Track Merging for Effective Video Query Processing

Daren Chao
University of Toronto
Toronto, Canada
drchao@cs.toronto.edu

Yueting Chen
York University
Toronto, Canada
ytchen@eecs.yorku.ca

Nick Koudas
University of Toronto
Toronto, Canada
koudas@cs.toronto.edu

Xiaohui Yu
York University
Toronto, Canada
xhyu@yorku.ca

***Abstract*—Video analysis frameworks supporting declarative queries are actively researched in recent years. A major prerequisite in executing such queries is the ability to accurately extract metadata at the frame level utilizing various computer vision algorithms, including object tracking models. Tracking models are of profound importance as they establish unique identifiers for the objects across frames.**

Despite the maturity of tracking algorithms, they still face challenges (such as occlusions, object glaze etc.) which diminish their quality and accuracy. This gives rise to the track fragmentation problem in which a single track is fragmented into multiple smaller tracks. This impacts downstream temporal querying applications degrading query accuracy.

In this paper, we propose an algorithm, TMerge for identifying and merging fragmented tracks that constitutes a pre-processing step during data ingestion for video query processing. The algorithm exploits the properties of the problem and utilizes a sampling methodology that significantly reduces the time required to pre-process and ingest the video sequence.

We comprehensively describe and analyze our proposals utilizing real data sets and also present the results of a detailed experimental evaluation varying parameters of interest. We demonstrate performance savings of up to two orders of magnitude without loss in accuracy.

***Index Terms*—video query processing, polyonymous tracks, track merging, multi-armed bandits, Thompson sampling.**

I. Introduction

Advances in Computer Vision (CV) and Deep Learning (DL) offer highly sophisticated algorithms for video analysis, such as classifying video frames [1], [2], detecting objects and tracking them across video frames [3]–[5] as well as recognizing actions and interactions among detected objects. Such algorithms form the foundation of a new generation of data management and query processing systems and techniques that support structured query processing over videos [6]–[8]. Recently an array of research prototypes and associated algorithms demonstrate advanced functionalities encompassing these algorithms [7], [9]. Such systems are able to answer declarative queries involving query-specified objects [6], locate sequences of frames involving objects interacting in certain ways [10], [11], detect groups of objects satisfying certain query-specified temporal patterns, possibly with additional spatial constraints [12], [13]. They demonstrate promising results and deliver functionalities that were not available before on streaming videos or over large video repositories.

A major prerequisite in delivering such functionalities is the ability to accurately extract metadata that uniquely identifies and tracks objects across video frames. This ability is delivered via object tracking algorithms [3]–[5], [14] that are deployed along with other applicable algorithms from CV (e.g., object detection algorithms [1], [2]) during metadata extraction. Accurate object tracking is imperative for query correctness, as object tracking establishes a unique identifier for a detected object across frames. Typically object tracking algorithms accept an initial set of detected objects from a video and track the movement of these objects across video frames, representing object locations by bounding boxes (*BBoxes*). Ideally, the same physical object across frames within a pre-defined time range should form a single *track* and be assigned a distinct tracking identifier (*tracking ID, TID*). The ability to derive such distinct tracking IDs is fundamental to accurate query processing.

Very often, however, well established state-of-the-art object tracking algorithms [14], [15] exhibit reduced accuracy due to various reasons, such as object occlusion and unfavorable lighting conditions (e.g., glare). More specifically, depending on the camera position it is common for an object to be occluded by other objects in the visual range of the camera across a number of frames. In such cases, depending on the physical proximity of the objects the tracking algorithm may assign two different track IDs to the same object before and after the occlusion. A similar situation may arise due to specific lighting conditions; object glare may confuse detection for a number of frames and as such the tracking algorithm may assign new tracking identifiers to objects detected after the glare vanishes. Figure 1 illustrates a concrete example. The ground truth (GT) track A of a red sedan is reported by the tracking algorithm as two separate tracks, each with a distinct ID, α and β. This is due to occlusion by an SUV between the two tracks. Such occlusions give rise to the *track fragmentation problem* reported in the literature [16]. Notice that depending on the scene conditions (number of objects, camera positions, etc) the track fragmentation problem can be aggravated arbitrarily. If such a problem is not remediated, the accuracy of any downstream query processing will suffer.

To aid accurate query processing, it is thus of critical importance to *identify and merge* (assigning them the same ID) those tracks that actually correspond to the same object. We refer to such tracks as *polyonymous tracks*. Existing works [4], [5], [14], [15] consider identifying polyonymous tracks as a part of tracking algorithms, where the main objective is to handle occlusions using tracking algorithms. In this paper, we consider the problem as a post-processing step with tracking results already obtained, which could further improve the quality of tracking results and is orthogonal to existing approaches. A brute-force approach to this problem would be to compute the pairwise similarity (for a suitable definition of similarity) of all tracks and consider the top-ranked pairs (or pairs with similarity exceeding a certain threshold) as candidates to be merged. Such merging can take place automatically or optionally be subject to further human inspection for increased accuracy. Different notions of similarity can be adopted between a pair of tracks; for example, similarity can be computed as some aggregate measure of frame-level pairwise distances between pairs of frames across tracks. One possible approach to obtain the frame-level distance could be to utilize Re-Identification (ReID) models [17]–[20] based on features extracted from the BBoxes containing the object(s) of interest in the pair of frames.

