\}$;
          $(\mathbf{w}, b) \leftarrow$ solution of Eq. (4) on $\mathcal{S}$;
          break ;
        else
          Execute action a ;
          $l \leftarrow l + 1$;
        end
      end
      if $l >$ *index of last frame of* t_{ij} **then**
        Mark target t_{ij} as successfully tracked;
      end
    end
  end
until *all targets are successfully tracked*;

Algorithm 1: Reinforcement learning of the binary classifier for data association

Feature Representation. One advantage of our reinforcement learning algorithm is that it is general and enables us to design and utilize features which are based on the status and the history of the target. We describe our design of the feature vector $\phi(t, d)$ which encodes the similarity between a target t and a detection d. First of all, the history of the target is represented by K templates in the past K video frames when the target is being tracked before it transfers to the lost state. Second, given the object detection d, we compute optical flow from each template to the detection in the same way as described in Sec. 3.2.2 but constrain the destination of the optical flow inside a neighborhood around the bounding box of the detection. Then we measure the quality of the optical flow in different aspects and use these metrics

Type	Notation	Feature Description
FB error	$\phi_1, \cdots, \phi_5$	Mean of the median forward-backward errors from the entire, left half, right half, upper half and lower half of the templates in optical flow
NCC	ϕ_6	Mean of the median Normalized Correlation Coefficients (NCC) between image patches around the matched points in optical flow
	ϕ_7	Mean of the NCC between image patches of the detection and the predicted bounding boxes from optical flow
Height ratio	ϕ_8	Mean of the ratios in bounding box height between the detection and the predicted bounding boxes from optical flow
	ϕ_9	Ratio in bounding box height between the target and the detection
Overlap	ϕ_{10}	Mean of the bounding box overlaps between the detection and the predicted bounding boxes from optical flow
Score	ϕ_{11}	Normalized detection score
Distance	ϕ_{12}	Euclidean distance between the centers of the target and the detection after motion prediction of the target with a linear velocity model

Table 1. Our feature representation for data association.

as features. Finally, we add features based on the similarity between the bounding boxes of the target and the detection. Table 1 summaries our feature representation.

3.3. Multi-Object Tracking with MDPs

After learning the policy/reward of the MDP, we apply it to the multi-object tracking problem. We dedicate a MDP for each object, and the MDP follows the learned policy to track the object. Given a new input video frame, targets in tracked states are processed first to determine whether they should stay as tracked or transfer to lost states. Then we compute pairwise similarity between lost targets and object detections which are not covered by the tracked targets, where non-maximum suppression based on bounding box overlap is employed to suppress covered detections, and the similarity score is computed by the binary classifier for data association. After that, the similarity scores are used in the Hungarian algorithm [29] to obtain the assignment between detections and lost targets. According to the assignment, lost targets which are linked to some object detections are transferred to tracked states. Otherwise, they stay as lost. Finally, we initialize a MDP for each object detection which is not covered by any tracked target. Algorithm 2 describes our multi-object tracking algorithm using MDPs in detail. Note that, tracked targets have higher priority than lost targets in tracking, and detections covered by tracked targets are suppressed to reduce ambiguities in data association.

input : A video sequence v and object detection $\mathcal{D} = \{d_k\}_{k=1}^{N}$ for v, binary classifier $(\mathbf{w}, b)$ for data association
output: Trajectories of targets $\mathcal{T} = \{t_i\}_{i=1}^{M}$ in the video

