Unsupervised Learning based Jump-Diffusion Process for Object Tracking in Video Surveillance

Xiaobai Liu[1,2], **Donovan Lo**[1], **Chau Thuan**[1]
[1] San Diego State University, San Diego, CA
[2] XreLab Inc., San Diego, CA
xiaobai.liu@sdsu.edu

Abstract

This paper presents a principled way for dealing with occlusions in visual tracking which is a long-standing issue in computer vision but largely remains unsolved. As the major innovation, we develop a unsupervised learning based jump-diffusion process to jointly track object locations and estimate their visibility statuses over time. Our method employs in particular a set of jump dynamics to change object's visibility statuses and a set of diffusion dynamics to track objects in videos. Different from the traditional jump-diffusion process that stochastically generates dynamics, we utilize deep policy functions to determine the best dynamic at the present step and learn the optimal policies from raw videos using reinforcement learning methods. Our method is capable of tracking objects with severe occlusions in crowded scenes and thus recovers the complete trajectories of objects that undergo multiple interactions with others. We evaluate the proposed method on challenging video sequences and compare it to other methods. Significant improvements are obtained for the videos including frequent interactions.

1 Introduction

Stochastic jump-diffusion process was first studied by Grenander and Miller [Grenander and Miller, 1994] in 1994 and a rigorous account for the reversibility was given by Green [Green, 1995] in 1995. The process is often used to sample a probabilistic distribution defined over a mixture of multiple subspaces. It employs both jump and diffusion dynamics to reconfigure the present sample, and simulates a Markov Chain towards the target distribution. A jump is used to move between different subspaces whereas a diffusion is used to move within the same subspace. While this method is barely used in computer vision community, the seminal works by Song-Chun Zhu et al. [Han *et al.*, 2004] showed that it can serve as a unified inference framework for solving multiple computer vision tasks, e.g., image segmentation, skeleton detection etc. In this work, we revisit jump-diffusion process and study how to apply it to solve the fundamental challenges of visual object tracking.

A long standing challenge to visual tracking [Yilmaz *et al.*, 2006] is how to track objects that are partially visible due to occlusions. Classic solutions to this issue often employ a hypothesis-testing strategy to determine object visibility status: *assume* the object of interest is visible, *match* image of object (or object part) into the next video frame, and *check* if the matching score is higher than a threshold. This strategy has two fundamental shortcomings. First, the threshold might be varying across time, locations, objects, scene types, and it is very difficult to find the optimal one while tracking an object moving in different scenes. Second, these methods ignore the time-dependent relationships between different visibility statuses of the same object. To our best knowledge, there is no previous efforts on systematically studying the temporal transitions of object visibility status. In this work, we develop a unified framework to explicitly reason object visibility statuses while tracking objects in videos.

The key idea of our method is to formulate the task of object tracking as a Markov decision process in a joint continuous-discrete space. At each time-step, our method will estimate the visibility status of the target object and track its location and size in videos. The former task is defined in a discrete space, comprising of 'visible' and 'occluded', and the latter task is in the continuous image space, i.e. location or size. These two tasks should be coordinately formulated and scheduled in order to avoid errors in inference. In particular, when the object of interest becomes occluded, traditional trackers do not work well and we need to employ re-identification methods to group discrete trajectories.

We develop a policy-based jump-diffusion process [Han *et al.*, 2004] to seek for the optimal solution in the continuous-discrete space. The process employs a set of dynamics to reconfigure the present solution in order to simulate a Markov Chain. A dynamic is either a jump that changes object visibility status or a diffusion that changes object location/size. In the literature [Han *et al.*, 2004], the traditional jump-diffusion process is often driven by randomly proposed dynamics or data-driven dynamics, and is subject to slow convergence. It is almost impossible to apply it over real-time systems, e.g., tracking. In this work, we employ discriminatively trained policy functions to propose dynamics in an effective way. We also parameterize the policy functions using deep neural networks to ensure stabilities and convergences, and train policy networks using the policy gradient method in the reinforce-

ment learning setting. Evaluations on multiple video benchmarks showed that our method can reliably estimate object visibility status over time and robustly track objects of interest with occlusions in crowded scenarios.

The contributions of this work include (I) A policy-based jump-diffusion method that can simultaneously track objects and reason their visibility statuses in crowded scenes; (II) an unsupervised method for learning policy functions from raw videos without human efforts. These techniques can be applied to address the challenges of other compute vision tasks that span on a mixture of subspaces, e.g, image segmentation, activity recognition, etc.

