# Reinforcement Learning Based Querying in Camera Networks for Efficient Target Tracking

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

## Abstract

Surveillance camera networks are a useful monitoring infrastructure that can be used for various visual analytics applications, where high-level inferences and predictions could be made based on target tracking across the network. Most multi-camera tracking works focus on re-identification problems and trajectory association problems. However, as camera networks grow in size, the volume of data generated is humongous, and scalable processing of this data is imperative for deploying practical solutions. In this paper, we address the largely overlooked problem of scheduling cameras for processing by selecting one where the target is most likely to appear next. The inter-camera handover can then be performed on the selected cameras via re-identification or another target association technique. We model this scheduling problem using reinforcement learning and learn the camera selection policy using Q-learning. We do not assume the knowledge of the camera network topology but we observe that the resulting policy implicitly learns it. We evaluate our approach using NLPR MCT dataset, which is a real multi-camera multi-target tracking benchmark and show that the proposed policy substantially reduces the number of frames required to be processed at the cost of a small reduction in recall.

Camera networks have emerged as a preferred sensing infrastructure for monitoring and surveillance of public spaces. With advances in visual analytics, we see an increasing number of practical applications like footfall estimation and prediction, crowd and traffic flow analysis, content based retrieval for forensics. These applications are driven by state-of-the-art visual detection, tracking and re-identification techniques that are robust to lighting variations, background clutter, occlusions, and non-overlapping fields of view of cameras in the network. An example of a camera network and trajectories of different targets is shown in figure 1. State-of-the-art techniques that have shown promise in overcoming these challenges often rely on deep learning architectures that are computationally expensive and have substantial hardware requirements like GPUs to run at an acceptable frame rate. This computational challenge is exacerbated by the deluge of video data generated from a network of cameras, making it challenging to implement these applications at scale.




Figure 1: Camera topology of NLPR_MCT dataset-4 (Chen, Chen, and Huang 2014). The figure shows the trajectories of person 5 and 6 across different cameras. The camera network is deployed in a parking area and all cameras have non-overlapping view.

While many vision techniques have been developed to tackle the problems of tracking and re-identification under illumination variations, clutter and occlusions, there is a much smaller body of work that addresses the problem of camera selection to efficiently identify target handovers across cameras. Inter-camera handovers are resolved by matching the current target's template with potential target instances detected in candidate cameras, and is typically handled using visual re-identification techniques. This problem of correctly resolving handovers is difficult, especially when candidate cameras have non-overlapping FOVs, implying that a target's transition time between two FOVs is non-deterministic and unknown. Ideally, no re-identification queries should be made during the target's transition period, as a larger number of queries result in an increased chance of false alarms, which can *severely deteriorate* the overall tracking performance. An auxiliary consequence of a higher number of queries is significantly higher computational cost. As the transition time is not deterministic, in order to minimize the false alarms, it is important to devise a camera selection policy that intelligently schedules camera re-identification queries.

Target tracking in a camera network has been explored extensively in the past using various approaches (Chen et al. 2014; Y. Cai 2014; Lee et al. 2018; W. Chen et al. 2017; Zhang et al. 2015; Kuo, Huang, and Nevatia 2010; Daliyot and Netanyahu 2013). Most of these approaches track targets in a two-step framework. First, single camera tracking (SCT) is applied to identify the trajectory of the target within a single camera's FOV. Second, inter-camera tracking (ICT) is performed to resolve handovers by finding associations for the target tracked by SCT. In addition to handovers, ICT is also invoked during SCT for handling periods of occlusion, which may vary significantly depending on the targets' speed and path. ICT requires searching targets in multiple cameras at different time instances, by making repeated re-identification queries until the target is found and SCT is invoked for the identified cameras. In order to reduce the search space for ICT, most existing techniques limit the set of candidate cameras to be queried by assuming knowledge of the camera network topology (W. Chen et al. 2017), while some attempt to model the transition time as a random variable (Lee et al. 2018; Javed et al. 2008).

Recently, (Lee et al. 2018) proposed a method that discovers a *camera-link* model that identifies candidate cameras for ICT based on appearance based features. They model the inter-camera transition time using a Gaussian distribution, which is used during test-time to generate a sample transition time for which their system waits before initiating re-identification queries. Not surprisingly, this approach for ICT shows an improvement over exhaustive search and nearest neighbor based search, however, it is not clear if a static distribution of transition time is the best model choice.

In this paper, we argue that the transition time distribution is conditioned on the target instance and its properties like speed, which itself may be time-varying. Consequently, our proposed ICT approach is a policy for scheduling candidate cameras for re-identification queries based on the target's most recent SCT trajectory, as well as the history of candidate cameras queried. We model our ICT approach as a reinforcement learning (RL) problem and learn a policy that picks an action of querying one of the candidate cameras or not querying any camera. The learned RL policy implicitly discovers the camera network topology, with our only assumption about the network being that all cameras are static. Since the focus of our work is on the camera selection policy for ICT, our state representation abstracts out the visual appearance based features and only retains spatial information from SCT. In our experiments, we evaluate our approach with real data (NLPR_MCT dataset (Chen, Chen, and Huang 2014)) where the targets are restricted to pedestrians, and compare them to the state-of-the-art.

The NLPR_MCT records real-world scenarios in four different sub-datasets. This dataset has both indoor and outdoor cameras deployed in a campus building, in parking areas, and along footpaths. We will be using this dataset for training and testing of our proposed approach. We will also compare our results with state of the art methods on this dataset. Figure 1 shows a sub-dataset of NLPR_MCT dataset.