However, such a brute-force approach to computing frame-level similarities for all pairs of frames in all pairs of tracks can be prohibitively expensive in practice. For example, each video feed in the commonly used benchmark dataset, MOT-17 [21], has an average of 825 frames, 11,867 BBoxes and 8,689,117 frame-level BBox pairs to compute. The brute-force approach has to extract features from all BBoxes and calculate the distance between all BBox pairs – more than 3 minutes to process a half-minute video feed (system configuration detailed in § V-B). Clearly, this approach is not scalable to handle scenarios with large video repositories, such as the analysis of surveillance videos or videos captured by autonomous vehicles (capturing cars on highways or pedestrians at intersections, etc.), where the inefficiency of the brute-force approach for identifying polyonymous tracks necessitates the investment of additional computing resources to achieve target processing capabilities. To solve this problem at scale and provide a general solution for videos of arbitrary length, more efficient solutions are needed.

In this paper, we propose algorithms that can drastically improve the efficiency of identifying (and subsequently merging) polyonymous tracks. Such algorithms are intended as a data pre-processing step after tracking algorithms have been applied but before downstream query processing takes place. They are periodically invoked during metadata extraction, and are general enough to support most, if not all, video query processing systems [7], [11]–[13]. A key observation driving the development of our algorithms is that only a small fraction of all track pairs are truly polyonymous tracks. Therefore, instead of spreading the computational effort evenly across all pairs of tracks, we should invest more on those pairs that are more promising to yield polyonymous tracks. For this purpose, we cast the problem of identifying polyonymous tracks as one of decision-making under uncertainty, and devise an algorithm, named TMerge, that can quickly focus on more promising track pairs for more extensive examination (by biasing the invocation of ReID models towards pairs of frames from such track pairs). In particular, we propose a variant of Thompson Sampling [22], [23], balancing between exploration (i.e., exploring more pairs of tracks) vs. exploitation (i.e., focusing computation on specific pairs of tracks).

We conduct extensive experiments on benchmark datasets to evaluate the efficiency and effectiveness of TMerge. Our results demonstrate that TMerge achieves 10x to 100x speedup over other applicable approaches and is able to identify 95% of all polyonymous tracks by examining only a small number of track pairs.

In summary, we make the following contributions.

- We initiate the study of identifying polyonymous tracks to support structured video query processing and model our problem as an instance of the Multi-Armed Bandit problem, with an intent to effectively allocate computational resources for fast identification of polyonymous tracks in large video repositories.
- We propose the TMerge algorithm based on Thompson Sampling to solve the polyonymous track identification problem, striking a balance between exploitation and exploration in a principled way. We demonstrate that our algorithm is able to yield highly accurate results with a limited number of invocations of expensive ReID models. TMerge is an essential first step during metadata extraction for any video analytics system and is able to improve the accuracy of query processing by improving the accuracy of tracking identifiers utilized by such systems.
- We report results of thorough experimental studies on real datasets utilizing state-of-the-art tracking algorithms to validate the performance of the proposed approach. Our results indicate that TMerge can dramatically reduce 90% of the overhead of applying ReID models over other applicable approaches, without loss of accuracy.

This paper is organized as follows. We formally define the problem in §II, and present a baseline approach in §III. We detail the proposed approach TMerge in §IV, and present the experimental results in §V. §VI discusses related work, and §VII concludes this paper.

II. Problem Definition

A video is a sequence of frames $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$, where $|\mathcal{V}|$ is the number of frames (length) of a video and can be fixed or unbounded. If $|\mathcal{V}|$ is unbounded we refer to $\mathcal{V}$ as a video stream. From the perspective of object detection, a frame may hold a variety of different object types and each object type may have multiple instances in a frame. Object tracking algorithms [3]–[5] accept an initial set of detected objects (along with associated BBoxes) from a video and track the movement of these objects across video frames. The sequence of the same physical object across frames forms a single track which is assigned a unique *tracking identifier (TID)*. The tracking results are crucial metadata for many automated video analysis tools, such as video query processing systems. However, as shown in Figure 1, due to track fragmentation a *ground truth* (GT) track may be mistakenly reported as two separate tracks (i.e., polyonymous tracks) and assigned two distinct IDs. We aim to identify (and then merge) the polyonymous tracks throughout the video.

We observe that the polyonymous track problem happens

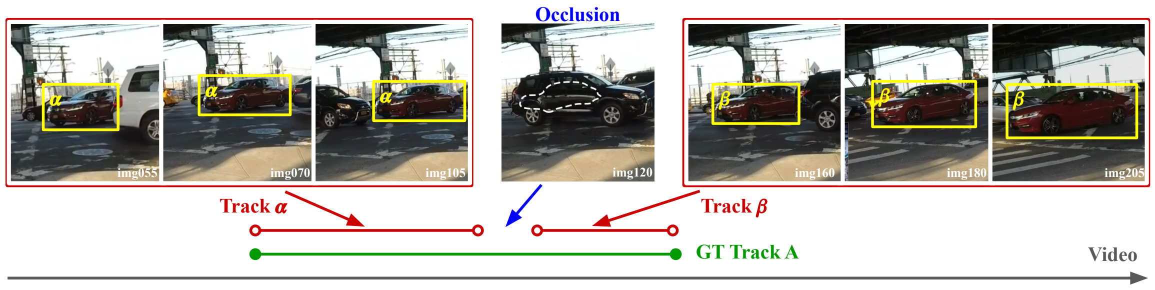

Fig. 1. An example of track fragmentation: due to occlusion, the ground truth (GT) track A is reported as two separate tracks with distinct tracking IDs α and β by the tracking algorithm. Each red rectangle represents a track, and a yellow rectangle represents a BBoxes. Only a portion of the track's BBoxes is depicted in the figure for readability.