2 Relationships to Previous Works

Object Tracking In the past decade, tracking-by-detection has become the mainstream framework [Ren *et al.*, 2015], mostly because of the technical advancements in object detection. Our approach follows this pipeline as well but focuses on the reasoning of object visibility status. A more recent advancement is to learn deep neural networks for tracking objects across video frames, e.g., MDNet [Nam and Han, 2016]. All these methods require a large amount of annotated videos for learning deep representations. While achieving impressive results, the quality of tracking outcomes are heavily dependent on the amount and quality of annotation data. In this work, we formulate visual tracking in the reinforcement learning setting and introduce an unsupervised learning method to train our tracker.

Occlusion Handling is perhaps the most fundamental problem in visual tracking, and has been extensively studied in the past literature. These methods can be roughly divided into three categories: i) using depth information [Ess *et al.*, 2009], ii) modeling partial occlusion relations [Tang *et al.*, 2014], iii) modeling object appearing and disappearing globally based on motion or other cues [Wang *et al.*, 2016]. These methods employ the hypothesis-test strategy to determine if occlusions happen. In contrast, this paper presents a principled way to explicitly reason object visibility status while tracking them in videos.

Deep Reinforcement Tracking Reinforcement learning methods [Sutton and Barto, 1998] aim to learn to sequentially choose actions that can maximize cumulative future rewards. Traditional reinforcement learning methods are limited to their poor performance and stability capabilities. The recent deep reinforcement learning (DRL) methods utilized deep representations to parameterize policy functions or value functions, and achieved encouraging successes in multiple fields, e.g., video games [Mnih *et al.*, 2013]. DRL was also applied to solve traditional computer vision problems.

In particular, Caicedo and Lazebnik [Caicedo and Lazebnik, 2015] employed DRL to narrow down the search areas while localizing objects, Yun et al. [Yoo *et al.*, 2017] extended this idea to track objects in videos. These methods achieve impressive results on scenarios without frequent object interactions but cannot directly deal with occlusion issues. In this work, we extend these works and introduce an unified tracking framework to explicitly reason object occlusion statuses.

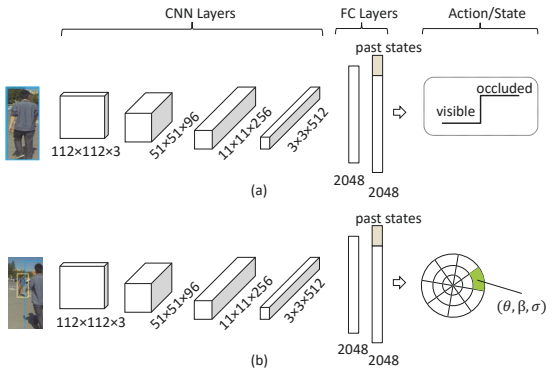

Figure 1: Deep policy networks. (a) Policy network used to switch the object visibility status from visible to occluded or vice versa. (b) Policy network used to shift or re-size object bounding boxes. Each continuous action is described as a triple (θ, β, σ) that represents the translation direction, translation distance and scale factors, respectively. The two networks share the same convolutional layers.

3 The Proposed Method

3.1 Policy-based Jump-Diffusion Tracker

We formulate object tracking as a stochastic jump-diffusion process. For a given video sequence, our method first detects objects of interest in videos. Then, for each object, our method randomly chooses a jump dynamic or a diffusion dynamic to change its visibility status or location/size. A critical problem for this jump-diffusion tracker is how to select an appropriate dynamic at each step. In this work, we consider the developed tracker as an agent and the input video sequence as the environment. The agent employs policy functions with the current state information to retrieve the optimal action, execute the action to get a new state, and receive a reward. The agent seeks for the optimal actions to change object visibility status or localize/resize the object while maximizing the accumulated future rewards.

We cast the selection of dynamics a Markov Chain Decision Process (MDP). A MDP is defined by a tuple $(s, a, \mathcal{T}, r)$, where $s \in S$ represents an object state, $a \in A$ denotes an action, $\mathcal{T}$ represents the transition between states, and $r(s, a)$ defines the immediate reward received after executing action a to state s.

State variables are used to characterize the current status of an object in the present environment. For each object, the state space comprises of two subspaces, $S = S_d \Cup S_c$, where S_d is composed of two discrete states: 'visible' and 'occluded', and S_c is composed of continuous object positions and sizes in video frames. Every state represents the information of the target object, in terms of appearances, locations, and history of states.