Our specific contributions are:

- We propose an intelligent camera selection approach for inter-camera tracking in a camera network. The goal is to learn a policy that schedules the re-identification query by selecting the next candidate camera where the target is likely to appear.
- We formulate the camera selection as a reinforcement learning problem and learn the policy using Q-learning (Sutton and Barto 1998), without any knowledge of the camera network topology.
- We demonstrate that the camera selection policy queries a very small number of frames by making a small trade-off on the recall values.
- We demonstrate our Q-learning based approach on NLPR_MCT (Chen, Chen, and Huang 2014) dataset implicitly learns the network topology.

The rest of the paper is structured as follows. We discuss related work for tracking in a camera network in the next section followed by the details of our proposed framework. We then describe the dataset, evaluation metric and experimental results before concluding the paper.

## Related Works

In this section, we survey related works for multi-camera tracking and scheduling of cameras for tracking. For multi-camera tracking, initial works such as (Hamid et al. 2010; Zhang, Zhu, and Roy-Chowdhury 2015; Khan and Shah 2003) use 3D coordinates of the target object to track in the camera network with overlapping field of views. This often require camera calibration and network topology for tracking using 3D coordinates. Some (Zhang, Zhu, and Roy-Chowdhury 2015) fuse the single-camera tracks from SCT to generate the 3D location of the target from the calibrated cameras and then track the target by assuming the spatial relationship between cameras. Tha approach in (Zhang, Zhu, and Roy-Chowdhury 2015) also assumes overlapping views and formulate a network flow problem for multi-camera tracking. With cameras having overlapping field of view, a homography matrix is estimated (Khan and Shah 2003) to find the 3D position of the target to enable tracking using Kalman filter or particle filter. However, an assumption of overlapping field of view is overly restrictive in real-world scenarios.

Many works exist that represent few initial works that work with non-overlapping field of views (Chen, An, and Bhanu 2015; Daliyot and Netanyahu 2013; Kuo, Huang, and Nevatia 2010; Chen et al. 2011). Few works such as (Kuo, Huang, and Nevatia 2010; Zhang et al. 2015; Daliyot and Netanyahu 2013; Zhang et al. 2015; Y. Cai 2014; Lee et al. 2018; W. Chen et al. 2017) assume that the tracks from individual camera are given and they extend these to multiple cameras using inter-camera associations by building an affinity model of the appearances of individual tracks (Kuo, Huang, and Nevatia 2010), using social grouping model (Zhang et al. 2015), using data association across different cameras (Makris, Ellis, and Black 2004; Chen, An, and Bhanu 2015; Daliyot and Netanyahu 2013) and graph-based techniques (W. Chen et al. 2017; Zhang et

al. 2015). (Chen et al. 2011) makes a spatio-temporal mapping between cameras for 3D coordinates. Few other works model the travel time of the moving target (Javed et al. 2008) to learn space-time relation between the cameras, (Ristani and Tomasi 2015; Ristani et al. 2016) use clique based methods, (Ristani et al. 2016) also formulates within and across camera tracking in a unified framework.

To track objects in disjoint views, appearance cues of the target are modeled in (Sunderrajan and Manjunath 2013; Zhang, Zhu, and Roy-Chowdhury 2015; Daliyot and Netanyahu 2013). The appearance cues were integrated with spatio-temporal reasoning (Hamid et al. 2010; Kuo, Huang, and Nevatia 2010) and graph-based methods (W. Chen et al. 2017). Huang et. al (Huang and Russell 1997) has integrated the color and the size of the object with velocities and arrival time using Bayesian inference to track a vehicle in two camera views. Pasula et. al (Pasula et al. 1999) extend the approach to more than two cameras by including hidden variables in the Bayesian framework. Matei et al. (Matei, Sawhney, and Samarasekera 2011) use a multi-hypothesis framework instead of Bayesian, while (Chen and Bhanu 2017) formulates the tracking problem using conditional random fields (CRF). The formulation uses spatio-temporal and appearance features to enable tracking. (W. Chen et al. 2017) enables tracking using a global graph model in which a MAP association problem is formulated and solved using flow graphs. (Lee et al. 2018) performs SCT and ICT separately by using human appearance features along with segmentation using change point detection. This approach is state of the art on NLPR_MCT dataset. Other approaches for tracking in a camera network use re-identification and data association by matching a query image with the templates in a gallery set (Zheng, Zheng, and Yang 2017). In this paper, we propose to select a camera given the current state of the target where the target is expected to appear next. We will show that the camera selection based approach polls very few frames as compared to the related methods. We learn a policy directly from the data to intelligently select cameras, and we formulate multi-camera tracking problem as a reinforcement learning problem that learns this policy using trial-and-error.

## Proposed Methodology

In this section, we formulate the camera selection approach and present an architecture that integrates camera selections with re-identification to enable target tracking.

### Problem Formulation

Target tracking in a camera network needs to handle inter-camera handovers by resolving associations between the tracked target and potential targets detected across all candidate cameras. This association problem is typically done by visual re-identification or verification methods. When camera networks have disjoint FOVs, a target may only reappear in another candidate camera after a certain *transition time*, which in turn depends on various factors like inter-camera distance and target speed, where the latter would typically be target-dependent and time-varying. Re-identification queries made at times when a target is unlikely to appear in a candidate camera can lead to unnecessary false associations, deteriorating the tracking performance. To counter this challenge of handling the time-varying nature of inter-camera transitions, we attempt to schedule the re-identification queries by intelligently selecting a candidate camera or waiting. Due to its time-varying nature, a befitting model for this problem is a reinforcement learning (RL) based camera selection policy that identifies a candidate camera or decides to wait. Thus the task is to learn the policy $\pi(s_t) = p(a_t|s_t)$ at time $t$, where $s_t$ is the current state (detailed in next paragraph), $a_t$ is the action taken at time $t$, which corresponds to selecting one of the $N$ cameras or to wait (by picking a *dummy camera*).