Initialization: $\mathcal{T} \leftarrow \emptyset$;
foreach *video frame* l *in* v **do**
  `// process targets in tracked states`
  foreach *tracked target* t_i *in* $\mathcal{T}$ **do**
    Follow the policy, move the MDP of t_i to the next state ;
  end
  `// process targets in lost states`
  foreach *lost target* t_i *in* $\mathcal{T}$ **do**
    foreach *detection* d_k *not covered by any tracked target* **do**
      Compute $f(t_i, d_k) = \mathbf{w}^T \phi(t_i, d_k) + b$;
    end
  end
  Data association with Hungarian algorithm for the lost targets ;
  foreach *lost target* t_i *in* $\mathcal{T}$ **do**
    Follow the assignment, move the MDP of t_i to the next state ;
  end
  `// initialize new targets`
  foreach *detection* d_k *not covered by any tracked target in* $\mathcal{T}$ **do**
    Initialize a MDP for a new target t with detection d_k ;
    if *action* a_1 *is taken following the policy* **then**
      Transfer t to the tracked state ;
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{t\}$;
    else
      Transfer t to the inactive state ;
    end
  end
end

Algorithm 2: Multi-Object Tracking with MDPs

4. Experiments

Datasets. We test our tracking framework on the recently introduced Multiple Object Tracking Benchmark [24] for people tracking. The MOT Benchmark collects widely used video sequences in the MOT community and some new challenging sequences. These sequences are divided into a training set and a test set each with 11 sequences. Since the annotations of the test set are not released, we separate a validation set of 6 sequences from the 11 training sequences to conduct analysis about our framework. The training and testing splitting for validation and testing is shown in Table 2. Except for AVG-TownCentre in the test set, for each of the other test sequences, there are training sequences which are captured in similar scenario indicated by the naming of the sequences. This property enables us to learn meaningful characteristics from training sequences and use them for testing . The MOT benchmark also provides object detections from the ACF detector [13]. By using the same object detection, we can make a fair comparison between different tracking methods.

Evaluation Metrics. We use multiple metrics to evaluate the multiple object tracking performance as suggested by the MOT Benchmark. These include Multiple Object Tracking Accuracy (MOTA) [18], Multiple Object Tracking Precision (MOTP) [18], Mostly Track targets (MT, percentage of ground truth objects who trajectories are covered by the tracking output for at least 80%), Mostly Lost targets (ML, percentage of ground truth objects who trajectories are covered by the tracking output less than 20%), the total number of False Positives (FP), the total number of False Negatives (FN), the total number of ID Switches (IDS), the total number of times a trajectory is Fragmented (Frag), and the number of frameworks processed in one second (Hz).

4.1. Analysis on Validation Set

Impact of the History. We first investigate the effect of the number of templates used in a lost state for data as-

Training	Testing
Validation on MOT Benchmark	
TUD-Stadtmitte	TUD-Campus
ETH-Bahnhof	ETH-Sunnyday, ETH-Pedcross2
ADL-Rundle-6	ADL-Rundle-8, Venice-2
KITTI-13	KITTI-17
Testing on MOT Benchmark	
TUD-Stadtmitte, TUD-Campus	TUD-Crossing
PETS09-S2L1	PETS09-S2L2, AVG-TownCentre
ETH-Bahnhof, ETH-Sunnyday, ETH-Pedcross2	ETH-Jelmoli, ETH-Linthescher, ETH-Crossing
ADL-Rundle-6, ADL-Rundle-8	ADL-Rundle-1, ADL-Rundle-3
KITTI-13, KITTI-17	KITTI-16, KITTI-19
Venice-2	Venice-1

Table 2. Training and Testing sequences for validation and testing on the MOT Benchmark.

K	MOTA	MOTP	MT	ML	FP	FN	IDS	Frag
1	24.7	73.2	10.3	55.1	3,597	13,651	147	303
2	25.7	73.5	9.8	53.4	3,548	13,485	121	349
3	23.0	73.6	8.5	56.0	3,727	13,907	134	325
4	26.3	**73.9**	9.8	53.8	3,191	13,726	**91**	300
5	**26.7**	73.7	**12.0**	53.0	3,386	**13,415**	111	331
6	19.5	73.7	5.6	68.8	3,393	14,920	269	321
7	26.1	73.6	10.7	55.6	3,092	13,838	132	306
8	25.8	73.8	10.7	55.6	3,221	13,785	122	305
9	**26.7**	73.6	**12.0**	**51.7**	3,290	13,491	133	328
10	26.6	73.8	9.8	55.1	**2,691**	14,130	123	**276**
11	25.3	73.5	**12.0**	52.1	3,672	13,436	136	317
12	24.8	73.4	11.5	55.6	3,637	13,585	139	321

Table 3. Tracking performance in terms of the number of templates on the validation set.