Actions and Transition Function can be performed to change the discrete or continuous object states. In the discrete space, an action represents a jump dynamic and is applied to change object's visibility status. In the continuous space, an action represents a diffusion dynamic and is applied to shift or resize the object bounding box. A continuous action can be denoted with a triplet (θ, β, σ), representing the transla-

tion and scaling changes over orientation, distance, and size. Figure 1 visualizes this vector in a log-polar coordination. This definition is different from the previous works [Yoo *et al.*, 2017; Caicedo and Lazebnik, 2015], which quantize the translation and scaling spaces into discrete bins and introduce a classification network to define the policy function. In this work, we argue that these dynamics should be defined in a 3-dimensional continuous space.

Reward We define the reward function $r(s, a)$ over the present state s, regardless of the actions a, i.e., $r(s, a) = r(s)$. The reward of an action is defined in either discrete or continuous space. In the discrete space, we apply an action to change the visibility status of the object of interest, and the reward function $r(s)$ returns 1 or 0 representing the prediction is correct or not. On the other hand, in the continuous space, we use an action to translate or re-size the bounding box of the target, and calculate the overlapping ratio, denoted as U, between the new bounding box and the expected box. We set the reward function $r(s)$ as:

$$r(s) = \begin{cases} +1, U > 0.7 \\ -1, U < 0.5 \qquad \forall s \in S_c \\ 0, \ Otherwise \end{cases} \tag{1}$$

In this work, we use the raw video sequences as inputs and there are no ground-truth object trajectories nor object visibility status. Instead, we employ on-the-shelf techniques to automatically generate the visibility statuses as well as short trajectories, as discussed later.

Policy Functions The policy function $\pi(a_t|s_t)$ in MDP is used to choose a proper action a_t given the state s_t at time t. The action can be chosen in either deterministic or stochastic manner. While it is possible for a policy function to take any form, the most recent studies [Yoo *et al.*, 2017] showed that it is beneficial to parameterize it using deep neural networks, which can significantly improve the stability of policy learning.

Heuristic Policies As an alternative to policy functions, the simplest approach to selecting actions might be pre-defined heuristics. In the discrete action space, for example, one can set an object to be visible if it can be detected with high confidences, or to be occluded if only object parts are detectable (with high confidences), or to be invisible if neither objects nor object parts are detectable. One can also evaluate the history of object movements, and set the visibility status of an object to be occluded if another object will walk in front of it. Similarly, in the continuous space, one can employ traditional trackers to determine the translations or scale changes of object bounding boxes. The above heuristics can be further enhanced through using the ground-truth data, instead of detectors or trackers. In this work, we use these heuristics to initialize policy training.

3.2 Architecture of Deep Policy Networks

We employ deep neural networks to parameterize the policy functions in either discrete or continuous spaces, which are used to choose the optimal action for the current state.

Discrete Policy Network Figure 1 (a) summarizes the sketch of the policy network used for selecting discrete actions. To determine the visibility status of a tracked object at time t, we use the bounding box at time $t-1$ to crop the image of object from the video frame at t, and fed it to a deep neural network. The network comprises of three convolution layers and two fully connected (fc) layers that are combined with the ReLu and dropout layers. The output of the second fc layer is concatenated with the vector of past visibility statuses which has 100 dimensions. The final layers are used to predict the probability of each visibility status. In particular, there are two output units, representing two visibility statuses: visible, and occluded, being combined with a softmax layer whose output is the conditional action probability distribution.

Continuous Policy Network We employ another network architecture to parameterize the policy network for selecting continuous actions, as shown in Figure 1 (b). The network shares the same convolutional layers with the discrete policy network, and includes two additional fully connected layers that are combined with the ReLu and dropout layers. The output of the second fc layer is concatenated with the list of past actions which has 100 dimensions. The final layer includes a single unit connected with a regression loss. We use the sum of square loss in this work. Note that this is different from the previous work [Yoo *et al.*, 2017] which employs classification loss to predict translation/scaling changes and ignores the continuous relationships between actions.

3.3 Reasoning Discrete and Continuous Rewards

In this work, we consider surveillance videos and employ on-the-shelf techniques to automatically generate labels needed for learning policy functions, including both object trajectories and object visibilities. This is feasible partially because we focus on the surveillance videos for which foreground regions can be readily extracted from raw videos. We also assume the background stuffs, e.g., tree, building, are automatically detected or manually labeled, but the studies over these topics already go beyond the scope of this work.