in a short period of time (frames). Thus, a straightforward way is to calculate the similarity between all track pairs in a short period to determine which track pairs are more likely to be polyonymous. However, since the number of video frames could be large or even unbounded, processing the entire video is not practical. We partition the video into a series of overlapping windows, W_1, W_2, ..., according to the chronological order of the frames. Each window has a fixed length of L frames. To avoid overlooking any possible polyonymous track pairs or visiting any track pair more than once, we half-overlap every two neighbor windows, as demonstrated in Fig. 2. For example, let W_c be the c-th window: W_c and W_{c-1} overlap by $\frac{L}{2}$ frames. We commence processing each individual window in order of succession. The tracking algorithm is applied on W_c when it is being processed. We denote by T_c the set of all tracks identified by the tracking algorithm in the first $\frac{L}{2}$ frames of window W_c:

$$T_c = \{t_{c,1}, t_{c,2}, \ldots, t_{c,|T_c|}\},$$

where $t_{c,k}$ represents the track with TID k in window W_c. For a track $t_{c,k}$, the sequence of its associated BBoxes is denoted as $B_{t_{c,k}}$:

$$B_{t_{c,k}} = \{b^1_{\text{c,k}}, b^2_{\text{c,k}}, ..., b^{|t_{c,k}|}_{\text{c,k}}\},$$

where $b^m_{\text{c,k}}$ represents the content in the m-th BBox of track $t_{c,k}$, and $|t_{c,k}|$ is the number of BBoxes associated with track $t_{c,k}$.

After obtaining the tracking results for a window, we seek to identify the polyonymous tracks. Let P_c denote the set of the track pairs for window W_c. To avoid missing any polyonymous track pairs, we form P_c by associating not only the track pairs inside W_c, but also those across W_c and its previous window W_{c-1}, that is,

$$\begin{aligned} P_c =& \{p_{i,j} \mid \forall t_i, t_j \in T_c : t_i \neq t_j\} \\ & \cup \{p_{i,j} \mid \forall t_i \in T_c,\ \forall t_j \in T_{c-1} : t_i \neq t_j\} \\ =& \{p_{i,j} \mid \forall t_i \in T_c,\ \forall t_j \in T_c \cup T_{c-1} :\ t_i \neq t_j\}. \end{aligned} \tag{1}$$

where $p_{i,j}$ represents the pair of tracks with TID t_i and t_j. To prevent a GT track from spanning more than two windows, we set $L \geq 2L_{\text{max}}$, where L_{max} represents the maximum number of frames that a GT track can span. By splitting windows and extracting P_c in this manner, it can also handle the condition

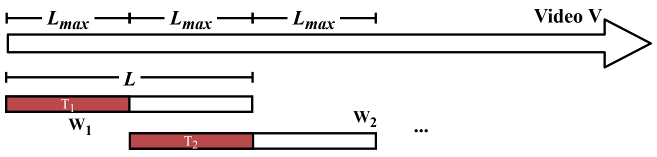

Fig. 2. The original video (stream) is partitioned into windows with $L = 2L_{\text{max}}$.

that occlusions happen very frequently (for example, the ground truth track is split into more than two tracks inside the same window), since P_c contains all possible pairs in the window W_c. Figure 2 presents an example of the video that is partitioned into windows with $L = 2L_{\text{max}}$. L is a hyper-parameter and can be determined either empirically by domain knowledge or by the requirements of downstream query processing. We will evaluate the effect of the length of L with experiments in later sections.

Some of the pairs in P_c may be polyonymous and some could be truly distinct tracks. Deciding whether a pair is indeed polyonymous can be determined utilizing a manual inspection approach. Let P^*_c be the set of track pairs in P_c that are polyonymous, namely

$$P^*_c = \{p_{i,j} \in P_c\}_{t_i \sim t_j}, \tag{2}$$

where $t_i \sim t_j$ represents that t_i and t_j refer to the same GT track—they are a pair of polyonymous tracks. Evidently $P^*_c \subset P_c$. Since the state-of-the-art tracking algorithms exhibit good accuracy [3]–[5] we expect P^*_c to be much smaller than P_c.

For a given subset of track pairs $\hat{P}_c \subseteq P_c$, let $\hat{P}_c \cap P^*_c$ be the set of all polyonymous tracks in $\hat{P}_c$. A recall rate is used to quantify the proportion of all real polyonymous track pairs included in $\hat{P}_c$, defined as follows:

$$REC(\hat{P}_c) = \frac{|\hat{P}_c \cap P^*_c|}{|P^*_c|}. \tag{3}$$

If we set $\hat{P}_c = P_c$, then $REC(\hat{P}_c) = 1$. However, inspecting all tracks in P_c for every window W_c manually would be costly and infeasible due to their sheer volume. Thus, we seek to minimize the size of $\hat{P}_c$ processed in order to keep the

TABLE I
DESCRIPTION OF COMMONLY-USED NOTATIONS.

