Intelligent Camera Selection Decisions for Target Tracking in a Camera Network

Anil Sharma, Saket Anand and Sanjit K Kaul
Indraprastha Institute of Information Technology (IIIT), Delhi
{anils, anands, skkaul}@iiitd.ac.in

Abstract

Camera Selection Decisions (CSD) are highly useful for several applications in a multi-camera network. For example, CSD benefit multi-camera target tracking by reducing the number of candidate cameras to look for the target's next location. The correct candidate cameras, decreases the number of false Re-ID queries as well as the computation time. Also, in multi-camera trajectory forecasting (MCTF) to predict where a person will re-appear in the camera network along with the transition time. These applications require a large amount of annotated data for training. In this paper, we use state-representation learning with a reinforcement learning based policy to effectively and efficiently make camera selection decisions. We further demonstrate that by using learned state representations, as opposed to hand-crafted state variables, we are able to achieve state-of-the-art results on camera selection, while reducing the training time for the RL policy. Along with this, we use a reward function that helps to reduce the amount of supervision in training the policy in a semi-supervised way. We report our results on four datasets: NLPR_MCT, DukeMTMC, CityFlow, and WNMF dataset. We show that an RL policy reduces unnecessary Re-ID queries and therefore the false alarms, scales well to larger camera networks, and is target-agnostic.

1. Introduction

Camera networks are pervasive and frequently used for various visual analytics applications like video surveillance, crowd behavior analysis, etc. Target tracking is a crucial task for these applications that aims to determine the position of a target at all times across the different cameras of the camera network. The number of cameras at an airport, train station, malls, etc. has rapidly increased, which makes automated tracking an essential task for visual analytics. Existing methods [30, 23] focus on executing single camera tracking and re-identification to locate the target in the camera network. This requires to make a large number of queries to the camera network and a critical drawback is large computational cost and degradation in performance due to false alarms in re-identifying the target. Another approach [41] is to predict the next camera so that search space can be reduced. However, such an approach doesn't incorporate the indefinite transition time of the target between cameras. In many scenarios, the camera network topology is not known, and the tracking algorithm should be able to track the target in the absence of this knowledge. For this, camera selection decisions [33] are shown to be an effective approach for enabling efficient tracking in a camera network. In this paper, we leverage state representation learning (SRL) to encode the state's history which helps to learn a reinforcement learning based policy that achieves better camera selection performance even on the larger camera networks.

Re-identification (Re-ID) and data-association are conventional ways [30, 20] used to associate individual tracklets from different cameras to form the multi-camera trajectory of a particular target. The indeterminate and unknown transition time of the target between two field-of-views (FOVs) makes this association problem very challenging. Longer transition times result in more uncertainty about the target's location, necessitating more Re-ID queries and thereby increasing the number of false alarms. These false alarms are severely detrimental as they lead to incorrect target association resulting in tracking an irrelevant target. On the other hand, a false negative from a Re-ID algorithm in a camera frame may not be detrimental so long as the target is re-identified in one of the subsequent frames of the camera. Therefore, to deal with longer transition times, one could learn to decide at every time step whether to make a Re-ID query or not, and if the former, which camera feed(s) to query. Such an *intelligent camera selection* strategy has been shown that reducing redundant querying can benefit the multi-camera tracking performance [23, 19, 33]. We investigate intelligent camera selection and focus on tackling the problem of camera-handovers[1], as we scale to larger camera networks.

Many approaches for multi-camera target tracking em-

[1]The words camera-transition and camera-handover will be used interchangeably.

ploy a two-step framework [23, 45, 30, 47]. First, SCT (Single-Camera Tracking) to find the target's trajectory within each camera. Second, ICT (Inter-Camera Tracking), to associate the SCT trajectories corresponding to the same identity across camera after the target has transitioned from one camera's FOV to another. Previous approaches have modeled the inter-camera transition time using static distributions like Gaussian [23] or Parzen window based non-parametric distributions [19]. However, various factors like target speed, congestion, etc., may influence the transition time and make their distribution time-dependent. In [33], the authors deal with this time-dependence by modeling a Markov Decision Process (MDP) for intelligent camera selection which uses hand picked state variables to learn a policy using RL. The learned policy selects cameras for Re-ID queries, and adds the selected camera in a history variable until the event of the target being found, upon which the history variable is reset.

Recently, in [34], the authors pointed out the limitation of using the exact approach of [33] and use deep-Q learning (DQN [26]) to alleviate the challenges with larger camera networks. We observe that in the absence of knowledge of the camera topology, the camera history is an important state variable. It holds information about the sequence of previously queried cameras, which influences the decision of which camera to select for the next query. In this paper, we argue that hand-crafted state variables are not representative enough and hinder the scalability of such an approach. Therefore, we instead propose a state representation learning [14] based approach and modify the state-vector accordingly. A representation helps to learn the variations and the history of observations in a low dimension vector and hence creates a generic state vector. Our final state vector leverages an LSTM-based autoencoder (AE) to summarize the camera history of previously queried cameras. We further show various advantages of using a learned state representation, including generalization across camera-network datasets, accommodating a generic DQN architecture across datasets (unlike [34]), and most importantly reduced training speeds.

Our specific contributions are summarized below:

1. We propose a novel method for camera selection decision using state representation learning (SRL). We use an LSTM based autoencoder (AE) for latent representation of the history vector of cameras. We will show empirically that these representations, as opposed to hand-crafted state variables, achieve state-of-the-art results and train faster.

2. Our reward function helps to reduce the amount of supervision in training the policy. We will show that it achieves comparable performance with the policy trained in a fully supervised manner.