For a video sequence, we first detect objects (e.g., vehicle) and match these detections with each other to generate a set of episodes, where each episode is composed of multiple object boxes of the same object over time. We generate these episodes based on the outcomes of object re-identification [Xu *et al.*, 2013]. In particular, we detect and match object boxes over time, and randomly group matched boxes to form episodes of varying lengths. Let $E_k = (, \ldots, B_i, B_j, \ldots)$ denote an episode, B_i and B_j represents two objects at time i and j, respectively. Note that the time-steps i and j might not be consecutive. The continuous rewards are defined over E_k. For example, if $i = 5, j = 7$, the reward $s(a)$ at the time 6 is defined to be zero; and the reward at the time 5 and 7 is defined using Eq. (3.1). In practice, we only use the high-confidence outcomes of the re-identification method [Xu *et al.*, 2013], which help reduce false alarms or errors. We empirically found that even these flawed labels are informative enough and can significantly improve tracking accuracies.

Similarly, we employ the sampled episodes to reason ob-

Algorithm 1: Training of Policy Networks.

Input: a training episode
For every video frame t
- Use the policy networks to generate conditional probabilities for discrete actions and continuous actions, respectively
- Choose and execute an action with a probability
- Accumulate gradients ΔW according to Eq. (2)

If tracking the object successfully
- Update network parameters $W = W + \Delta W$

Otherwise, update network parameters $W = W - \Delta W$

ject visibility statuses over time. For every episode, the initial status of the object is set to be 'visible' for every time-step. For every two episodes that are temporally concurrent and spatially adjacent, we reason the 'occluded' boxes as follows. First, we perform for each episode bilinear interpolation between consecutive boxes to generate bounding boxes at each time-step during the course of the episode. Second, for object boxes that overlap with other boxes and their bottom line is above the other boxes, we set the their statuses to be 'occluded'. We will discuss implementation details in the section of experiments.

3.4 Learning Policy Networks

We formulate the learning of the proposed two policy networks in the reinforcement setting. During training, the agent (tracker) will receive a reward from the environment (the input video sequences) after executing an action at time t. The objective is to maximize the total rewards the agent can receive in the future. Note that the two policy networks are trained alternatively so that the developed jump-diffusion tracker can choose either continuous or discrete actions for the given state.

We use a variant of REINFORCE algorithm [Williams, 1992] with accumulated policy gradients to learn the parameters of the two policy networks. To do so, we randomly generate multiple tracking episodes with varying lengths from training videos such that each episode covers the lifespan (or partial lifespan) of an object of interest. Then, we perform object tracking on each training episode and draw actions as follows: first employs the current policy reworks to generate the conditional action probabilities in both discrete and continuous spaces; then stochastically selects one of the actions as the current decision. In this way, we can probabilistically generate action sequences from the episode. Once selected the action, being discrete or continuous, we consider it as the ground-truth label and run the back-propagation method to calculate gradients. Taking the discrete policy network for an example, let $\phi(\alpha_t|s_t; W)$ denote the policy function and W the network parameters. We can accumulate the gradients for all decisions in the same episode as:

$$\Delta W = c \sum_{t=1}^{T} \Delta \log \pi(\alpha_t|s_t; W)\gamma^{T-t} \quad (2)$$

where c is the learning rate, Δ represents the derivative with respect to the network parameters W, T is the length of the episode and $\gamma \in (0, 1]$ is a discounting factor that assigns higher weight to decisions made later during the course of the episode. Similar gradient calculations are applied for the continuous policy network. We use ΔW to update network parameters when our tracker successfully localize the object by the end of the episode. Otherwise, if our tracker fails to localize the object, we use negative gradients, i.e. $-\Delta W$, to update the network parameters.

Algorithm 1 summarizes the sketch of the proposed training algorithm. It is noteworthy that such a reinforcement learning algorithm does not need to access the annotations (i.e. object boxes or visibility status) at every time-step. Instead, it only requires disjoint matched boxes (i.e. episode) to specify the rewards. In this way, high-confidence outcomes of the object re-ID module are sufficient to learning decent policy networks as shown in the experiment part. For each object of interest, our tracker utilizes the two policy networks to choose discrete actions or continuous actions to reconfigure the current state.

4 Experiments

4.1 Datasets, Implementation, and Metrics

We use three video datasets to test and evaluate the proposed method.