Notation	Description
$\mathcal{V}$	the sequence of frames in the target video (stream).
W_c	the c-th window that the target video is partitioned into.
L	window size (in frames).
$L_{\max}$	the maximum size (in frames) of a GT track.
T_c	the set of tracks of window W_c.
$t_{c,k}$	the track with TID k in W_c.
$B_{t_{c,k}}$	the sequence of BBoxes of track $t_{c,k}$.
$b^m_{\text{c,k}}$	the content in the m-th BBox of track $t_{c,k}$.
P_c	the set of the track pairs for window W_c.
$p_{i,j}$	the pair of tracks with TID t_i and t_j.
P^*_c	the set of track pairs in P_c that are polyonymous.
$\hat{P}^*_{c\|K}$	the candidate subset of track pairs with size $\lceil K{\cdot}\|P_c\|\rceil$ that maximizes the number of polyonymous track pairs.
K	a decimal ($K{\in}[0,1]$) that specifies the size of $\hat{P}^*_{c\|K}$.
$f(b)$	the feature vector of a BBox b extracted by the ReID model.
$d(b_1,b_2)$	the distance between the feature vectors of BBoxes $b_1\&b_2$.
$\tilde{d}(b_1,b_2)$	the normalized BBox pair distance.
$s_{i,j}$; $\tilde{s}_{i,j}$	the score and the normalized score of a track pair $p_{i,j}$.
$BP'_{i,j}$	a subset of BBox pairs extracted from B_{t_i} and B_{t_j}.
$s'_{i,j}$	the track pair score estimation of $p_{i,j}$ w.r.t. $BP'_{i,j}$.
$\tau_{\max}$	the maximum number of desired iterations.
$S_{i,j}, F_{i,j}$	the shape parameters of Beta distribution for $p_{i,j}$.
$DisS_{i,j}$	the spatial distance of a track pair $p_{i,j}$.
$\Phi(b)$	the coordinates of the center of a BBox b.
thr_S	a hyper-parameter for BetaInit.

cost[1] manageable while seeking to maximize REC. Let $\hat{P}_{c|K}$ denote a set with $|\hat{P}_{c|K}| = \lceil K{\cdot}|P_c|\rceil$, where $K \in [0,1]$. This problem presents an interesting *trade-off*: identifying $\hat{P}_{c|K}$ that minimizes K while maximizing REC.

For a given value of K, we are interested to identify a $\hat{P}^*_{c|K} \subseteq P_c$, with the maximum number of polyonymous track pairs, namely

$$\hat{P}^*_{c|K} = \max_{\hat{P}_{c|K}} REC(\hat{P}_{c|K}), \quad \text{given } K \in [0,1] \tag{4}$$

In §IV we will present an algorithm to efficiently obtain $\hat{P}^*_{c|K}$ for a specified value K.

III. A Baseline Approach

In this section, we introduce a *baseline* approach that is capable of identifying polyonymous track pairs in P_c. Such an approach is based on the observation that if two tracks are polyonymous (i.e., they correspond to the same GT track), the content of their associated BBoxes should be similar, such as the tracks α and β in Figure 1 that both feature a red sedan; otherwise, we expect that the content should be less related.

ReID models [17]–[19] are trained to classify whether two input images[2] contain the same object. Specifically, ReID models map two images of the same object to nearby vectors (i.e., feature vectors) in the feature space; if the images are of different objects, their feature vectors are further apart [19]. Let $f(b)$ be the feature vector of a BBox b extracted by a ReID model; the feature vectors of BBoxes with similar-looking objects have a smaller Euclidean distance. The distance between two BBoxes b_1 and b_2, $d(b_1,b_2)$, is the Euclidean distance of their feature vectors $f(b_1)$ and $f(b_2)$ obtained through a ReID model.

We now define the track pair score when measuring the distance between two tracks. Let $s_{i,j}$ be the score of a track pair $p_{i,j}$, which is determined by aggregating the pairwise BBox distances of the corresponding object BBoxes across the two tracks. The lower the score, the more likely it is a polyonymous track pair. Let $B_{t_i} \times B_{t_j}$ represent the set of all the BBox pairs across tracks t_i and t_j.

Definition 3.1 (Track Pair Score): The track pair score between a pair of two tracks t_i and t_j, denoted as $s_{i,j}$, is the mean value of the pairwise BBox distances across the two tracks,

$$s_{i,j} = \text{avg}_{\forall(b_\alpha,b_\beta)\in B_{t_i}\times B_{t_j}} \left\{d\left(b_\alpha,b_\beta\right)\right\}. \tag{5}$$

After calculating the scores of all track pairs in P_c through Equation (5), the following ranking is derived:

$$R_c = \{p_{i_k,j_k} : p_{i_k,j_k} \in P_c\}_{\forall\alpha\leq\beta,\ s_{i_\alpha,j_\alpha}\leq s_{i_\beta,j_\beta}}, \tag{6}$$

where $k{=}1,...,|P_c|$. Then, the track pairs with top-$\lceil K{\cdot}|P_c|\rceil$ lowest scores are considered as the polyonymous track pair candidates, denoted as $\hat{P}^*_{c|K}$,

$$\hat{P}^*_{c|K} = \{p_{i_k,j_k} : p_{i_k,j_k} \in R_c\}_{k=1,2,...,\lceil K\cdot|P_c|\rceil}. \tag{7}$$

The baseline approach is presented in Algorithm 1. The input consists of P_c, the BBoxes of the tracks involved in P_c, and K. The algorithm iterates over all BBox pairs to calculate the score for each track pair (Lines 1 to 5) and considers track pairs with the top-$\lceil K{\cdot}|P_c|\rceil$ lowest scores as the polyonymous track pair candidates (Lines 6 to 7).