3. The extensive experiments show that the proposed method is superior than most state-of-the-art methods on several real datasets and it is target agnostic. We will also show that it benefits two real applications, multi-target multi-camera (MTMC) tracking, and multi-camera trajectory forecasting (MCTF) in a camera network.

4. We demonstrate the camera selection performance on four real datasets, NLPR MCT dataset [8], Duke MTMC dataset [30], WNMF dataset [41], and CityFlow dataset [43, 27].

2. Related Works

Multi-camera tracking is looked from various viewpoint in both overlapping and non-overlapping cameras. Works such as [18, 51, 21, 1, 3] assumed overlapping camera field-of-views (FOVs). These require camera calibration and knowledge of camera network topology to obtain the 3D coordinates. But tracking in non-overlapping cameras is more challenging because non-overlapping cameras are require to handle the blind spot areas between cameras.

To resolve camera handovers in non-overlapping FOVs, a few initial works have created a social group model [50] to associate target tracklets, affinity model [22] of target's appearance for inter-camera association. Other works formulate various data association methods [25, 9, 12] to resolve camera handovers and use graph [50, 45, 5, 16, 17, 46, 24] based approaches for inter-camera tracking. Spatio-temporal contextual information [47], clique based methods [29, 28], part based model [40, 2] are also a few other common approaches. Many work perform pairwise matching [4, 11, 13, 15, 31, 42] of the templates to form trajectories. Template re-identification [48, 42] approaches are leading for matching target's template with other candidate templates. In this regard, works [19, 25] use the travel time of the target to estimate the transition time of the camera handover. Works [23] have estimated a transition time distribution using a Gaussian distribution. In comparison to these works, we propose a reinforcement learning based policy that selects a camera index where the target is likely to reappear. This policy handles the indefinite transition time and also reduces the number of search queries.

Recent works for multi-camera target tracking perform tracking task in a two step framework. First, they perform single camera tracking (SCT) and then inter-camera tracking (ICT) to resolve the camera handover separately. Works such as [23, 45, 47, 10, 30, 44] use such a two step framework for tracking in multiple cameras. [23] has proposed an online method using sophisticated features of human appearance along with segmentation using change point detection. To perform ICT, they form a camera link model and estimate the travel time using a Gaussian distribution. Other common approaches estimate entry-exit points across cameras [25, 47] and some predict the future trajectory of the target [41, 35, 37]. Current state-of-the-art in appearance features is in deep learning based methods to re-identify a target [48, 30, 20] including deep feature representation learning, deep metric learning and ranking optimization. [30] have proposed a weighted triplet loss to learn better features of target's appearance. However, their approach makes a very large number of Re-ID queries to the camera network and fewer false alarms are crucial for an automated system [39, 36]. Works in [33, 34, 38, 32] have shown that camera selections are crucial for efficient target tracking. In this paper, we use state representation learning with RL that achieves state-of-the-art results for camera selections. We will show how RL can be used to train the policy in a semi-supervised manner.

3. Proposed Method

In this section, we will explain the formulation of camera selection decisions using Markov Decision Process (MDP), the neural network model for camera selection policy and its training.

3.1. Camera Selection as an MDP

We formulate the camera selections as an MDP (Markov Decision Process) which is defined as a tuple of elements $(S, \mathcal{A}, f, R, \gamma)$, where S is the state space, $\mathcal{A}$ is the action space, $f(s_t, s_{t+1})$ is the state transition function, $R(s, a)$ is the reward function and γ is the discount factor. We model the camera selection problem as a finite horizon discounted sum reward problem.

The individual elements of the MDP are described below:

State: In a camera network, we have access to the initial location (bounding box) of the target in a given camera frame [49, 34]. The location of the target is represented as (c, b), where c is the camera index (encoded by a one-hot vector) and b is the bounding box (represented as $[x, y, w, h]^\top$). To include the direction of motion of the target, we include the deltas of the bounding box ($\Delta b_t = b_t - b_{t-1}$) in the state vector. To handle inter-camera transitions of the target, we include a time-progress variable τ that monitors the timesteps elapsed since the last time the target was observed. Additionally, we also maintain a history h_t of past actions (camera selections) as part of the state vector. The final state is given by the set $s_t = (c, b, \Delta b, \tau, h_t)$. It is worth emphasizing that the camera history in its raw form is a sequence of one-hot encoded vectors representing the cameras queried, so we use an autoencoder model to learn latent embeddings that are fixed-length representations for this state variable. These representations capture the variations in the environment [14] and hence help to achieve better performance.

Action: The action space is encoded as $\mathcal{A} = \{0, 1, \ldots, N-1, C_\times\}$, where N is the number of cameras and an action $C_\times$ is included as a 'null camera', suggesting that the target is making an handover and is not visible in the camera network. The policy selects an action $C_\times$ to indicate that no Re-ID queries need to be made.

Reward: We define a reward function for each state action pair.

$$r_{t+1}(s_t, a_t) = \begin{cases} +1 & a_t = y_t \;\&\; \tau > 20 \\ +0.5 & a_t = y_t \;\&\; \tau \leq 20 \\ 0.01 & a_t = y_t = C_\times \\ -1 & \text{otherwise} \end{cases} \tag{1}$$

a_t is the action taken and y_t is the ground truth camera. τ is the transition time and it is thresholded to distinguish the occlusions and the camera handovers. We found from NLPR and Duke datasets that the occlusions were not more than 2 sec so we threshold this at 20 frames. The camera handovers lasts for more than 20 frames which are few in the whole trajectory, hence a higher reward is provided. A smaller reward value is given for the action $C_\times$, which happens to be the most frequent action for ICT.

State transition function: With s_t as the state at time t, the policy selects an action $a_t \in \mathcal{A}$. The next state is updated based on the camera selection. If the target is found at the selected camera, then the location (c, b) is updated. If not, then τ is incremented and the last policy decision is appended to the history h_t. The latent representation of h_t is used in the state vector.