**Algorithm 1** Target tracking in a camera network using proposed RL based method. $\pi$ is camera selection policy. `c,b` are current camera and corresponding bounding box for target location.

**procedure** TRACK(`c,b`, $\pi$)
  `traj` $\leftarrow []$ ▷ Stores computed trajectory
  `traj.append((c,b))`
  $h \leftarrow$ `ZEROS` ▷ Initialize history with ZEROS
  $\tau \leftarrow$ `ZERO` ▷ Initialize telapse to `ZERO`
  `rt` $\leftarrow$ `getRT(b)` ▷ Discretize the frame to get region of target's location
  $s \leftarrow$ `[c,rt,h,`$\tau$`]` ▷ Concatenate location and history
  **while** True **do**
    `cprob` $\leftarrow \pi(s)$ ▷ Get distribution using policy
    **if** all(cprob(:) == cprob(1)) **then**
      c = `randi(length(cprob))`
    **else**
      c = `argmax` (cprob)
    `b` $\leftarrow$ `get the bounding box location`
    **if** `b` is not empty **then**
      `rt` $\leftarrow$ `getRT(b)`
      `traj.append((c,b))`
    $s \leftarrow$ `f(s,c)`
  **return** `traj`

To model the RL problem, we will define the state space, action space, and the training methodology.

**State**: The state $s_t$ at time $t$ captures the spatial and temporal information of the target. The spatial information include the position $r_t$ of the target in the current camera FOV and the temporal information include camera history $h_t$, and time elapsed $\tau$. The state vector is following,

$$s_t = (x_t, h_t, \tau). \quad (1)$$

The individual elements of the state space are following:

1. $x_t$: it is the last seen location of the target. It consists of $(c, r)$, where $c$ is the last seen camera and $r$ is the spatial location of the target in camera $c$. To compute, $r$, the input image is divided into a $8 \times 8$ grid, and all the cells are numbered in row-major order. The cell numbers corresponding to the target's bounding box are identified and one of these is used as $r$ as shown in figure 2.
2. $h_t$: it represents the history of the cameras polled by the learned policy in past $N$ time steps, where $N$ is the number of cameras.

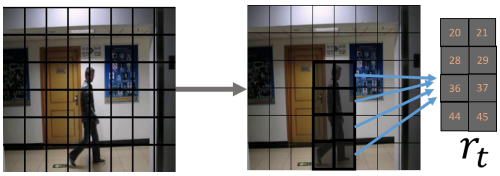


Figure 2: The cells in the grid capture the spatial location of the target. The image is first discretized in $8 \times 8$ grid, and all cells are numbered sequentially in row-major order. The cell number on the target position is used as the spatial location of the target named $r_t$ at time $t$.

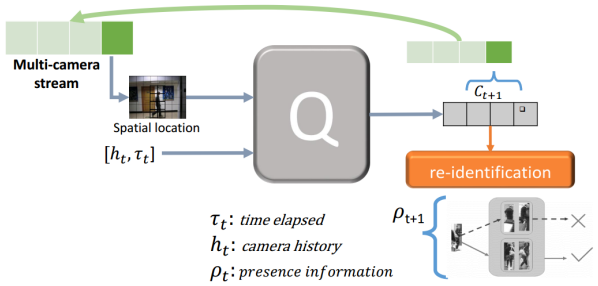


Figure 3: The proposed architecture using reinforcement learning. The architecture shows two blocks, block $Q$ and block *presence*. Block Q learns a policy to select a new camera using current state and block *presence* verifies whether the target is present in the camera frame chosen.

3. $\tau$: it captures the time elapsed since the target was last seen in a camera. The elapsed time is discretized with a step size of $0.25$ sec and with every time step that the target is not found, the value of $\tau$ increments by $1$. It resets to $0$ once the target is found.

**Actions**: The action $a_t$ at time $t$ is encoded by $N+1$ dimension vector, where $N$ is the number of cameras in the camera network. The action $N+1$ is selected when the policy selects no camera, i.e., the target is not visible in all the cameras.

**State evolution**: After deciding an action $a_t$, the next state $s_{t+1}$ is decided by following state evolution function:

$$s_{t+1} = f(s_t, a_t) \tag{2}$$

The function appends the selected camera $c_t$ to the camera history. If the target is found in polled camera then last seen location is updated to new $(c, r)$ otherwise telapse is incremented accordingly.