Algorithm 1: Baseline

```
Input: Track Pairs $P_c$ and BBoxes Information of
       each track included in $P_c$; $K$;
Output: Top-$\lceil K\cdot|P_c|\rceil$ Polyonymous Track Pair
        Candidates: $\hat{P}^*_{c|K}$;
1 for each track pair $p_{i,j} \in P_c$ do
2     for each BBox pair $(b_\alpha, b_\beta) \in B_{t_i} \times B_{t_j}$ do
3         Extract feature vectors $f(b_\alpha)$ and $f(b_\beta)$ through
          the ReID model.
4         $d(b_\alpha, b_\beta) = Euclidean(f(b_\alpha), f(b_\beta))$.
5     $s_{i,j} = \textbf{avg}_{\forall(b_\alpha,b_\beta)\in B_{t_i}\times B_{t_j}} \{d(b_\alpha,b_\beta)\}$.
6 $R_c = \{p_{i_k,j_k} : p_{i_k,j_k} \in P_c\}_{\forall\alpha<\beta,\ s_{i_\alpha,j_\alpha}\leq s_{i_\beta,j_\beta}}$.
7 $\hat{P}^*_{c|K} = \{p_{i_k,j_k} : p_{i_k,j_k} \in R_c\}_{k=1,2,...,\lceil K\cdot|P_c|\rceil}$.
```

$REC(\hat{P}^*_{c|K})$ represents the recall of the baseline algorithm for a specific K on the window W_c. By averaging the recall over all windows of all videos in a dataset, we can get the overall recall (REC) of the algorithm on that dataset. Setting window length $L{=}2000$, by varying the value of input K, one obtains the REC-K curves of Figure 3. The chance of missing polyonymous tracks can be computed as $1 - REC$. The figure

[1]"Cost" here refers to the time and/or labor expense to manually inspect which track pairs are polyonymous; elsewhere in this paper, cost has a consistent meaning for computing (or estimating) similarities for all track pairs, i.e., finding $\hat{P}^*_{c|K}$.

[2]For some ReID models [24], two fixed-length image sequences may be accepted as input. Our entire discussion and techniques equally apply to this case as well.

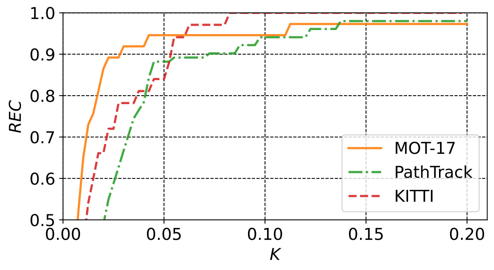

Fig. 3. *REC*-*K* curves on the videos of three datasets.

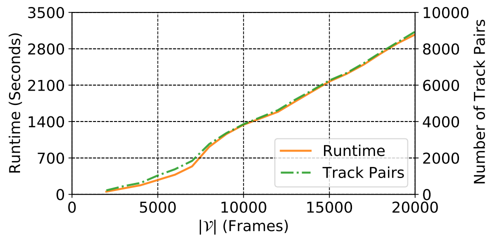

Fig. 4. Runtime for processing videos of varying lengths and the number of track pairs accumulated. The window size is set to be 2000 frames.

illustrates the trade-off: higher REC requires increasing K, i.e. requiring more overhead to process $\hat{P}^*_{c|K}$ via a manual inspection procedure. For the tracking results of Tracktor [5] on the MOT-17 dataset [21], in which each window has an average of 400 track pairs, with 8 polyonymous tracks (that is of 2% polyonymous rate) on average, the REC surpasses 0.95 when K is greater than 0.05. For the tracking results of Tracktor [5] on the PathTrack dataset, [25], in which each window has an average of 470 track pairs, the REC exceeds 0.95 when K is greater than 0.085. It is evident that in these experiments a small value of K, with $\hat{P}^*_{c|K}$ less than 10% of P_c, is sufficient to achieve a high REC value. In unknown environments, a sample of representative videos can be adopted to calibrate the value of K. In all of our experiments (reported later in §V) utilizing diverse datasets, a value of K less than 0.05 consistently achieved accuracy over 0.9.

However such an approach is not able to scale when dealing with large video repositories or video streams. Fig. 4 presents the time required and the number of track pairs accumulated to process videos from the dataset PathTrack using Algorithm 1, as the length of the video increases. Each window has an average of 145 tracks and each track contains an average of 105 BBoxes. As seen in the figure, the time and the number of track pairs both increase dramatically and synchronously as the video length grows. The reason is that Algorithm 1 extracts the feature vectors of all BBoxes involved in all track pairs and calculates the BBox distances for all the BBox pairs (Lines 1 to 5 of Algorithm 1). It is evident that this style of processing is not feasible for realistic video lengths. In the next section we propose a more efficient algorithm for estimating $\hat{P}^*_{c|K}$ for set values of K exhibiting REC comparable to that of Algorithm 1.

IV. TMerge

In this section, we propose an efficient algorithm, called TMerge, for obtaining top-$\lceil K\cdot|P_c|\rceil$ polyonymous track pair candidates $\hat{P}^*_{c|K}$ in P_c.

A. Basic Idea

Unlike the baseline algorithm that ranks all the track pairs by computing their scores, TMerge estimates the ranking utilizing a sampling methodology. Generally, TMerge randomly extracts a subset of BBox pairs, namely $BP'_{i,j} \subseteq B_{t_i} \times B_{t_j}$, for each track pair $p_{i,j} \in P_c$, and calculates the distances of the extracted BBox pairs rather than the whole set of BBox pairs in each track pair. Then, TMerge uses the mean of the distances of the extracted BBox pairs $BP'_{i,j}$ to estimate the track pair score of $p_{i,j}$, namely $s'_{i,j}$,

$$s'_{i,j} = \text{avg}_{\forall(b'_\alpha, b'_\beta)\in BP'_{i,j}} \left\{ d\left(b'_\alpha, b'_\beta\right)\right\}, \tag{8}$$

where obviously $\mathbb{E}[s'_{i,j}] = s_{i,j}$.