Q-learning: In reinforcement learning, an agent interacts with its environment by executing an action $a_t \in \mathcal{A}$ at time t by which the environment transitions into the next state s_{t+1} and provides a reward r_{t+1} to the agent. We use Q-value function $Q(s_t, a_t)$ which is the expected discounted sum reward which the agent receives starting from state s_t and taking action a_t at time t. The optimal Q-values $Q^*(s_t, a_t)$ are defined when an optimal policy π^* is followed. Our state-space is continuous and huge and hence we learn a parameterized Q-values $Q^*(s, a|\theta)$ using a neural network.

The optimal Q-values are learned by iteratively updating the parameters θ using deep Q-learning [26]. An optimal policy utilizes these Q-values to select an optimal action given the target current state as:

$$\pi_t^*(s_t) = arg \max_a Q^*(s_t, a) \tag{2}$$

3.2. System Architecture

The proposed architecture is shown in figure 1. The architecture consists of three important parts. First, the autoencoder based learned state representation vector, obtained by encoding the action history into a single fixed length vector. Second, the neural network based policy function,

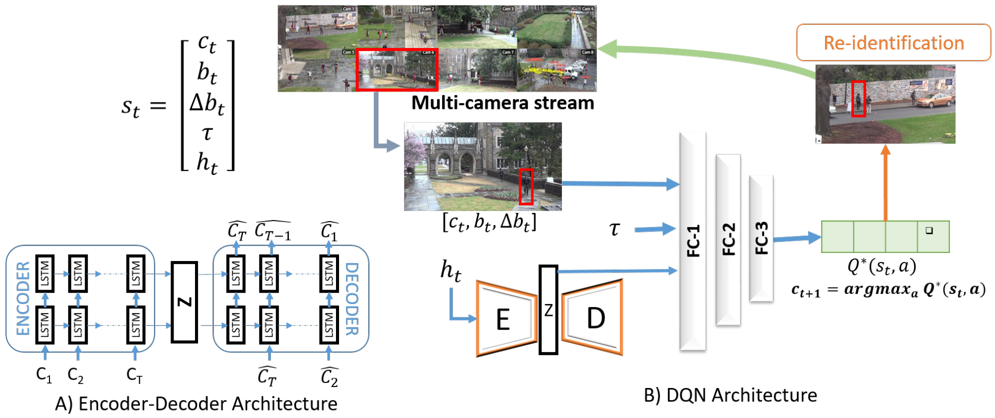

Figure 1. The DQN architecture used to learn the camera selection policy. It shows the neural network model that learns the policy which takes as input the different state variables and the LSTM based Autoencoder (E-Encoder, D-Decoder) which encodes the action history (h_t) in a fixed length latent representation (Z).

which learns to select a camera given the initial location of the target. Third, the re-identification (Re-ID) algorithm which utilizes the policy-based camera selection to find whether the target is present in the selected camera frame. For this, we extracted target features from existing Re-ID algorithm [6] and then a matching is established with candidate targets using cosine similarity. The threshold and other parameters are explained in the results sec 4.3.1.

Learned State Representation: The state variable capturing the past action history of the policy is an important part of our model. The history encodes previous camera selection decisions and it influences the decision of which camera to select for the next query. A long history is necessary to make well-informed camera selection decisions. History vector in its raw form (sequence of one-hot encoded vectors) has high cardinality and state vector becomes very large for larger camera networks. Therefore, we make use of an LSTM-based autoencoder to learn a fixed-vector representation for this sequence of past actions.

The AE structure is shown in figure 1, and it is trained using the cross-entropy loss. The input given is an action sequence $a_{1:T} = c_1, c_2, \ldots, c_T$, where each c_i is a one-hot vector. The latent vector is the encoder's last layer cell state at time T. The latent vector preserves the information of the sequence, which is used to reconstruct the input using the decoder. We need not retrain or fine-tune the AE for another dataset, as we observed during our experiments that the AE generalizes well across different datasets. The only requirement is that the number of cameras at test time N_{test} should be smaller than that at train time N_{train}. We can then encode the past action history by zero-padding the one-hot encoded vector to make it of size N_{train}. Ablation study on LSTM parameters is shown in supplementary doc.

Camera Selection Policy Model: The architecture in figure 1 shows the neural network model which represents the policy for camera selections. The neural network model contains three fully-connected layers with size (2048,1024,256) and ReLu activation function. The output layer is equal to the number of actions $(N + 1)$ with linear activation to represent the Q-value function $Q(s_t, a), \forall a \in \mathcal{A}$. We use the MSE loss and *Adam* optimizer to learn the optimal weights for the policy using deep-Q learning. The action history is represented by the auto-encoder which is *trained separately* from the policy network.

We used the epsilon-greedy exploration strategy [26] during training and the policy is learned using deep Q-learning with experience replay (ER). ER is a technique to store the previous experiences of the policy to prevent catastrophic forgetting in the neural networks. For back-propagation, these experience are sampled from the replay buffer to create a minibatch. The minibatch should be diverse enough to contain experiences of different situations to learn an optimal policy. In case of camera selections, we observed that the experiences having the action $C_\times$ were frequent which creates an imbalanced minibatch and hence biases the policy to the most frequently occurring action. To create a diverse minibatch, we segregated the replay buffer into three buffers, first, to store the experiences for most frequent action $C_\times$, second, for the experiences ended in positive reward and third, the experiences ended in a negative reward. Then we sampled the experiences *uniformly* from all replay buffers to create a minibatch of all possible experiences. Let the minibatch be $\mathcal{B} = (s_t, a, s_{t+1}, r_{t+1})$ which is used to generate an empirical estimate of the expected loss $L(\theta_t)$, shown in eqn. 3

$$L(\theta_t) = \frac{1}{|\mathcal{B}|}\sum_{i=0}^{|\mathcal{B}|}[(r_{t+1} + \gamma max_a Q(s_{t+1}, a)) - Q(s_{t+1}, a|\theta_t)]^2 \tag{3}$$

where $(r_{t+1} + \gamma max_a Q(s_{t+1}, a))$ is the target and $Q(s_{t+1}, a)$ is the output of the neural network.This error