**Reward**: The reward function $R(s)$ is defined for each state irrespective of the action $a$. At time $t$, it is following:

$$R(s_t) = \begin{cases} +1 & \text{if target is present in } s_t \\ -1 & otherwise \end{cases} \tag{3}$$

**Training procedure**: We define state-action value function Q to estimate the values (reward) of actions at a given state. The estimates will then be used to make the action selection decision. The function $Q(s, a)$ estimates the value of state action pair $(s, a)$. The goal is to learn optimal state-action function $Q^*$ (Sutton and Barto 1998). Traditionally, the Q-functions are iteratively learned using Q-learning (Sutton and Barto 1998) as shown below:

$$Q(s_t, a_t) \Leftarrow Q(s_t, a_t) + \alpha \Big( R(s_{t+1}) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \Big) \tag{4}$$

Where $\alpha$ is the learning rate, and $\gamma$ is the discount factor. Sufficient exploration is essential for Q-learning methods to explore complete state-space, we use epsilon-greedy exploration strategy (Sutton and Barto 1998) with epsilon annealing. **Policy**: The policy $\pi$ selects an optimal action from the learned Q-functions. After learning, given the target state, it selects an optimal action in-state $s_t$ as:

$$\pi_t^*(s_t) = arg \max_a Q^*(s_t, a) \tag{5}$$

Table 1: Details of NLPR_MCT dataset (W. Chen et al. 2017), which has four subsets. The table shows number of cameras (#Cameras), duration of the capture, frame rate (FPS) and the number of people (#People) captured in each subset.

| | Set1 | Set2 | Set3 | Set4 |
|---|---|---|---|---|
| #Cameras | 3 | 3 | 4 | 5 |
| Duration | 20 min | 20 min | 3.5 min | 24 min |
| FPS | 20 | 20 | 25 | 25 |
| #People | 235 | 255 | 14 | 49 |

## System Architecture

The system architecture is shown in figure 3. The architecture consists of two blocks, first, block Q which learns a policy $\pi$ to select the next camera where the target will appear given target's current state. Second, the *presence* block which will verify whether the target is present in the camera selected by the policy at that time frame. The *presence* block takes as input the selected camera frame and will return 1 if the target is present along with the bounding box otherwise it returns a 0. The presence block can be implemented using person re-identification (Zheng, Zheng, and Yang 2017). In this paper, we simulate the presence block with errors to falsely identify the presence. Details of this block are given in the results sections. The policy (block Q) takes as input the current state of the target (the spatial information and the temporal history) and selects a camera where the target is likely to appear. The policy is learned using Q-learning (Sutton and Barto 1998). Algorithm 1 show the pseudo-code for tracking a target using the proposed method.

# Experiments and Results

In this section, we present details of the dataset used, the evaluation metric and the experimental results.

## Dataset and Evaluation Metric

*Dataset*: We have used NLPR_MCT data set (Chen, Chen, and Huang 2014) for training and testing of our proposed

approach. The dataset consists of four sub-datasets each having $3-5$ cameras with a resolution of $320 \times 240$. Details of the dataset are given in Table 1. The dataset comprises cameras installed in both indoor and outdoor environments with significant illumination variation across different cameras. The set-1 and set-2 have the same environment and network topology. The set-3 was captured in an office building, and the set-4 was captured in a parking area. We learn a separate policy for set-3, set-4, and set-1. Since the camera network in set-2 is same as set-1, we use the same policy for both subsets. The training and the testing sets are constructed by randomly selecting half the people for training and the remaining half for testing for each dataset. We expect the policy to implicitly learn the network topology, and so long as the network is static, the policy should work for all new, unseen target individuals. Typically, CCTV network topologies in the real-world are seldom modified.

We define evaluation metrics over the entire sequence of frames generated by the a camera network. The sequence is indexed by time-steps corresponding to the time of frame capture for the cameras. Since the cameras operate on the same frame rate for a given subset, we can ignore any synchronization errors without any significant impact on the camera selection and tracking performance.

Table 2: Table is showing confusion matrix of the camera selections made by the proposed policy for DB-3. Rows are the ground truth cameras and columns are the cameras polled by the policy. Values are percentages rounded off to third decimal.

| $\downarrow GT/p \rightarrow$ | $C_1$ | $C_2$ | $C_3$ | $C_4$ | $C_\times$ |
|---|---|---|---|---|---|
| $C_1$ | 0.91 | 0.003 | 0.005 | 0.002 | 0.081 |
| $C_2$ | 0.006 | 0.796 | 0.008 | 0.011 | 0.178 |
| $C_3$ | 0.015 | 0.02 | 0.828 | 0.015 | 0.120 |
| $C_4$ | 0.003 | 0.005 | 0.004 | 0.767 | 0.219 |
| $C_\times$ | 0.08 | 0.08 | 0.079 | 0.072 | 0.685 |

*Evaluation Metric*: To evaluate the camera selection performance, we report camera selection accuracy, precision and recall computed over the entire sequence of each subset. In order to consider instances when the target is not visible in any of the cameras, we introduce a dummy *null* camera and denote it by $C_\times$. Given a target, let the ground truth sequence of cameras in which it appears be contained in the vector $g$ and sequence of cameras polled by the policy be in vector $p$ with the $i^{th}$ element indicated using a subscript. The Accuracy (A), precision (P) and recall (R) are defined as following for a single target

$$A = \frac{\sum_i (p_i == g_i)}{\texttt{Length}(g)} \tag{6}$$

$$P = \frac{\sum_i ((p_i == g_i) \ \wedge \ (p_i != C_\times))}{\sum_i (p_i != C_\times)} \tag{7}$$

$$R = \frac{\sum_i ((p_i == g_i \ \wedge \ g_i != C_\times)}{\sum_i (g_i != C_\times)} \tag{8}$$

The final value for each of these metrics is reported as an average computed over all targets. Along with $A, P, R$, we also report number of frames polled ($F$) during an inter-camera transition of the target. It is defined as

$$F = \sum_i ((g_i == C_\times) \ \wedge \ (p_i != C_\times)) \quad + \quad \sum_i ((p_i != g_i) \ \wedge \ (g_i != C_\times) \ \wedge \ (p_i != C_\times)) \tag{9}$$

$F$ is an important measure because with a large number of frames polled, the chance of false alarms during a re-identification query as well as the computational complexity is substantially increased.