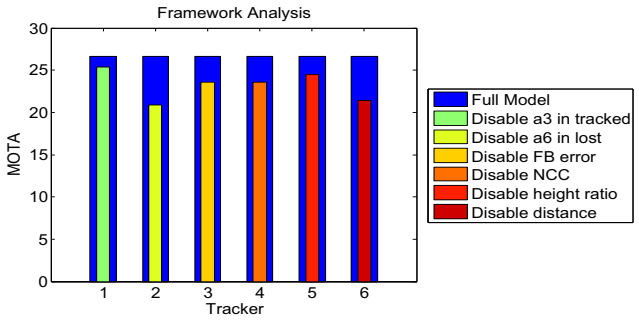

Figure 4. Analysis of our framework on the validation set by disabling different components.

sociation (Sec. 3.2.3). Intuitively, the more templates we use, the longer history of the target is captured. Table 3 shows the tracking performance in terms of the number of templates on the validation set, where we accumulate the statistics across all the 6 testing sequences for evaluation. From the table, we observe two peaks for the tracking performance. One is around using 5 templates, and the other is around using 9 templates, which demonstrates that using multiple templates to capture the history of the object is helpful. With 9 templates, we see significant improvements in terms of mostly tracked (MT) and mostly lost (ML). This indicates that the tracker is able to generate long tracks to cover the target, which in turn reflects that the data association is more effective.

Contribution of Different Components. We investigate the contribution of different components in our framework by disabling a component at one time and then examining the performance drop in terms of MOTA on the validation set (Fig. 4). 1) We disable action a_3 in tracked states (Fig. 2). Then the template tracking is disabled and a tracked target directly transfers to a lost state. We do not see significant performance drop in this case, since the framework can still rely on data association in lost states to continue tracking. Template tracking is helpful when the detector misses the target. 2) We disable action a_6 in lost states (Fig. 2), i.e., data association for lost targets is disabled. In this case, we see a significant loss in performance in Fig. 4. Especially, ID switches are more than 3 times compared to the full model. Data association is a crucial component in our framework. 3-6) Finally, we investigate the contribution of different features used in data association (Table 1). Fig. 4 shows the performance drop by disabling FB error in optical flow ($\phi_1, \cdots, \phi_5$), Normalized Correlation Coefficient (NCC, ϕ_6 and ϕ_7), ratio between the heights of bounding box (ϕ_8 and ϕ_9), and distance between the target and the detection (ϕ_{12}) respectively. As we can see, the four types of features all contribute, and distance is relatively more important than other features. In addition, we do not see performance drop by disabling bounding box overlap (ϕ_{10}) and detection score (ϕ_{11}) on the validation set.

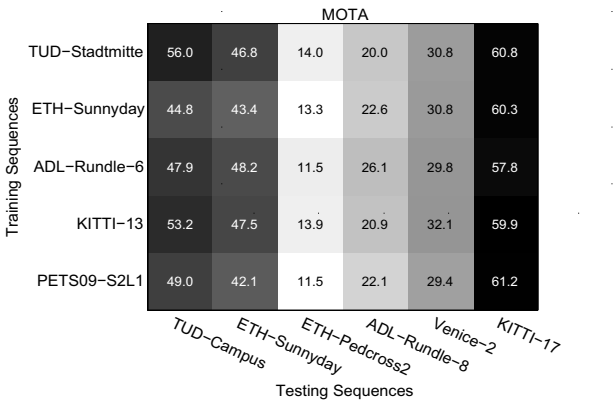

Figure 5. Tracking performance in MOTA with different pairs of training and testing sequences.