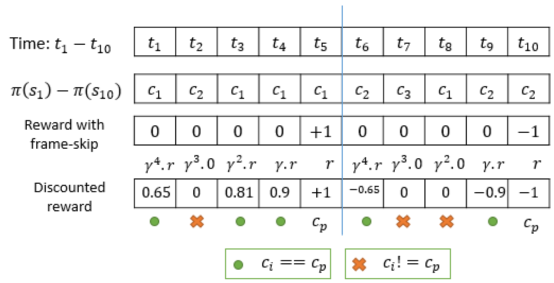

Figure 2. Figure shows the modification in the reward function (equation 1) for semi-supervised training.

is also referred to as TD (temporal-difference) error and is minimized by backpropagation to learn the optimal weights of the policy network.

Semi-supervised Training To train the neural network model without full supervision, the reward function in equation 1 is modified keeping other variables same for training. The reward is given after skipping a few frames. For example, if the reward is given after n frames then the discounted reward is accumulated to the previous $n-1$ frames. The discounted reward is computed using discount factor $\gamma = 0.9$. The figure 2 shows the reward given to the neural network during training phase. The figure shows 10 steps of training phase and cameras selected by the policy in the second row. The third row shows the reward given after every 5 frames ($n = 5$) using reward function defined in equation 1. The last row shows the reward is discounted for all previous frames where a reward is not given. At any time step t_i, a discounted reward ($\gamma^{n-i}.r$) is given if policy selects same camera as the camera where reward (r) is given otherwise a 0 reward is given. The impact of number of frame-skip is shown in the next section. A Re-ID method is also not used during training when a reward is not given, where a bounding box is picked using intersection-over-union (IOU). A bounding box in the current frame which has 0.6 or more IOU with the previous frame's bounding box is selected for that frame.

4. Results

We will now describe the experimental setup, performance evaluation for camera selection and target tracking.

4.1. Experimental Setup

Datasets: We have used NLPR_MCT data set [23], DukeMTMC [28], CityFlow [43, 27], and WNMF [41] dataset to evaluate the proposed method for camera selections. These datasets are detailed in the table 1. The CityFlow dataset has multiple scenarios, we select two large scenarios (scenario 4 having 25 cameras and scenario 5 having 19 cameras).. All datasets were used at 10 FPS and WNMF at 5 FPS. The training and testing splits and evaluation experiments are taken from the state-of-the-art methods [10, 45, 23, 33, 34] and we show comparison with various methods through these experiments.

Table 1. Details of the datasets used for training and performance evaluation. The table shows the number of cameras (#Cameras), duration of the videos, frame rate (FPS), the number of targets (#Target) captured in each dataset.

	#Cam	Duration	FPS	#Target
NLPR-Set1	3	20 min	20	235
NLPR-Set2	3	20 min	20	255
NLPR-Set3	4	3.5 min	25	14
NLPR-Set4	5	24 min	25	49
DukeMTMC	8	1hr 25min	60	2834
CityFlow S04	25	17.97 mins	10	71
CityFlow S05	19	2hr 3mins	10	337
WNMF	15	600 hrs	5	-

Performance metric: We evaluate camera selection, inter-camera tracking, and multi-camera multi-target tracking performance separately. To evaluate camera selection performance, we use precision (P) and F1 scores [33]. To evaluate the inter-camera tracking and multi-camera tracking performance, we use commonly used Multi-Camera Tracking Accuracy (MCTA) metric [23]. Readers are requested to see [28, 23] for details about the MCTA metric.

To quantify the computational performance, we use number of frames polled (F metric) [34]. For inter-camera tracking (ICT), we define the measure Percentage Camera Handover (PCH) as the percentage of target transitions (from Camera C_i to C_j, $i \neq j$) that are correctly detected by using the learned policy. This is included because missing more target transitions hurts overall tracking performance and hence PCH should be higher for better performance.

4.2. Camera Selection Decisions

4.2.1 Impact of State-Representation on Performance

In this experiment, we study the state-representation in detail by observing the impact of sequence length on the camera selection performance. For this, we generate all possible sequences of length $10, 20$, and 50 on various datasets to train the LSTM based encoder-decoder. We observed that AE trained on the sequences of larger dataset (CityFlow with 40 cameras in total) can also encode the sequences of the smaller datasets (DukeMTMC and NLPR). To use AE on smaller datasets, zeros are padded to the one-hot vector of smaller dataset representing a camera index.

Table 3 shows the impact of sequence length on the camera selection performance on NLPR-Set4. We quantify the impact in terms of PCH metric on the testing set and the number of episodes required to train the policy. In RL, all states between initial and terminal state is one episode. For example, one game of chess. We created multiple configu-

Table 2. Camera Selection performance of our proposed method and its comparison with state-of-the-art approaches using precision (P) and F1-score (F1) metrics. CityFlow-S05 and S04 are scenario-5 and scenario-4 of the CityFlow dataset. *Ours-SS* is our method with semi-supervised (SS) training.