We also evaluate the overall performance of target tracking in a camera network, when our camera selection policy is used for ICT. We use the standard Multi-Camera Tracking Accuracy (MCTA), which gives a single scalar value for all components involved in multi-camera tracking, i.e., F1-score for detection, number of target handovers for single camera tracking, and the number of handovers in inter-camera tracking. The metric is defined as

$$MCTA = \underbrace{\left(\frac{2P_T R_T}{P_T + R_T}\right)}_{F1-score} \underbrace{\left(1 - \frac{\sum_t \mu_t^s}{\sum_t tp_t^s}\right)}_{within-camera} \underbrace{\left(1 - \frac{\sum_t \mu_t^c}{\sum_t tp_t^c}\right)}_{cross-camera} \tag{10}$$

where $P_T$ is the precision, $R_T$ is recall for target IDs. The number of target-ID mismatches at time $t$ is given by $\mu_t$ and $tp_t$ is the number of true positives in a single camera at time $t$. The superscripts $s$ and $c$ denote the single camera tracking (SCT) or cross-camera tracking (ICT) scenario. Readers are requested to see (Ristani et al. 2016; Chen, Chen, and Huang 2014) for details about the MCTA metric.

Table 3: Table showing average time taken (in the number of frames) by all targets from camera $C_i$ (row) to camera $C_j$ (column). The values $(g, p)$ are ground truth $(g)$ time and time taken by the policy $(p)$ to find the target in the next camera. The values are averaged over all targets in the test set of sub-dataset 3.

| Camera | $C_1$ | $C_2$ | $C_3$ | $C_4$ |
|---|---|---|---|---|
| $C_1$ | (57,71) | (100,105) | (0,0) | (0,0) |
| $C_2$ | (77,88) | (20,0) | (282,288) | (0,0) |
| $C_3$ | (0,0) | (78,119) | (5,40) | (275,280) |
| $C_4$ | (0,0) | (0,222) | (51,98) | (190,329) |

## Experiments

We design three experiments to evaluate our camera selection approach independently, as well as a part of a target tracking framework. As in most tracking settings, we assume the initial location of the target to be known as given by the camera and a bounding box around the target.

*Experiment-1: Evaluation of Camera Selection:*

In this experiment, we initialized the state of a target with its initial location and used the learned policy to poll cameras at subsequent time steps. Based on the presence or absence of the target, the state vector is appropriately updated.

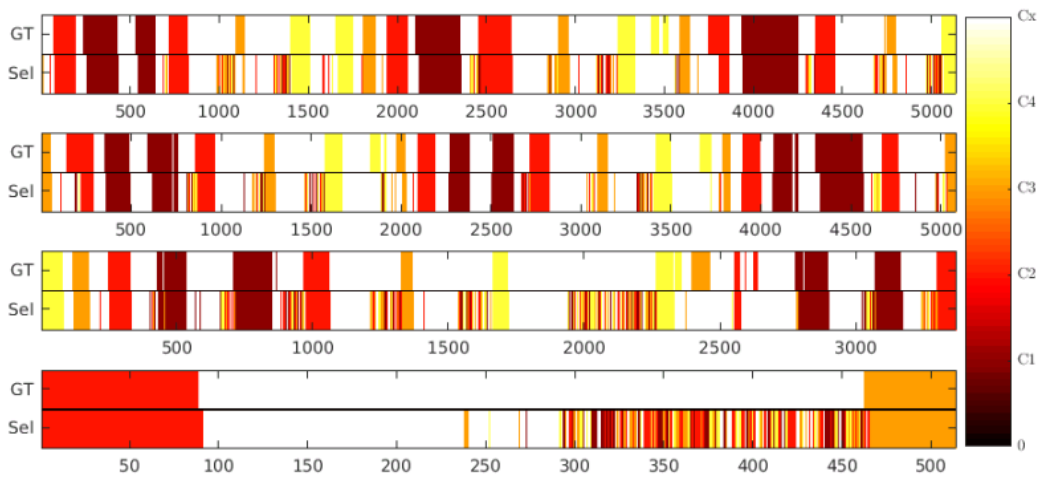


Figure 4: The figure shows the transitions for 4 targets in the testing set of dataset-3. $GT$ is the sequence of cameras in ground-truth, $Sel$ is the sequence of cameras selected/polled by the policy. Horizontal axis is the time. White color is the time when the target is transitioning between cameras and colorbar depicts the camera numbers in the plot.