Cross-domain Tracking. In order to test the generalization power of our method, we also conduct experiments by testing the trained tracker in different scenarios. The results are presented in Fig. 5. First, we can see from the table that performing training and testing in similar scenarios is beneficial. For example, the tracker trained on ADL-Rundle-6 achieves the best performance on ADL-Rundle-8. Second, trackers trained on the five training sequences perform reasonably well on all the test sequences. In some cases, cross-domain testing even improves the results. For instance, on the test sequence KITTI-17, the tracker trained on PETS09-S2L1 achieves better performance than the one trained on KITTI-13. Recall that our features used in data

Tracker	Tracking Mode	Learning Mode	MOTA	MOTP	MT	ML	FP	FN	IDS	Frag	Hz
DP_NMS [36]	Batch	N/A	14.5	70.8	6.0%	40.8%	13,171	34,814	4,537	3,090	**444.8**
TC_ODAL [4]	Online	Online	15.1	70.5	3.2%	55.8%	12,970	38,538	**637**	1,716	1.7
TBD [14]	Batch	Offline	15.9	70.9	6.4%	47.9%	14,943	34,777	1,939	1,963	0.7
SMOT [12]	Batch	N/A	18.2	71.2	2.8%	54.8%	8,780	40,310	1,148	2,132	2.7
RMOT [42]	Online	N/A	18.6	69.6	5.3%	53.3%	12,473	36,835	684	1,282	7.9
CEM [27]	Batch	N/A	19.3	70.7	8.5%	46.5%	14,180	34,591	813	1,023	1.1
SegTrack [26]	Batch	Offline	22.5	**71.7**	5.8%	63.9%	**7,890**	39,020	697	**737**	0.2
MotiCon [23]	Batch	Offline	23.1	70.9	4.7%	52.0%	10,404	35,844	1,018	1,061	1.4
MDP OFL (Ours)	Online	Offline	30.1	71.6	10.4%	41.3%	8,789	33,479	690	1,301	0.8
MDP REL (Ours)	Online	Online	**30.3**	71.3	**13.0%**	**38.4%**	9,717	**32,422**	680	1,500	1.1

Table 4. Tracking performance on the test set of the MOT Benchmark. More comparisons are available at [2].

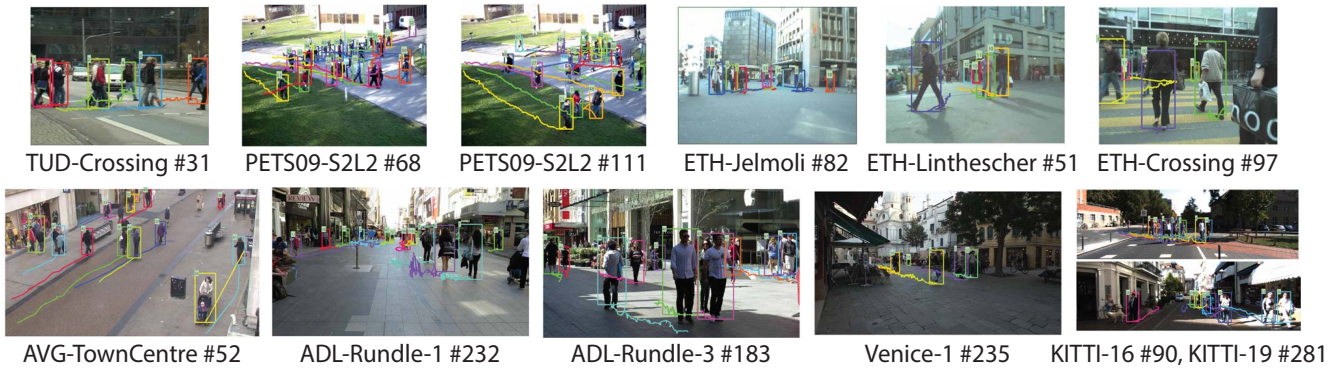

Figure 6. Tracking results on the test sequences in the MOT benchmark.

association are similarity metrics between targets and detections, which are not designed for specific scenarios. As a result, our method learns the similarity function which can be generalized across different sequences.