	NLPR Set-1		NLPR Set-2		NLPR Set-3		NLPR Set-4		Duke MTMC		CityFlow-S05		CityFlow-S04	
	P	$F1$	P	$F1$	P	$F1$	P	$F1$	P	$F1$	P	$F1$	P	$F1$
Exhaustive	0.24	0.37	0.22	0.34	0.10	0.19	0.11	0.19	0.042	0.11	0.05	0.10	0.01	0.03
Neighbor	0.36	0.37	0.32	0.34	0.14	0.25	0.18	0.29	0.042	0.33	0.32	0.49	0.22	0.36
CamSel [33]	0.95	0.86	0.94	0.82	0.64	0.72	0.61	0.66	Out of Memory		Out of Memory		Out of Memory	
nStep [34]	0.76	0.75	0.69	0.81	0.60	0.70	0.73	0.78	0.49	0.55	0.40	0.40	0.45	0.48
Ours	0.92	0.92	0.92	0.93	0.68	0.76	0.72	0.71	0.91	0.91	0.82	0.81	0.84	0.84
Ours-SS	0.86	0.89	0.82	0.86	0.66	0.75	0.62	0.68	0.87	0.90	0.80	0.79	0.81	0.82

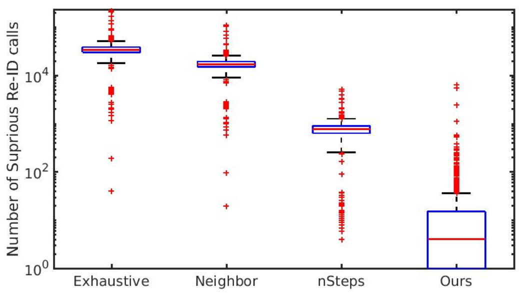

Figure 3. The re-identification calls made by different methods on DukeMTMC dataset.

Table 3. PCH on NLPR-Set4 when trained without AE and with AE. AE(same) represents AE is trained on same dataset, AE (N) represents that AE is trained on a bigger dataset with sequence length N.

Configuration	Episodes	PCH
Without AE	25587	53.6
AE (same)	21704	65
AE (10)	25588	62.4
AE (20)	24883	64
AE (50)	25549	64.8

rations to test impact of AE, like training the policy without AE, using AE trained on the training set of the same dataset which is named as AE(same), finally using AE which is trained on a larger dataset named as AE(Num) (with sequence length equals Num). AE(same) train fastest and achieves highest PCH. AE(num) achieves similar performance but train little slow than AE(same). Training without AE couldn't improve PCH beyond 53.6. AE(num) is selected as final configuration because it avoids retraining and has comparable performance. We choose final sequence length to be 20 for all further experiments because PCH for sequence length 20 and 50 is very close but recall for length 20 (78%) is higher than length 50 (74%). PCH with AE is significantly higher than without AE which means using state-representation learning not only trains policy faster but also provide better performance.

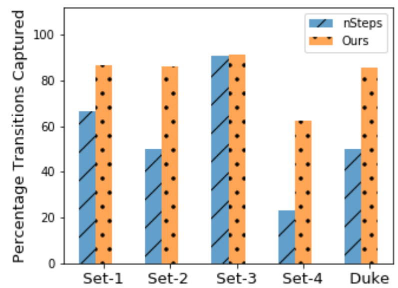

Figure 4. Figure compares the Percentage Camera Handover (PCH) of our method with state-of-the-art method.

4.2.2 Camera Selection Performance

We first evaluate the performance of our camera selection policy in terms of Precision (P), F1-score (F1) metrics. For this experiment, we use the initial location of the target to make the initial state and history vector is represented by AE with a sequence of all zeros as input. The proposed policy selects a camera from the initial state and then if the target is found in the selected camera then the state is updated accordingly using the state-transition function. The selected camera is appended in the action history and a representation is taken from AE for next decision. For this experiment, the target is re-identified using ground truth to evaluate the camera selection decisions alone. The camera selection performance is shown in the table 2. We compare the performance with state-of-the-art methods [33, 34] and other baseline methods *Neighbor*, *Exhaustive*. *Exhaustive* queries all cameras at all times. *Neighbor* assumes that the camera network topology is known and queries only the neighboring cameras. Our proposed policy performs better on various cases especially on the larger datasets. The figure 3 shows the number of Re-ID queries made by different methods on DukeMTMC dataset and our proposed policy makes very fewer queries than other related and baseline methods (CamSel [33] goes out of memory for this case).

The figure 4 shows the PCH captured by our method and nSteps [34] which is the state-of-the-art camera se-

Table 4. Camera selection performance for semi-supervised training on NLPR Set-4.

frame-skip	P	R	F1	PCH
20	0.498	0.569	0.507	0.6
10	0.594	0.752	0.634	0.592
5	0.621	0.839	0.677	0.736
2	0.67	0.84	0.72	0.728

lection method on these dataset. PCH is computed as the percentage of target transitions (from Camera C_i to C_j, $i \neq j$) that are correctly detected by using the learned policy. Missing more target transitions hurts overall tracking performance, and increased chances of not finding the target again. The figure shows that our proposed method leads to an absolute improvement of 39% for NLPR-Set4 and 35% for DukeMTMC datasets over nSteps method. This increase is substantial for NLPR-Set4 and DukeMTMC that have higher target transition times.

4.2.3 Semi-supervised Training

To show that camera selection policy can be trained with limited supervision, we provide reward after skipping a few frames as explained in sec 3. Table 4 shows the Precision (P), Recall (R), F1-scores (F1), and Percentage camera handover (PCH) for different number of frames skipped before providing reward on NLPR-Set4 dataset. Finally, we choose a frame-skip value of 5 for all datasets for making camera selections. By using a reward every 5 frames, the annotation cost is reduced by 5 times. The camera selection performance on all datasets is shown in table 2 as *Ours-SS*.