The extraction of the subset $BP'_{i,j}$ for each track pair $p_{i,j} \in P_c$ is an iterative sampling process: at each iteration, we first sample a track pair from P_c, and then randomly choose a BBox pair from all BBox pairs in this track pair. After τ iterations, TMerge obtains the subset $BP'_{i,j}$ for each track pair $p_{i,j}$, where $\sum_{p_{i,j}\in P_c} |BP'_{i,j}| \leq \tau$. However, we do not want to sample track pairs uniformly from P_c throughout iterations. We observe that track pairs corresponding to the same object (i.e., polyonymous track pairs), which we are aiming to identify, account for a small proportion (<1%) of P_c. As a result, we intend to bias the sampling toward track pairs with lower scores [3], which have a higher probability of being polyonymous track pairs. TMerge maintains a prior distribution on the score $s_{i,j}$ for each track pair $p_{i,j}$, which determines the chance of each track pair being sampled at each iteration. This prior is updated after each iteration by considering the newly computed distances, and is applied to bias the sampling in the next iteration.

B. The Algorithm

We first discuss how to choose an appropriate prior distribution, and then provide a detailed description of the proposed sampling method. We first normalize all BBox pair distances (defined in §III) into the range [0, 1]. Let $\tilde{d}(b_\alpha, b_\beta)$ denote the normalized BBox pair distance of b_α, b_β. Consequently, the track pair score (defined in Equation (5)) is also normalized into the range [0, 1]. Let $\tilde{s}_{i,j}$ represent the normalized track pair score for $p_{i,j}$.

We show that the iterative computation of BBox pair distances through repeated sampling can be modeled as a sequence of Bernoulli trials. For each track pair $p_{i,j}$, we assume that the normalized distances of all BBox pairs come from the same but unknown distribution, with the normalized track pair score $\tilde{s}_{i,j}$ as the mean,

$$\mathbb{E}\left[\tilde{d}(b_\alpha, b_\beta)\right] = \tilde{s}_{i,j}. \quad \forall(b_\alpha, b_\beta) \in B_{t_i} \times B_{t_j} \tag{9}$$

[3]Note that two BBoxes containing the same object will have lower Euclidean distance than two BBoxes containing different objects.

For a BBox pair randomly selected from any track pair $p_{i,j}$ with normalized distance $\tilde{d}$, we conduct a Bernoulli trial utilizing $\tilde{d}$ as the success probability, obtaining the output $r \in \{0, 1\}$. The Bernoulli trials on BBox pairs containing similar objects (i.e., those with lower scores) will have a higher probability to have output $r = 0$. The expectation of the Bernoulli output (i.e., the probability of observing $r = 1$) is equal to the mean of the unknown distribution (i.e., the normalized track pair score $\tilde{s}_{i,j}$),

$$\begin{aligned}\mathbb{E}[r] = Pr(r = 1) &= \int_0^1 Pr\left(r = 1 \mid \tilde{d}\right) \mathsf{f}_{i,j}\left(\tilde{d}\right) \mathrm{d}\tilde{d} \\ &= \int_0^1 \tilde{d}\; \mathsf{f}_{i,j}\left(\tilde{d}\right) \mathrm{d}\tilde{d} \\ &= \mathbb{E}[\tilde{d}] = \tilde{s_{i,j}},\end{aligned}$$

where $\mathsf{f}_{i,j}$ denotes the (unknown) probability density function of the unknown distribution for the normalized distances in track pair $p_{i,j}$. Thus, we have transformed the sequence of BBox pair distances (for a single track pair) computed across iterations into a sequence of Bernoulli trials with the same expectation.

A natural choice for the prior distribution is the Beta distribution [22], which is a conjugate prior to the Bernoulli distribution. Thus, after observing a new Bernoulli output r, the posterior distribution is simply $Be(S+1, F)$ or $Be(S, F+1)$, depending on whether the output r equals to 1 or 0. In other words, when obtaining a BBox pair distance $\tilde{d}$ in a new iteration, Beta distribution enables the newly observed Bernoulli trial r to *easily* update the prior. The mean of the Beta distribution is $\frac{S}{S+F}$, indicating that after some iterations, the sequence of Bernoulli trials corresponding to BBox pair distances between track pairs with more r=0 outputs have Beta distributions with a lower mean. Thus, track pairs whose BBox pairs contain the same (or similar) objects have corresponding Beta distributions with a lower mean, and the corresponding BBox pair distances are smaller. As a result, using the Beta distribution as a prior can assist our sampling process, guiding (biasing) the sampling process towards track pairs with lower scores.

Algorithm 2 presents TMerge. The input consists of P_c, the BBoxes of the tracks involved in P_c, the value of K, and the maximum number of desired iterations τ_{max}. At the start, TMerge initializes a Beta distribution for each track pair $p_{i,j}$ (Line 1), namely $Be(S_{i,j}, F_{i,j})$. One way to initialize it is to set $S_{i,j} = F_{i,j} = 1$. An improved initialization method for Beta distributions will be introduced as Algorithm 3 in the sequel. Then, TMerge repeats the steps in Lines 4-14 for τ_{max} iterations. At the τ-th iteration, it samples a value $\theta^{\tau}_{i,j}$ from the Beta distribution $Be(S_{i,j}, F_{i,j})$ of each track pair $p_{i,j}$ (Lines 4-5), and chooses the track pair with the smallest value, namely p_{i_τ,j_τ}, (Line 6)

$$p_{i_\tau,j_\tau} = \arg\min_{p_{i,j}} \theta^{\tau}_{i,j}.$$

Then, it randomly samples a BBox pair $(b_{\alpha_\tau}, b_{\beta_\tau}) \in B_{t_{i_\tau}} \times B_{t_{j_\tau}}$ without replacement from the track pair p_{i_τ,j_τ}, extracts the feature vectors $f(b_{\alpha_\tau})$ and $f(b_{\beta_\tau})$ with the ReID model, and