4.2. Evaluation on Test Set

After the analysis on the validation set, we perform training with all the training sequences, and test the trained trackers on the test set according to Table 2, where we use 10 templates in data association. We submitted our results to the MOT Benchmark website [2] for evaluation. Table 4 shows our tracking performance on the test set, where we compare our tracker (MDP REinforcement Learning, MDP REL) with the state-of-the-art methods tested on the MOT benchmark. As we can see from the table, our tracker improves 7% in MOTA compared with the second best published tracker, and achieves the best performance in terms of mostly tracked and mostly lost targets even though it works in the online mode. The superior performance demonstrates the advantages of our learning to track strategy with MDPs. Fig. 6 shows sampled tracking results on the 11 sequences in the test set (see [1] for the technical report with evaluation on individual test sequences and the tracking videos).

We also evaluated a variation of our tracking method (MDP OFfline Learning, MDP OFL), where we construct training examples to learn the similarity function offline as in the traditional way. In order to use the same features as in MDP REL, we link true positive detections to form trajectory of the target using the ground truth annotations. Positive (Negative) examples are pairs of target and detection that should (not) be linked between adjacent video frames. We collect 45,005 examples to learn 6 similarity functions according to Table 2, and use them in our MDP framework for testing. As we can see in Table 4, MDP OFL also achieves very competitive performance compared to other methods, which verifies the robustness of our tracking framework. More importantly, MDP REL achieves better performance than offline training by using 1,397 training examples only in our experiments. With 3% of the training data as in offline learning but achieving similar or even better performance, we demonstrate the benefit of our reinforcement learning algorithm for multiple object tracking.

5. Conclusion

We have proposed a novel online multi-object tracking framework based on Markov decision processes, where the lifetime of an object is modeled with a MDP with four subspaces of states (Active, Tracked, Lost and Inactive). The state transitions in the MDP naturally handle the birth/death and appearance/disappearance of objects in tracking. A similarity function for data association is learned as part of the MDP policy with reinforcement learning. Our framework is general to be integrated with different techniques in object detection, single object tracking and data association by using them for MDP policy learning. We have tested our implementation of the tracking framework on the challenging MOT Benchmark, which outperforms the state-of-the-art methods tested on the benchmark by notable margins.

Acknowledgments. We acknowledge the support of DARPA UPSIDE grant A13-0895-S002. We thank David Held and Christopher B. Choy for helpful discussions.

References

[1] MDP Tracking. http://cvgl.stanford.edu/projects/MDP_tracking. 8

[2] Multiple object tracking benchmark. http://motchallenge.net. 8

[3] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *TPAMI*, 33(8):1619–1632, 2011. 2, 3, 4

[4] S.-H. Bae and K.-J. Yoon. Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning. In *CVPR*, pages 1218–1225, 2014. 1, 2, 8

[5] C. Bao, Y. Wu, H. Ling, and H. Ji. Real time robust l1 tracker using accelerated proximal gradient approach. In *CVPR*, pages 1830–1837, 2012. 2, 3, 4

[6] R. Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957. 2

[7] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *TPAMI*, 33(9):1806–1819, 2011. 1, 2

[8] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992. 3

[9] J.-Y. Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5:1–10, 2001. 4

[10] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Online multiperson tracking-by-detection from a single, uncalibrated camera. *TPAMI*, 33(9):1820–1833, 2011. 2

[11] 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, 2

[12] C. Dicle, O. I. Camps, and M. Sznaier. The way they move: Tracking multiple targets with similar appearance. In *ICCV*, pages 2304–2311, 2013. 8

[13] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *TPAMI*, 36(8):1532–1545, 2014. 6

[14] 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

[15] S. Hare, A. Saffari, and P. H. Torr. Struck: Structured output tracking with kernels. In *ICCV*, pages 263–270, 2011. 2, 3, 4