4.3. Benefits of Camera Selection Decisions

4.3.1 Multi-Camera Target Tracking

In multi-camera target tracking, we need to identify the position of a target at all times across all cameras as it is moving in the camera network. For this, the initial position of a target and the state representation of zero-initialized action history are used to make the initial state. The initial state is then used by the learned policy to select a camera where the target is expected to reappear at the next time instant. If the target is present in the selected camera frame then the next state is updated using the state-transition function (sec 3). The procedure is repeated until the video sequence ends. For re-identification, we have used pre-trained model of ABDNet [6] for DukeMTMC dataset. We used same model for all other datasets of NLPR to avoid re-training for Re-ID. For this, Re-ID features of all candidate targets are extracted using the pre-trained model and then matched with the template features of the target using threshold based cosine similarity. If the distance is less than the threshold than the target is found otherwise not. A threshold of 1.8 is used (check supplementary doc for detailed analysis). Please note that our method is single target multi-camera tracking approach and to make it work for multiple targets, we run multiple parallel pipeline of our method starting from the initial location of the target.

Camera selections inherently improves the tracking performance as shown in the table 5 which shows tracking performance on NLPR and DukeMTMC dataset using MCTA metric. The methods in the table are separated based on how these methods resolve the camera handover (re-identifying the target). In the table, *Self* means that the methods have proposed their own approach to resolve the camera handover, *GT* signifies that methods use ground truth for resolving the handover, and *Re-ID* means that a Re-ID method in [6] is used. There are two experiments as in the literature [23, 10, 7, 45, 47]. In first, only the inter-camera tracking (ICT) performance is evaluated. For this, detection and single camera tracking are taken from the ground truth. For our method, the camera selection decisions are taken at all times during ICT and a Re-ID query is made when a non-$C_\times$ camera is selected. In second (ICT+SCT), only the detections are taken from ground truth. In this, the policy is used at all times both when the target is transitioning and moving in a particular camera FOV. A Re-ID query is resolved accordingly using ABDNet [6] as explained above. The table shows that our method is better on most of the datasets especially that have higher number of cameras. The approach *Neighbor* is a baseline method where we assume that the camera topology is known and only the neighboring cameras are queried, our policy achieves better performance than this baseline on all cases. Hence, this *intelligent camera selection* strategy improves the tracking performance.

4.3.2 Multi-Camera Trajectory Forecasting

Multi-Camera Trajectory Forecasting (MCTF) is a task where the future trajectory of an object is predicted in a camera network [41]. This requires to identify the next camera where the target re-appears, the transition time, and the target's location in the identified camera. Our framework is used in the same manner as used in target tracking in a camera network in section 4.3.1. The camera selection policy gives the next camera where the target will re-appear, the length of $C_\times$ selection gives the transition time, and the Re-ID block gives the location of the target in next camera.

We compare the performance with several baseline methods as described in [41]. These baselines focus only on the next camera prediction and ignore the transition time. Hence, we introduce another baseline method where we use an LSTM based approach for making camera selection decisions in a sequential manner as compared to direct prediction. This baseline is trained as an encoder-decoder using supervised learning. It selects a camera each timestep and

Table 5. Average MCTA values for ICT alone and both SCT-ICT case on NLPR_MCT and DukeMTMC dataset. The results are separated based on the type of association method. *Self* means a method uses its own association, *GT* represents ground truth, and *Re-ID* signifies that a Re-ID method is used for association. We used ABDNet [6] for Re-ID. Mem means that the method goes out of memory.

		Inter-camera tracking (ICT)					**Single-camera tracking + ICT**				
Approach	Association	Set-1	Set-2	Set-3	Set-4	Duke	Set-1	Set-2	Set-3	Set-4	Duke
[47]	Self	0.9152	0.9132	0.5163	0.7152	-	0.8831	0.8397	0.2427	0.4357	-
[8]	Self	0.7425	0.6544	0.7369	0.3945	-	0.7477	0.6561	0.2028	0.2650	-
[7]	Self	0.6617	0.5907	0.7105	0.5703	-	0.6903	0.6238	0.0848	0.1830	-
[10]	Self	0.3203	0.3456	0.1381	0.1562	-	0.8162	0.7730	0.1240	0.4637	-
[23]	Self	0.9610	0.9264	0.7889	0.7578	-	-	-	-	-	-
[45]	Self	0.835	0.703	0.742	0.385	-	0.8525	0.7370	0.4724	0.3778	-
CamSel [33]	GT	0.8210	0.7498	0.9099	0.8993	Mem	0.8235	0.7503	0.9134	0.9118	Mem
nSteps [34]	GT	0.9016	0.8741	0.9038	0.8074	0.8027	0.9018	0.8806	0.9058	0.7871	0.8191
Ours	GT	0.968	0.963	0.914	0.759	0.902	0.966	0.961	0.906	0.776	0.894
Neighbor	Re-ID	0.6405	0.3627	0.2618	0.5386	0.6784	0.5119	0.2564	0.1445	0.4426	0.5487
Ours	Re-ID	0.9292	0.8806	0.8426	0.7808	0.8855	0.7639	0.7594	0.3547	0.5258	0.7308

Table 6. The camera selection performance on WNMF dataset. The baseline methods are taken from dataset baselines [41] and LSTM (Cam. Sel.) is a camera selection based baseline.

Model	Accuracy(%) Top 1	Top 3
Shortest real-world distance	46.8	92.2
Most frequent transition	65.7	91.8
Most similar trajectory	69.7	94.5
Hand-crafted features	70.7	94.1
Fully-connected network	73.4	95.1
LSTM (Pred.)	74.4	94.2
GRU (Pred.)	75.1	94.9
Ours (Pred.)	79	94
LSTM (Cam. Sel.)	63.1	91.5
Ours (Cam. Sel.)	93.28	96.27

if the selected camera is $c_\times$ then nothing changes otherwise the bounding box location of the target is picked using a Re-ID method. This baseline is named LSTM (cam. sel.).