Tracking Interacting Objects (TIO) dataset. We collect a new video dataset to justify the effectiveness of the proposed method. The videos are captured in a parking-lot and a plaza, and include multiple human-object interactions, e.g., loading, unloading, stopping etc. The object of interest include person and vehicle. In contrast, most existing tracking benchmarks, e.g., PETS09 [Ferryman and Shahrokni, 2009], OTB [Wu *et al.*, 2015], KITTI dataset [Geiger *et al.*, 2012], do not include frequent occlusions. All these sequences are captured by a GoPro camera, with frame rate 30fps and resolution 1920×1080. There exist severe occlusions and large scale changes, making this dataset very challenging for traditional tracking methods.

People-Car dataset [Wang *et al.*, 2016][1]. This dataset consists of video sequences on a parking lot with two synchronized bird-view cameras, with length varying between 300 and 5100 frames. In this dataset, there are many instances of people getting in and out of cars. This dataset is challenging for the frequent interactions, light variation and low object resolution.

PPL-DA dataset. We collect another video dataset that covers daily human activities in 3 public facilities: court/garden, office reception, and plaza. The scenes are recorded with GoPro cameras, mounted on around 1.5 meters high tripods. The produced videos are also around 4 minutes long and in 1080P quality.

Beside the above testing data, we collect a set of video clips for model training. To avoid over-fitting, we set up different camera positions, different people and vehicles from the testing settings. The training data consists of 380 video clips, covering multiple events, e.g., opening vehicle door, entering

[1] cvlab.epfl.ch/research/surv/interacting-objects

Figure 2: Exemplar results of visibility status estimation on **TIO dataset**. t: index of video frames.

Scene	ULJDTracker	Baseline
Plaza	0.75	0.61
Parking-lot	0.71	0.54
Average	0.73	0.57

Table 1: Results of visibility status recognition (accuracy) on the **TIO** dataset. ULJDTracker: the proposed method; Baseline: the heuristic method generating object visibility statuses

vehicles, loading baggage etc. Both the datasets and short clips are annotated with the bounding boxes for people, suitcase, and vehicles. We use these annotations to train various baseline trackers as introduced later.

Implementation We pre-train the policy networks on the ImageNet dataset [Deng *et al.*, 2009], and then fine-tune the network parameters on the generated training episodes. We resize all images to be 112×112 pixels before feeding into the policy networks. We use dropout regularization for fully-connected layers, with drop rate 0.7. Each convolution layer is followed by the rectified linear unit (ReLU) activation function. To train the network, we set the learning rate c to be 0.0001, $\gamma = 0.95$. We use the pre-trained vehicle and object detectors [Ren *et al.*, 2015] and call the re-ID method [Xu *et al.*, 2013] to match object detections with each other. We randomly generate 8000 episodes from the training dataset, with length varying from 30 to 300 frames. We consider a predicted object box to be correct if its overlapping ratio with the ground box is at least 0.5, and consider an episode to be successfully tracked if at last 40% predicted bounding box are correct. At the the early stage of training, we use the heuristic policies, instead of the policy networks, to choose actions in order to accelerate the exploration stage. We use experience replay method [Schaul *et al.*, 2015] during training and retain in the replay memory 5000 successful samples and 5000 failure samples. To update the network parameters, we sample 50 samples from the memory and accumulate the gradients. We set an object to be 'invisible' if it is labeled as 'occluded' for more than 20 frames. We implement the proposed tracker using MatConvNet toolbox and run all experiments on a workstation with CPU: Intel Core i7-7700K, GPU: Nvidia GeForce GTX 1050, and Memory: 8GB. Without code optimization, the proposed tracker can run about 35 fps.

Metrics We evaluate the proposed method from two aspects. *First*, the ability to track objects. We adopt the widely used CLEAR metrics [Kasturi *et al.*, 2009] to measure the performances of tracking methods. It includes four metrics, i.e., Multiple Object Detection Accuracy (MODA), Detection Precision (MODP), Multiple Object Tracking Accuracy (MOTA) and Tracking Precision (MOTP). We also report the number of false positives (FP), false negatives (FN), identity switches (IDS) and fragments (Frag). A higher value means better for TA and TP while a lower value means better for FP, FN, IDS and Frag. If the Intersection-over-Union (IoU) ratio of tracking results to ground truth is above 0.5, we accept the tracking result as a correct hit. The other metrics used include **MT**, *mostly tracked*, percentage of ground truth trajectories which are covered by tracker output for more than 80% in length; and **ML**$^{\downarrow}$, *mostly lost*, percentage of ground-truth trajectories which are covered by tracker output for less than 20% in length. *Second*, the ability to estimate object visibility status. We simply compare the prediction to the ground-truth labels and calculate the percentage of correctness, i.e. accuracy.