[16] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *TPAMI*, 34(7):1409–1422, 2012. 2, 3, 4

[17] S. Karayev, M. Fritz, and T. Darrell. Anytime recognition of objects and scenes. In *CVPR*, pages 572–579, 2014. 2

[18] B. Keni and S. Rainer. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1:1–1:10, 2008. 6

[19] Z. Khan, T. Balch, and F. Dellaert. Mcmc-based particle filtering for tracking a variable number of interacting targets. *TPAMI*, 27(11):1805–1819, 2005. 2

[20] S. Kim, S. Kwak, J. Feyereisl, and B. Han. Online multi-target tracking by large margin structured learning. In *ACCV*, pages 98–111. 2012. 1, 2

[21] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *ECCV*, pages 201–214. 2012. 2

[22] C.-H. Kuo, C. Huang, and R. Nevatia. Multi-target tracking by online learned discriminative appearance models. In *CVPR*, pages 685–692, 2010. 1, 2

[23] L. Leal-Taixé, M. Fenzi, A. Kuznetsova, B. Rosenhahn, and S. Savarese. Learning an image-based motion context for multiple people tracking. In *CVPR*, pages 3542–3549, 2014. 1, 8

[24] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking. *arXiv:1504.01942 [cs]*, 2015. 1, 2, 6

[25] Y. Li, C. Huang, and R. Nevatia. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *CVPR*, pages 2953–2960, 2009. 1, 2

[26] A. Milan, L. Leal-Taixé, K. Schindler, and I. Reid. Joint tracking and segmentation of multiple targets. In *CVPR*, pages 5397–5406, 2015. 8

[27] A. Milan, S. Roth, and K. Schindler. Continuous energy minimization for multitarget tracking. *TPAMI*, 36(1):58–72, 2014. 1, 2, 8

[28] 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

[29] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial & Applied Mathematics*, 5(1):32–38, 1957. 2, 6

[30] A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *ICML*, pages 663–670, 2000. 3

[31] J. C. Niebles, B. Han, and L. Fei-Fei. Efficient extraction of human motion volumes by tracking. In *CVPR*, pages 655–662, 2010. 2

[32] S. Oh, S. Russell, and S. Sastry. Markov chain monte carlo data association for multi-target tracking. *IEEE Transactions on Automatic Control*, 54(3):481–497, 2009. 2

[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] S. Oron, A. Bar-Hillel, and S. Avidan. Extended lucas kanade tracking. In *ECCV*, pages 142–156, 2014. 2

[35] L. Paletta, G. Fritz, and C. Seifert. Q-learning of sequential attention for visual object recognition from informative local descriptors. In *ICML*, pages 649–656, 2005. 2

[36] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR*, pages 1201–1208, 2011. 2, 8

[37] O. Russakovsky, L.-J. Li, and L. Fei-Fei. Best of both worlds: human-machine collaboration for object annotation. In *CVPR*, pages 2121–2131, 2015. 2

[38] X. Song, J. Cui, H. Zha, and H. Zhao. Vision-based multiple interacting targets tracking via on-line supervised learning. In *ECCV*, pages 642–655. 2008. 1, 2

[39] J. S. Supancic III and D. Ramanan. Self-paced learning for long-term tracking. In *CVPR*, pages 2379–2386, 2013. 2

[40] Y. Xiang, C. Song, R. Mottaghi, and S. Savarese. Monocular multiview object tracking with 3d aspect parts. In *ECCV*, pages 220–235. 2014. 2

[41] R. Yao, Q. Shi, C. Shen, Y. Zhang, and A. van den Hengel. Part-based visual tracking with online latent structural learning. In *CVPR*, pages 2363–2370, 2013. 2

[42] J. H. Yoon, M.-H. Yang, J. Lim, and K.-J. Yoon. Bayesian multi-object tracking using motion context from multiple objects. In *WACV*, pages 33–40, 2015. 8

[43] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *CVPR*, pages 1–8, 2008. 2