The performance comparison of these baselines with our method is shown in table 6. The table shows top-1 and top-3 accuracy for next camera prediction. The table shows all baselines from the dataset results [41] and our methods. LSTM (pred.) is a method which predicts the next camera using an LSTM, Ours (Pred.) is our method used for prediction which achieves 4% better top-1 accuracy than other baselines in prediction. In this, the first non-$c_\times$ camera is used as the next camera from the sequence of selected cameras. Whereas, when our method is used as a selection framework (Ours (cam. sel.) in table) then its top-1 accuracy is 18% more than the second best baseline.

Another important task to perform in MCTF is identifying the transition time. The total time-steps when $c_\times$ is selected by the policy or till when the target is not re-identified is the transition time for our method. This is shown in figure 5. It shows the ground truth (oval markers) and predicted transition time (cross marker). The difference in transition time is represented by vertical lines. Overall, 80% of the tracks have a difference of less than 10 frames.

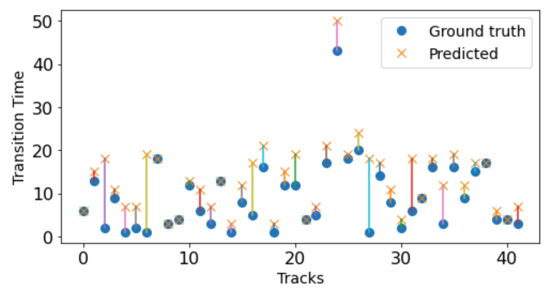

Figure 5. Figure shows the difference in the transition time captured of multiple tracks/targets of WNMF dataset.

5. Conclusion

We proposed a novel method to make camera selection decisions using DQN approach in reinforcement learning. We encoded the action history using LSTM based Auto-Encoder (AE) that helped to learn the policy faster that also achieves better performance. Through various other experiments, we showed that our method achieves better camera selection and tracking performance on larger camera networks such as DukeMTMC and CityFlow dataset. Later, we showed that our method trains in a semi-supervised manner and achieves comparable performance.

6. Acknowledgement

We gratefully acknowledge Infosys Center for Artificial Intelligence (CAI) at IIIT-Delhi for its partial support for conducting this research work.

References

[1] M. Ayazoglu, B. Li, C. Dicle, M. Sznaier, and O. I. Camps. Dynamic subspace-based coordinated multicamera tracking. In *2011 International Conference on Computer Vision*, pages 2462–2469, Nov 2011.

[2] Bo Wu and R. Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 90–97 Vol. 1, Oct 2005.

[3] M. Bredereck, X. Jiang, M. Körner, and J. Denzler. Data association for multi-object tracking-by-detection in multi-camera networks. In *2012 Sixth International Conference on Distributed Smart Cameras (ICDSC)*, pages 1–6, Oct 2012.

[4] Visesh Chari, Simon Lacoste-Julien, Ivan Laptev, and Josef Sivic. On pairwise costs for network flow multi-object tracking. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5537–5545, 2015.

[5] K. W. Chen, C. C. Lai, P. J. Lee, C. S. Chen, and Y. P. Hung. Adaptive learning for target tracking and true linking discovering across multiple non-overlapping cameras. *IEEE Transactions on Multimedia*, 13(4):625–638, Aug 2011.

[6] Tianlong Chen, Shaojin Ding, Jingyi Xie, Ye Yuan, Wuyang Chen, Yang Yang, Zhou Ren, and Zhangyang Wang. Abd-net: Attentive but diverse person re-identification, 2019.

[7] W. Chen, L. Cao, X. Chen, and K. Huang. A novel solution for multi-camera object tracking. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 2329–2333, Oct 2014.

[8] W. Chen, X. Chen, and K. Huang. Multi-Camera Object Tracking (MCT) Challenge. `http://http://mct.idealtest.org/Datasets.html/`, 2014.

[9] X. Chen, L. An, and B. Bhanu. Multitarget tracking in nonoverlapping cameras using a reference set. *IEEE Sensors Journal*, 15(5):2692–2704, May 2015.

[10] X. Chen and B. Bhanu. Integrating social grouping for multitarget tracking across cameras in a crf model. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(11):2382–2394, Nov 2017.

[11] R. T. Collins. Multitarget data association with higher-order motion models. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1744–1751, June 2012.

[12] Shahar Daliyot and Nathan S. Netanyahu. A framework for inter-camera association of multi-target trajectories by invariant target models. In Jong-Il Park and Junmo Kim, editors, *Computer Vision - ACCV 2012 Workshops*, pages 372–386, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[13] Abir Das, Anirban Chakraborty, and Amit K. Roy-Chowdhury. Consistent re-identification in a camera network. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 330–345, Cham, 2014. Springer International Publishing.

[14] T. de Bruin, J. Kober, K. Tuyls, and R. Babuška. Integrating state representation learning into deep reinforcement learning. *IEEE Robotics and Automation Letters*, 3(3):1394–1401, 2018.

[15] A. Dehghan, S. M. Assari, and M. Shah. Gmmcp tracker: Globally optimal generalized maximum multi clique problem for multiple object tracking. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4091–4099, June 2015.

[16] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):267–282, Feb 2008.

[17] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):267–282, Feb 2008.

[18] R. Hamid, R. K. Kumar, M. Grundmann, K. Kim, I. Essa, and J. Hodgins. Player localization using multiple static cameras for sports visualization. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 731–738, June 2010.