Algorithm 2: TMerge

```
Input: Track Pairs $P_c$ and BBoxes Information of
       each track included in $P_c$; $K$; $\tau_{\text{max}}$;
Output: Estimated Top-$\lceil K{\cdot}|P_c|\rceil$ Polyonymous Track
        Pair Candidates: $\hat{P}^*_{c|K}$;
1  Initialize a Beta Distribution $Be(S_{i,j}, F_{i,j})$ for each
     track pair $p_{i,j} \in P_c$ through Algorithm 3.
2  $P_{\text{skip}} = \phi$.                          # pruned track pairs
3  for $\tau = 1, ..., \tau_{max}$ do
4      for each track pair $p_{i,j} \in P_c \backslash P_{skip}$ do
5          Sample a value $\theta^{\tau}_{i,j}$ from $Be(S_{i,j}, F_{i,j})$.
6      $p_{i_\tau,j_\tau} = \arg\min_{p_{i,j}} \theta^{\tau}_{i,j}$.
7      Randomly select a BBox pair
         $(b_{\alpha_\tau}, b_{\beta_\tau}) \in B_{t_{i_\tau}} \times B_{t_{j_\tau}}$ from track pair $p_{i_\tau,j_\tau}$
         without replacement.
8      $\tilde{d}_\tau = \tilde{d}(b_{\alpha_\tau}, b_{\beta_\tau})$.
9      Perform a Bernoulli trial with success probability
         $\tilde{d}_\tau$ and observe output $r_\tau$.
10     if $r_\tau = 1$ then
11         $S_{i_\tau,j_\tau}$ = $S_{i_\tau,j_\tau}$ + 1.
12     else if $r_\tau = 0$ then
13         $F_{i_\tau,j_\tau}$ = $F_{i_\tau,j_\tau}$ + 1.
14     Update $P_{\text{skip}}$ through Algorithm 4.
15 $\hat{P}^*_{c|K} = \{p_{i,j} \in P_c$ with $\lceil K{\cdot}|P_c|\rceil$ lowest $\frac{S_{i,j}}{S_{i,j}+F_{i,j}}\}$.
```

calculates the normalized distance $\tilde{d}_\tau = \tilde{d}(b_{\alpha_\tau}, b_{\beta_\tau})$ (Lines 7-8). As an optimization, if either of the BBoxes' feature vectors has been extracted in previous iterations it can be *reused* in this iteration. The distance is then normalized to be in the interval $[0, 1]$, denoted as $\tilde{d}_\tau$. After that, TMerge performs a Bernoulli trial using the normalized distance $\tilde{d}_\tau$ as the success probability and obtains the binary output r_τ (Line 9). It then updates the Beta distribution based on the Bernoulli output r_τ (Lines 10-13): if r_τ=1, increase S_{i_τ,j_τ} by 1; otherwise, if r_τ=0, increase F_{i_τ,j_τ} by 1.

Moreover, at the end of each iteration (Line 14), we update P_{skip} (initialized in Line 2), which stores the pruned track pairs that cannot be part of $\hat{P}^*_{c|K}$ in the following iterations, and will be used in Algorithm 4 to further improve the algorithm efficiency. After the loop, TMerge returns the track pairs with the top-$\lceil K{\cdot}|P_c|\rceil$ lowest mean values of their Beta distributions as the estimated top-$\lceil K{\cdot}|P_c|\rceil$ polyonymous track pair candidates $\hat{P}^*_{c|K}$ (Line 15),

$$\hat{P}^*_{c|K} = \{p_{i,j} \in P_c \text{ with } \lceil K{\cdot}|P_c|\rceil \text{ lowest } \frac{S_{i,j}}{S_{i,j} + F_{i,j}}\}.$$

C. *Initializing Beta Distributions*

Next, we introduce a method, BetaInit, for initializing Beta distributions utilising our inherent knowledge of track pairs. In particular, BetaInit exploits correlations between the score of a track pair and the distance between bounding boxes in the frame sequence. Recall that $p_{i,j}$ represents the pair of tracks t_i, t_j. We define the *spatial distance* of a track pair $p_{i,j} \in P_c$,

Algorithm 3: BetaInit

```
    Input: $P_c$; $thr_S$
    Output: $\forall p_{i,j} \in P_c : Be(S_{i,j}, F_{i,j})$
1   for each track pair $p_{i,j} \in P_c$ do
2       Calculate the spatial distance of $p_{i,j}$: $DisS_{i,j}$.
3       $S_{i,j}$, $F_{i,j}$ = 1, 1.
4       if $DisS_{i,j} < thr_S$ then
5           $F_{i,j} = F_{i,j} + 1$.
6       Initialize a Beta Distribution $Be(S_{i,j}, F_{i,j})$ for $p_{i,j}$.
```

Algorithm 4: ULB

```
    Input: $P_c$; $P_{\text{skip}}$; $\tau$;
    Output: Update $P_{\text{skip}}$;
1   for each track pair $p_{i,j} \in P_c$ do
2       $n_{i,j} = |\{\forall \lambda = 1, ..., \tau : p_{i_\lambda, j_\lambda} = p_{i,j}\}|$.
3       $U_{i,j} = \sqrt{\frac{2 \log \tau}{n_{i,j}}}$.
4       $\tilde{s}'_{i,j} = \sum_{\forall \lambda = 1,...,\tau : p_{i_\lambda, j_\lambda} = p_{i,j}} \tilde{d}_\tau$.
5   for each track pair $p_{i,j} \in P_c \backslash P_{skip}$ do
6       if $|\{\forall p_{i',j'} \in P_c : \tilde{s}'_{i,j} + U_{i,j} > \tilde{s}'_{i',j'} - U_{i',j'}\}| \leq K{-}1$ or
        $|\{\forall p_{i',j'} \in P_c : \tilde{s}'_{i,j} - U_{i,j} > \tilde{s}'_{i',j'} + U_{i',j'}\}| \geq K$
        then
7           Add $p_{i,j}$ into $\text{P}_{\text{skip}}$.
```