Plaza	MOTA ↑	MOTP ↑	FP ↓	FN ↓	IDS ↓	Frag ↓
ULJDTracker	**48.7%**	**76.5%**	11	**368**	**1**	4
MHT_D [Kim *et al.*, 2015]	34.3%	73.8%	56	661	15	18
MDP [Xiang *et al.*, 2015]	32.9%	73.2%	24	656	9	7
DCEM [Milan *et al.*, 2016]	32.3%	76.5%	**2**	675	2	**2**
SSP [Pirsiavash *et al.*, 2011]	31.7%	72.1%	19	678	21	25
DCO [Andriyenko *et al.*, 2012]	29.5%	76.4%	22	673	6	2
JPDA_m [Hamid Rezatofighi *et al.*, 2015]	13.5%	72.2%	163	673	6	3
ParkingLot	MOTA ↑	MOTP ↑	FP ↓	FN ↓	IDS ↓	Frag ↓
ULJDTracker	**38.6%**	**79.3%**	**339**	**1637**	**10**	**7**
MDP [Xiang *et al.*, 2015]	30.1%	76.4%	397	2296	26	22
DCEM [Milan *et al.*, 2016]	29.4%	77.5%	383	2346	16	15
SSP [Pirsiavash *et al.*, 2011]	28.9%	75.0%	416	2337	12	14
MHT_D [Kim *et al.*, 2015]	25.6%	75.7%	720	2170	15	12
DCO [Andriyenko *et al.*, 2012]	24.3%	78.1%	536	2367	38	10
JPDA_m [Hamid Rezatofighi *et al.*, 2015]	12.3%	74.2%	1173	2263	28	17

Table 2: Quantitative tracking results on the **TIO dataset**. The best scores are marked in **bold**.

People-Car	Method	MODA ↑	FP ↓	FN ↓	IDS ↓
Seq.0	ULJDTracker	**0.70**	**0.04**	**0.11**	**0.01**
	TIF-MIP [Wang *et al.*, 2016]	0.67	0.07	0.25	0.04
	KSP [Berclaz *et al.*, 2011]	0.49	0.10	0.41	0.07
	LP2d [Leal-Taixé *et al.*, 2014]	0.47	0.05	0.48	0.06
	POM [Fleuret *et al.*, 2008]	0.47	0.06	0.47	-
	SSP [Pirsiavash *et al.*, 2011]	0.20	0.04	0.76	0.04
Seq.1	ULJDTracker	**0.63**	**0.15**	**0.16**	**0.03**
	TIF-MIP [Wang *et al.*, 2016]	0.58	0.17	0.25	0.04
	KSP [Berclaz *et al.*, 2011]	0.04	0.71	0.25	0.12
	LP2D [Leal-Taixé *et al.*, 2014]	0.02	0.77	0.21	0.17
	POM [Fleuret *et al.*, 2008]	-0.21	0.98	0.23	-
	SSP [Pirsiavash *et al.*, 2011]	0.00	0.75	0.25	0.12

Table 3: Quantitative tracking results on the **People-Car** dataset. The best scores are marked in **bold**.

4.2 Results for Visibility Status Estimation

We apply the proposed method **ULJDTracker** over the TIO dataset and test its ability to estimate object visibility status. Figure 2 visualizes the results on a video sequence. There are three pedestrians walking in a parking-lot and a vehicle moving in the nearby area. Taking the pedestrian#1 for an instance, his visibility status changes from 'visible' (t=1, t=60) to 'occluded' (t=120) as he approaches and gets into the vehicle. Our method can correctly estimate the changes of visibility and enable high-level video understanding, e.g., activity recognition. Table 1 reports the quantitative results of visibility status estimation while applying the proposed tracker over

the TIO videos. We also include the results of the method used in Section 3.3. These comparisons demonstrate that the proposed method can significantly improve the accuracy of visibility status estimation.

4.3 Results for Object Tracking

For the **TIO** dataset, we compare the proposed method (**ULJDTracker**) with 6 popular trackers [Pirsiavash *et al.*, 2011] [Kim *et al.*, 2015] [Xiang *et al.*, 2015] [Milan *et al.*, 2016] [Andriyenko *et al.*, 2012] [Hamid Rezatofighi *et al.*, 2015]. We use the public implementations of these methods. Table 2 reports the quantitative results and comparisons on TIO dataset. From the results, we can observe that our method achieved encouraging performance while using most metrics. In particular, the IDS of our method is much lower than any other methods. This indicates that the ability of occlusion reasoning can improve the robustness of visual tracking systems when there are frequent object interactions.