[19] Omar Javed, Khurram Shafique, Zeeshan Rasheed, and Mubarak Shah. Modeling inter-camera space-time and appearance relationships for tracking across non-overlapping views. *Comput. Vis. Image Underst.*, 109(2):146–162, Feb. 2008.

[20] Na Jiang, SiChen Bai, Yue Xu, Chang Xing, Zhong Zhou, and Wei Wu. Online inter-camera trajectory association exploiting person re-identification and camera topology. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, page 1457–1465, New York, NY, USA, 2018. Association for Computing Machinery.

[21] S. Khan and M. Shah. Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1355–1360, Oct 2003.

[22] Cheng-Hao Kuo, Chang Huang, and Ram Nevatia. Inter-camera association of multi-target tracks by on-line learned appearance affinity models. In *Proceedings of the 11th European Conference on Computer Vision Part I*, ECCV2010, pages 383–396, Berlin, Heidelberg, 2010. Springer-Verlag.

[23] Y. Lee, Z. Tang, J. Hwang, and Y. Online-learning-based human tracking across non-overlapping cameras. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):2870–2883, Oct 2018.

[24] Li Zhang, Yuan Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[25] D. Makris, T. Ellis, and J. Black. Bridging the gaps between cameras. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–205–II–210 Vol.2, June 2004.

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[27] Milind Naphade, Zheng Tang, Ming-Ching Chang, David C. Anastasiu, Anuj Sharma, Rama Chellappa, Shuo Wang,

Pranamesh Chakraborty, Tingting Huang, Jenq-Neng Hwang, and Siwei Lyu. The 2019 ai city challenge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[28] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision workshop on Benchmarking Multi-Target Tracking*, 2016.

[29] Ergys Ristani and Carlo Tomasi. Tracking multiple people online and in real time. In Daniel Cremers, Ian Reid, Hideo Saito, and Ming-Hsuan Yang, editors, *Computer Vision – ACCV 2014*, pages 444–459, Cham, 2015. Springer International Publishing.

[30] Ergys Ristani and Carlo Tomasi. Features for multi-target multi-camera tracking and re-identification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[31] Shafique and Shah. A non-iterative greedy algorithm for multi-frame point correspondence. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 110–115 vol.1, Oct 2003.

[32] Anil Sharma. Intelligent querying in camera networks for efficient target tracking. In *IJCAI*, pages 6458–6459, 2019.

[33] Anil Sharma, Saket Anand, and Sanjit K. Kaul. Reinforcement learning based querying in camera networks for efficient target tracking. In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS), 2019*.

[34] Anil Sharma, Saket Anand, and Sanjit K. Kaul. Intelligent querying for target tracking in camera networks using deep q-learning with n-step bootstrapping. *Image and Vision Computing*, 2020.

[35] Anil Sharma and Arun Balaji Buduru. Foresee: Attentive future projections of chaotic road environments. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2073–2075, 2018.

[36] Anil Sharma and Sanjit Kaul. Two-stage supervised learning-based method to detect screams and cries in urban environments. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(2):290–299, 2015.

[37] Anil Sharma and Prabhat Kumar. Foresee: Attentive future projections of chaotic road environments with online training. *arXiv preprint arXiv:1805.11861*, 2018.

[38] Anil Sharma, Mayank K Pal, Saket Anand, and Sanjit K Kaul. Stratified sampling based experience replay for efficient camera selection decisions. In *2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM)*, pages 144–151. IEEE, 2020.

[39] Vishal Sharma, Arun S Varma, Atul Singh, Divyansh Singh, and Bikarama Prasad Yadav. A critical review on the application and problems caused by false alarms. *Intelligent Communication, Control and Devices*, pages 371–380, 2018.

[40] G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah. Part-based multiple-person tracking with partial occlusion handling. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1815–1821, June 2012.

[41] O. Styles, T. Guha, V. Sanchez, and A. Kot. Multi-camera trajectory forecasting: Pedestrian trajectory prediction in a network of cameras. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4379–4382, 2020.

[42] S. Tang, M. Andriluka, B. Andres, and B. Schiele. Multiple people tracking by lifted multicut and person re-identification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3701–3710, July 2017.

[43] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David Anastasiu, and Jenq-Neng Hwang. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[44] Yonatan Tariku Tesfaye, Eyasu Zemene, Andrea Prati, Marcello Pelillo, and Mubarak Shah. Multi-target tracking in multiple non-overlapping cameras using constrained dominant sets. *CoRR*, abs/1706.06196, 2017.

[45] L. Cao W. Chen, X. Chen, K. Huang, K. Huang, and K. Huang. An equalized global graph model-based approach for multicamera object tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(11):2367–2381, Nov 2017.

[46] J. Wan and Liu Li. Distributed optimization for global data association in non-overlapping camera networks. In *2013 Seventh International Conference on Distributed Smart Cameras (ICDSC)*, pages 1–7, Oct 2013.

[47] G. Medioni Y. Cai. Exploring context information for inter-camera multiple target tracking. In *IEEE Winter Conference on Applications of Computer Vision*, pages 761–768, March 2014.

[48] Mang Ye, Jianbing Shen, Gaojie Lin, Tao Xiang, Ling Shao, and Steven C. H. Hoi. Deep learning for person re-identification: A survey and outlook, 2020.

[49] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Y. Choi. Action-decision networks for visual tracking with deep reinforcement learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1349–1358, July 2017.

[50] S. Zhang, E. Staudt, T. Faltemier, and A. K. Roy-Chowdhury. A camera network tracking (camnet) dataset and performance baseline. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 365–372, Jan 2015.

[51] Shu Zhang, Yingying Zhu, and Amit Roy-Chowdhury. Tracking multiple interacting targets in a camera network. *Computer Vision and Image Understanding*, 134:64 – 73, 2015. Image Understanding for Real-world Distributed Video Networks.