Table 4: Table is showing camera selection accuracy (A), precision (P) and recall (R) for the proposed method and baseline approaches for NLPR dataset for ICT alone and for both SCT with ICT.

| | Set-1 | | | Set-2 | | | Set-3 | | | Set-4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $A$ | $P$ | $R$ | $A$ | $P$ | $R$ | $A$ | $P$ | $R$ | $A$ | $P$ | $R$ |
| | **Inter-camera Tracking (ICT)** | | | | | | | | | | | |
| Exhaustive | 0.025 | 0.008 | 1.0 | 0.019 | 0.007 | 1.0 | 0.008 | 0.002 | 1.0 | 0.017 | 0.003 | 1.0 |
| Neighbor | 0.025 | 0.013 | 1.0 | 0.019 | 0.009 | 1.0 | 0.008 | 0.003 | 1.0 | 0.017 | 0.006 | 1.0 |
| Gaussian | 0.435 | 0.215 | 0.127 | 0.40 | 0.16 | 0.195 | 0.36 | 0.007 | 0.571 | 0.33 | 0.0078 | 0.168 |
| Proposed | 0.85 | 0.042 | 0.31 | 0.86 | 0.037 | 0.31 | 0.685 | 0.026 | 0.929 | 0.519 | 0.027 | 0.808 |
| | **Single-camera tracking + Inter-camera Tracking** | | | | | | | | | | | |
| Exhaustive | 0.72 | 0.24 | 1.0 | 0.65 | 0.22 | 1.0 | 0.42 | 0.10 | 1.0 | 0.56 | 0.11 | 1.0 |
| Neighbor | 0.72 | 0.36 | 1.0 | 0.65 | 0.32 | 1.0 | 0.42 | 0.14 | 1.0 | 0.56 | 0.18 | 1.0 |
| Proposed | 0.91 | 0.95 | 0.83 | 0.88 | 0.94 | 0.78 | 0.76 | 0.64 | 0.86 | 0.77 | 0.61 | 0.91 |

In this experiment, we use the ground truth location for determining the presence of the target. We make this simplifying choice in this experiment to eliminate the uncertainty introduced due to the re-identification performance. The policy continues polling of cameras until the target exits the camera network or the sequence terminates. Metrics like accuracy, precision and recall encapsulate overall performances and allow comparative analysis as shown in Table 4, which reports the camera selection performance. In addition to the proposed policy's performance, we also compare with exhaustive search and nearest neighbor search as used in multiple related works discussed in the following text. The *Exhaustive* approach is a brute-force approach which polls each camera at all time steps until the target is found in one of the cameras. The table shows that it has $100\%$ accuracy but poor precision. The *Neighbor* approach assumes that the camera network topology is known and searches the target by polling only in the neighboring cameras. Approaches proposed in (Zhang et al. 2015; Chen and Bhanu 2017) searches the target in the adjacent cameras and hence process the same number of frames as the *neighbor search* approach. Along with these two approaches, we also compare camera selection performance with a method proposed in (Lee et al. 2018). The approach proposed in (Lee et al. 2018) first estimates the distribution of the camera transitions assuming the fact that the multiple targets generally follow same paths and then samples a transition time to reduce the number of frames to be processed. They estimate a Gaussian distribution and hence we named this approach as *Gaussian*. After the transition time, they start searching the target in cameras using a camera link model which will link different cameras having a path for transition. We repeated their experiment by estimating a Gaussian distribution from the train set and sampling a transition time for each person in the test set. The camera link model is used as set of neighboring cameras. The metrics computed in Table 4 are reported for two cases: For ICT, the metrics are computed using equations (6, 7, and 8), but only using the time instances when the target is transitioning from one camera to the other. For SCT + ICT, the entire sequences are used. As expected, we see that the proposed policy has better precision than the other competing approaches. The

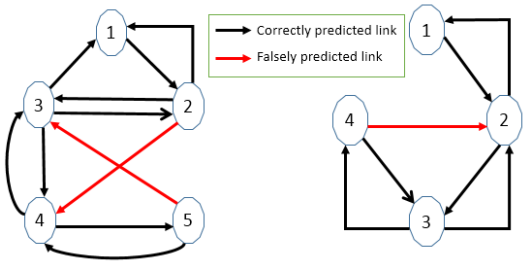


Figure 5: The figure shows the learned topology for set 4 (5 cameras) and 3 (4 cameras). A black arrow indicates the correct prediction and red arrow indicate a false positive.

Gaussian method is excluded in case of SCT + ICT, as the distribution is only defined for the ICT case.

While the A, P and R measures indicate the overall performance of camera selection, a confusion matrix shows the pairwise miss-classification in camera selection. Based on the cameras being polled by our policy at various time steps, we report a confusion matrix for DB-3 as shown in Table 2.

Figure 4 show the sequence of cameras polled by the policy as compared to what is seen in the ground truth. Horizontal axis is time and vertical axis shows the camera schedules in ground truth ($GT$) and polled by policy ($Sel$). The dark colors are camera schedules (mapped with colormap) and white is the transition time. The figure reflects that the policy starts polling the camera early when the transition time is large and skips few frames while selecting a correct camera when target enters a new camera FOV. This is because of varying speed of different target individuals.

*Experiment-2: Evaluation of Inter-Camera Tracking:*

In this experiment, we evaluate only inter-camera tracking (ICT) of our proposed method to capture the performance when the target is navigating from one camera FOV to another. Unlike experiment-1, ICT includes ground truth sequence during SCT. For ICT, it is not only required to find the next camera where the target is likely to move but also the transition time the target will take before appearing in the next camera. The transition time is critical because it will increase the number of search operations until the target is found. In this experiment, the single-camera tracking is taken from ground-truth. In our approach, we use the learned policy to select a camera at any given time using the current state of the target and when the target is located in a camera frame then ground truth results are used until it disappears from the camera FOV either due to an occlusion or a possible transition. Unlike (Lee et al. 2018), we do not model the transition time explicitly. However, the policy implicitly captures the transition time by selecting a camera at all times. The policy selects action $N+1$ when the target is not visible in all the $N$ cameras of the camera network otherwise it selects an action from 1 to $N$. The sequence of $(N+1)^{th}$ action selection gives us the transition time of the target. Using this experiment, we will show that the learned policy can learn the camera network topology and captures the transition time of the target and hence is able to reduce the number of frames to be processed (i.e., number of re-identification calls) at any given time. Visually, it can be seen in the figure 4. The length of white color in $GT$ is the actual transition time between any two cameras and in the same time duration the length of white color in $Sel$ gives the length of transition time captured by the policy.