For the **People-Car** dataset, we compare the proposed method with other 5 trackers [Pirsiavash *et al.*, 2011] [Berclaz *et al.*, 2011] [Fleuret *et al.*, 2008] [Leal-Taixé *et al.*, 2014] [Wang *et al.*, 2016]. The quantitative results are reported in Table 3. Results show that our method obtains better performance than other methods.

PPL-DA dataset We compare the proposed method to four popular trackers [Fleuret *et al.*, 2008] [Berclaz *et al.*, 2011] [Xu *et al.*, 2016]. We include the results of the recent neural network based method, MDNet [Nam and Han, 2016], which employs a Multi-Domain Convolutional Neural Network for visual tracking and achieves state-of-the-art tracking performance in multiple visual tracking benchmarks [Nam and Han, 2016]. We also include the tracker **ActionNet** proposed by Yoo et al. [Yoo *et al.*, 2017]. ActionNet employs reinforcement learning techniques to learn to shift object boxes and can be considered as a supervised variant of the proposed method. Both MDNet and ActionNet are trained on the training videos and annotations. Results show that the proposed method is more effective these two state-of-the-art methods although they all employ reinforcement learning techniques. These comparisons demonstrate the superiority of jump-diffusion process.

Table 4 reports the quantitative results of various methods on PPL-DA dataset. Notably, our method can significantly reduce the number of ID switches (IDS) on all scenarios, which is a critical indicator of the superiority of our method. Our method also outperforms ActionNet, which clearly demonstrates the superiority of the proposed policy-based jump-diffusion process.

Seq-Court	TA(%)	TP(%)	MT(%)	ML(%) ↓	IDSW ↓	FRG ↓
ULJDTracker	**53.5**	**82.7**	**33.9**	**18.1**	**19**	58
MDNet [Nam and Han, 2016]	53.1	82.1	32.2	21.3	53	**42**
ActionNet [Yoo *et al.*, 2017]	45.6	78.1	28.7	26.8	61	51
HTC [Xu *et al.*, 2016]	29.5	71.9	14.8	25.9	91	77
KSP [Berclaz *et al.*, 2011]	24.7	64.4	0.00	44.4	318	291
POM [Fleuret *et al.*, 2008]	22.3	65.4	0.00	51.9	296	269
Seq-Office	TA(%)	TP(%)	MT(%)	ML(%) ↓	IDSW ↓	FRG ↓
ULJDTracker	**63.1**	**88.7**	**58.2**	0.00	**23**	**21**
MDNet [Nam and Han, 2016]	60.3	87.1	54.1	0.00	33	28
ActionNet [Yoo *et al.*, 2017]	57.3	78.5	55.3	0.00	34	32
HTC [Xu *et al.*, 2016]	41.2	70.7	28.6	0.00	66	59
KSP [Berclaz *et al.*, 2011]	39.6	58.0	28.6	0.00	83	76
POM [Fleuret *et al.*, 2008]	36.9	58.8	28.6	0.00	89	82
Seq-Plaza	TA(%)	TP(%)	MT(%)	ML(%) ↓	IDSW ↓	FRG ↓
UULJDTracker	**42.6**	**71.3**	**34.8**	**12.1**	**69**	**87**
MDNet [Nam and Han, 2016]	27.4	68.9	18.5	12.7	112	98
ActionNet [Yoo *et al.*, 2017]	32.5	65.8	29.5	21.5	99	103
HTC [Xu *et al.*, 2016]	23.1	66.2	11.6	18.6	202	178
KSP [Berclaz *et al.*, 2011]	17.3	57.5	7.0	27.9	356	311
POM [Fleuret *et al.*, 2008]	16.7	57.9	4.6	32.6	339	295

Table 4: Quantitative tracking results on the **PPL-DA** dataset that includes three video sequences.

5 Conclusions

This paper revisited the classical jump-diffusion process and studied a novel way to make it computationally feasible for dealing with real-time problems. We focus on visual tracking in this work, and study in particular how to jointly track objects and reason their visibility statues in videos. Our method employs two different dynamics to reconfigure the current solution in both discrete and continuous space, and chooses the optimal dynamic at each step using policy functions that are trained from raw videos. The developed techniques can be applied to solve other video-related tasks, e.g., activity recognition, which will be studied in the future.