For the experiment, the state is initialized with the initial position of the target. Whenever the target moves out of the current camera FOV, the policy selects a camera (equivalently, selects an action in reinforcement learning) where the target is expected to appear. As discussed earlier, the policy selects $(N+1)^{th}$ camera when the target is expected to be absent in all camera FOVs. The selected camera frame is then used by the *presence* block to find whether the target is present in the frame (in camera $N+1$, the target is always absent). As stated earlier, the *presence* block can be implemented using re-identification, and we will simulate different kinds of errors in re-identification. The tracking performance is reported in terms of MCTA metric in table 5. The related methods are taken from the dataset benchmark available at (Chen, Chen, and Huang 2014). These approaches are for multi-camera multi-target tracking and hence we have used our policy for multiple targets to make it a multi-target tracking approach. For ICT, the tracking performance is reflected only for the duration of transition till the right camera is predicted/selected. We have simulated errors in a typical re-identification pipeline from $0\%$ (no error) to $20\%$ (high error). In the table, the set-1 and set-2 show approx. same values for all percentage of errors in Re-ID and this is because the many of the target individuals are only in the single camera view and making the average value for all target high for all cases.

During training, we did not assume the camera network topology, and the policy learns the links. The transition time is recorded from the frame when the target disappears from current camera FOV to the camera frame where it is found present by the *presence* block. Table 3 show the time taken in the ground truth and identified by the policy for different camera transitions in dataset-3. The values in the table are the number of frames during camera transitions as found in ground truth and by the policy. Based on these transitions captured by the policy in this experiment, we have made a graph of the camera network. The predicted camera network topology is shown in figure 5 for sub-dataset 4 (5 cameras) and 3 (4 cameras). A black arrow in the figure shows the right prediction by the policy; red arrow show a link is predicted by the policy but it does not exist in ground truth. The figure shows that the policy learns most of the links and hence the target's current state helps in finding the next camera. There are false positives and false negatives due to scheduling queries of different target individuals. Camera schedules of target 2 in Figure 4 shows that the transition at time frame 3500 is missed and hence this has generated a false positive in DB-3 topology.

One crucial aspect of target tracking is the number of frames polled (the number of re-identification calls) because a large number of frames will incur a more considerable delay in locating the target. For single camera tracking (SCT), the target will be searched only in the same camera whereas,

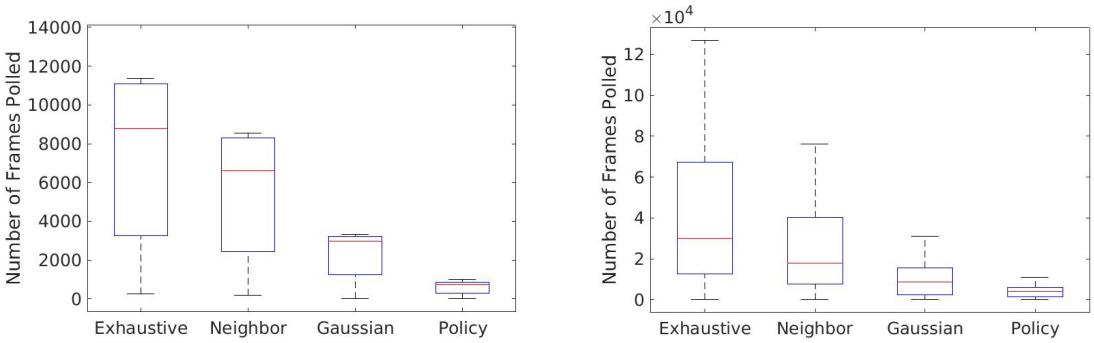


Figure 6: Boxplot of number of frames polled (metric F, see equation 9) on two datasets of NLPR dataset for our proposed policy and other baseline approaches.

Table 5: The table is showing average MCTA values for inter-camera tracking (ICT) and both SCT-ICT on the test set of NLPR_MCT dataset. The related approaches are multi-camera multi-target tracking approaches taken from the benchmark dataset (Chen, Chen, and Huang 2014). The last 5 rows show the MCTA values for the proposed approach with simulated re-identification errors from $0\%$ to $20\%$.

| | **Inter-camera tracking (ICT)** | | | | **Single-camera tracking + ICT** | | | |
|---|---|---|---|---|---|---|---|---|
| Approach | Set-1 | Set-2 | Set-3 | Set-4 | Set-1 | Set-2 | Set-3 | Set-4 |
| (Y. Cai 2014) | 0.9152 | 0.9132 | 0.5163 | 0.7152 | 0.8831 | 0.8397 | 0.2427 | 0.4357 |
| (Chen, Chen, and Huang 2014) | 0.7425 | 0.6544 | 0.7369 | 0.3945 | 0.7477 | 0.6561 | 0.2028 | 0.2650 |
| (Chen et al. 2014) | 0.6617 | 0.5907 | 0.7105 | 0.5703 | 0.6903 | 0.6238 | 0.0848 | 0.1830 |
| (Chen, Chen, and Huang 2014) | 0.3203 | 0.3456 | 0.1381 | 0.1562 | 0.8162 | 0.7730 | 0.1240 | 0.4637 |
| (Lee et al. 2018) | 0.9610 | 0.9264 | 0.7889 | 0.7578 | - | - | - | - |
| (W. Chen et al. 2017) | 0.835 | 0.703 | 0.742 | 0.385 | 0.8525 | 0.7370 | 0.4724 | 0.3778 |
| RL+ReID-0 | 0.8210 | 0.7498 | 0.9099 | 0.8993 | 0.8235 | 0.7503 | 0.9134 | 0.9118 |
| RL+ReID-5 | 0.8188 | 0.7481 | 0.8766 | 0.8137 | 0.7778 | 0.7064 | 0.7949 | 0.7338 |
| RL+ReID-10 | 0.8219 | 0.7511 | 0.8848 | 0.7140 | 0.7355 | 0.6635 | 0.6791 | 0.6769 |
| RL+ReID-15 | 0.8171 | 0.7468 | 0.7862 | 0.7128 | 0.7004 | 0.6160 | 0.6229 | 0.5879 |
| RL+ReID-20 | 0.8203 | 0.7519 | 0.7101 | 0.6625 | 0.6281 | 0.5323 | 0.5541 | 0.5288 |

for ICT, it will be searched across multiple cameras which will increase the number of frames to be polled to find the right camera where the target has appeared. The number of re-identification calls or the number of frames polled (F) are shown in figure 6. The proposed policy queries approx. 10 times less number of camera frames compared to other baseline methods (explained earlier in experiment-1).

*Experiment-3: Evaluation of Complete Pipeline:* In this experiment, we evaluate our proposed policy during ICT along with single-camera tracking (SCT) for multi-camera multi-target tracking. The experimental setup is same as experiment-2 but camera sequence during SCT is taken from policy as compared to the ground truth. Table 5 show the comparison of our approach with other related methods for this experiment. Unlike previous experiment, both SCT and ICT performance is compared. The re-identification is simulated with error from $0\%$ to $20\%$. The policy outperforms many methods and for set-3 and set-4, it has best best performance even at high error in re-identification.

## Conclusion

We have proposed a method to identify the camera schedule of a target's motion in a network of cameras. For this, we have learned a reinforcement learning based policy to select the next camera where the target is likely to appear at next time. ICT is very challenging and existing methods has to poll a large number of frames to search the target after a transition. We used spatial and temporal information of the target to learn a policy that selects the next camera. We showed through various experiments that the proposed approach is also useful to capture the transition time required by a target to move from one camera FOV to other camera.

## Acknowledgement

We gratefully acknowledge Infosys Center for Artificial Intelligence (CAI) at IIIT-Delhi for its partial support for conducting this research work. We would also like to thank Karthik Ayyar for having suggested the application of tracking in camera networks.

## References


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

Chen, X., and Bhanu, B. 2017. 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.

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

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

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

Daliyot, S., and Netanyahu, N. S. 2013. A framework for inter-camera association of multi-target trajectories by invariant target models. In Park, J.-I., and Kim, J., eds., *Computer Vision - ACCV 2012 Workshops*, 372–386. Berlin, Heidelberg: Springer Berlin Heidelberg.

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

Huang, T., and Russell, S. 1997. Object identification in a bayesian context. In *In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97*, 1276–1283. Morgan Kaufmann.

Javed, O.; Shafique, K.; Rasheed, Z.; and Shah, M. 2008. Modeling inter-camera space-time and appearance relationships for tracking across non-overlapping views. *Comput. Vis. Image Underst.* 109(2):146–162.

Khan, S., and Shah, M. 2003. 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.

Kuo, C.-H.; Huang, C.; and Nevatia, R. 2010. 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, 383–396. Berlin, Heidelberg: Springer-Verlag.

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

Makris, D.; Ellis, T.; and Black, J. 2004. 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, II–205–II–210 Vol.2.

Matei, B. C.; Sawhney, H. S.; and Samarasekera, S. 2011. Vehicle tracking across nonoverlapping cameras using joint kinematic and appearance features. In *CVPR 2011*, 3465–3472.

Pasula, H.; Russell, S.; Ostland, M.; and Ritov, Y. 1999. Tracking many objects with many sensors. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, 1160–1167. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Ristani, E., and Tomasi, C. 2015. Tracking multiple people online and in real time. In Cremers, D.; Reid, I.; Saito, H.; and Yang, M.-H., eds., *Computer Vision – ACCV 2014*, 444–459. Cham: Springer International Publishing.

Ristani, E.; Solera, F.; Zou, R.; Cucchiara, R.; and Tomasi, C. 2016. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision workshop on Benchmarking Multi-Target Tracking*.

Sunderrajan, S., and Manjunath, B. S. 2013. Multiple view discriminative appearance modeling with imcmc for distributed tracking. In *2013 Seventh International Conference on Distributed Smart Cameras (ICDSC)*, 1–7.

Sutton, R. S., and Barto, A. G. 1998. *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1st edition.

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

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

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

Zhang, S.; Zhu, Y.; and Roy-Chowdhury, A. 2015. Tracking multiple interacting targets in a camera network. *Computer Vision and Image Understanding* 134:64 – 73. Image Understanding for Real-world Distributed Video Networks.

Zheng, Z.; Zheng, L.; and Yang, Y. 2017. A discriminatively learned cnn embedding for person reidentification. *ACM Trans. Multimedia Comput. Commun. Appl.* 14(1):13:1–13:20.