Acknowledgments

Xiaobai Liu is supported by the DARPA SIMPLEX program (No. 58723A), National Science Foundation (No. 1657600) and ONR grant (No. N00014-17-1-2867).

References

[Andriyenko *et al.*, 2012] Anton Andriyenko, Konrad Schindler, and Stefan Roth. Discrete-continuous optimization for multi-target tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[Berclaz *et al.*, 2011] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE transactions on pattern analysis and machine intelligence*, 33(9):1806–1819, 2011.

[Caicedo and Lazebnik, 2015] Juan C Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2488–2496, 2015.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[Ess *et al.*, 2009] Andreas Ess, Konrad Schindler, Bastian Leibe, and Luc Van Gool. Improved multi-person tracking with active occlusion handling. In *IEEE ICRA Workshop*, 2009.

[Ferryman and Shahrokni, 2009] James T Ferryman and Ali Shahrokni. Pets2009: Dataset and challenge. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2009.

[Fleuret *et al.*, 2008] Francois Fleuret, Jerome Berclaz, Richard Lengagne, and Pascal Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):267–282, 2008.

[Geiger *et al.*, 2012] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[Green, 1995] Peter J Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

[Grenander and Miller, 1994] Ulf Grenander and Michael I Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 549–603, 1994.

[Hamid Rezatofighi *et al.*, 2015] Seyed Hamid Rezatofighi, Anton Milan, Zhen Zhang, Qinfeng Shi, Anthony Dick, and Ian Reid. Joint probabilistic data association revisited. In *IEEE International Conference on Computer Vision*, 2015.

[Han *et al.*, 2004] Feng Han, Zhuowen Tu, and Song-Chun Zhu. Range image segmentation by an effective jump-diffusion method. *IEEE Transactions on pattern analysis and machine intelligence*, 26(9):1138–1153, 2004.

[Kasturi *et al.*, 2009] Rangachar Kasturi, Dmitry Goldgof, Padmanabhan Soundararajan, Vasant Manohar, John Garofolo, Rachel Bowers, Matthew Boonstra, Valentina Korzhova, and Jing Zhang. Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):319–336, 2009.

[Kim *et al.*, 2015] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M Rehg. Multiple hypothesis tracking revisited. In *IEEE International Conference on Computer Vision*, 2015.

[Leal-Taixé *et al.*, 2014] Laura Leal-Taixé, Michele Fenzi, Alina Kuznetsova, Bodo Rosenhahn, and Silvio Savarese. Learning an image-based motion context for multiple people tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[Milan *et al.*, 2016] Anton Milan, Konrad Schindler, and Stefan Roth. Multi-target tracking by discrete-continuous energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):2054–2068, 2016.

[Mnih *et al.*, 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[Nam and Han, 2016] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4293–4302, 2016.

[Pirsiavash *et al.*, 2011] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[Ren *et al.*, 2015] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Conference on Neural Information Processing Systems*, 2015.

[Schaul *et al.*, 2015] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[Sutton and Barto, 1998] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[Tang *et al.*, 2014] Siyu Tang, Mykhaylo Andriluka, and Bernt Schiele. Detection and tracking of occluded people. *International Journal of Computer Vision*, 110(1):58–69, 2014.

[Wang *et al.*, 2016] Xinchao Wang, Engin Turetken, Francois Fleuret, and Pascal Fua. Tracking interacting objects using intertwined flows. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 38(11):2312–2326, 2016.

[Williams, 1992] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[Wu *et al.*, 2015] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.

[Xiang *et al.*, 2015] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *IEEE International Conference on Computer Vision*, 2015.

[Xu *et al.*, 2013] Yuanlu Xu, Liang Lin, Wei-Shi Zheng, and Xiaobai Liu. Human re-identification by matching compositional template with cluster sampling. In *proceedings of the IEEE International Conference on Computer Vision*, pages 3152–3159, 2013.

[Xu *et al.*, 2016] Yuanlu Xu, Xiaobai Liu, Yang Liu, and Song-Chun Zhu. Multi-view people tracking via hierarchical trajectory composition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[Yilmaz *et al.*, 2006] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13, 2006.

[Yoo *et al.*, 2017] Sangdoo Yoo, Jongwon Yun, Youngjoon Choi, Kimin Yun, and Jin Young Choi. Action-decision networks for visual tracking with deep reinforcement learning. 2017.