namely $DisS_{i,j}$, as the Euclidean distance between the center points of the last BBox of track t_i and the first BBox of track t_j. For example, for the pair of tracks (t_i, t_j),

$$DisS_{i,j} = \left\| \Phi(b_{\text{i}}^{|t_i|}) - \Phi(b_{\text{j}}^1) \right\|_2,$$

where $\Phi(b)$ represents the coordinates of the center of b; $b_{\text{i}}^{|t_i|}$ and b_{j}^1 represent the last BBox of track t_i and the first BBox of t_j respectively. We conducted experiments on several datasets which showed that the Pearson correlation coefficient [26] between the scores and the spatial distances of track pairs is at least 0.3. [4] Thus, empirically we observe that polyonymous track pairs $p_{i,j} \in P_c^*$ are more likely to have lower spatial distance $DisS_{i,j}$. We can utilize this observation to design a more effective initialization strategy for the Beta distributions. For track pairs that are spatially closer, BetaInit reduces the mean of their Beta distributions to increase the probability of them being sampled. For each track pair $p_{i,j}$, its Beta distribution is first set to $Be(1, 1)$, i.e., $S_{i,j} = F_{i,j} = 1$. Then, if its spatial distance $DisS_{i,j}$ is below a set threshold thr_S, BetaInit reduces the mean of the corresponding Beta distribution, i.e., $F_{i,j} = F_{i,j} + 1$. Although some track pairs that are not polyonymous are also initialized with a lower mean Beta distribution, this will be alleviated after a number of sampling iterations. Algorithm 3 presents this procedure. The thresholds thr_S is a hyper-parameter for TMerge and its impact will be evaluated in the experiments.

D. Pruning Track Pairs

We further improve the efficiency of TMerge by an approach, namely ULB, that prunes certain track pairs for further processing if specific conditions are met. After each sampling iteration, we calculate $s'_{i,j}$, the average distance of BBox pairs that have been extracted from each track pair $p_{i,j} \in P_c$ through Equation (8), and $\tilde{s}'_{i,j}$, the normalized average distance for $p_{i,j}$. Let $n_{i,j}$ denote the number of times that a track pair $p_{i,j}$ has been sampled until the current iteration. After the τ-th iteration, for each track pair $p_{i,j} \in P_c$, according to the Hoeffding inequality [22], [27], we have

$$Pr\left(\left|\tilde{s}'_{i,j} - \mathbb{E}[\tilde{s}'_{i,j}]\right| \leq U_{i,j}\right) \geq 1 - \exp(-2U_{i,j}^2 n_{i,j}), \quad (10)$$

[4]We empirically evaluated other sources of information such as the temporal distance $DisT_{i,j}$ which is the time difference (in frames) between the frame in which t_i's last BBox appears and the frame in which t_j's first BBox appears. It turns out that the temporal distance is not significantly correlated to the track pair score (Pearson coefficient < 0.1) and is thus not considered in BetaInit.

where $\mathbb{E}[\tilde{s}'_{i,j}] = \tilde{s}_{i,j}$ and $U_{i,j}$ can be any positive real number. Let $U_{i,j} = \sqrt{\frac{2 \log \tau}{n_{i,j}}}$, Inequality (10) becomes

$$Pr\left(\left|\tilde{s}'_{i,j} - \tilde{s}_{i,j}\right| \leq U_{i,j}\right) \geq 1 - \frac{2}{\tau^4}.$$

This indicates that the probability of the real track pair score $\tilde{s}_{i,j}$ falling outside of $[\tilde{s}'_{i,j} - U_{i,j}, \tilde{s}'_{i,j} + U_{i,j}]$ is lower than $\frac{2}{\tau^4}$, which becomes very small after a few iterations. Thus, for each track pair $p_{i,j} \in P_c$, after the τ-th iteration, the upper and lower bounds of its normalized score are $\tilde{s}'_{i,j} + U_{i,j}$ and $\tilde{s}'_{i,j} - U_{i,j}$ respectively. For a chosen τ that makes the probability bound of the inequality large enough, for each track pair $p_{i,j} \in P_c$, if the upper bound of its score is larger than at most $K{-}1$ lower bounds of scores of other track pairs, i.e.,

$$\left|\left\{\forall p_{i',j'} \in P_c : \tilde{s}'_{i,j} + U_{i,j} > \tilde{s}'_{i',j'} - U_{i',j'}\right\}\right| \leq K - 1,$$

then we are confident that it is in $\hat{P}^*_{c|K}$ and can be pruned from further processing in subsequent iterations, i.e., this track pair will no longer be sampled. Similarly, if the lower bound of its score is larger than at least $\lceil K{\cdot}|P_c| \rceil$ upper bounds of other track pairs' scores, i.e.,

$$\left|\left\{\forall p_{i',j'} \in P_c : \tilde{s}'_{i,j} - U_{i,j} > \tilde{s}'_{i',j'} + U_{i',j'}\right\}\right| \geq \lceil K{\cdot}|P_c| \rceil,$$

we are confident that it is not in $\hat{P}^*_{c|K}$ and can be pruned from subsequent iterations. Algorithm 4 presents the procedure of ULB for calculating the upper and lower bounds for every track pair score and determining the track pairs that are pruned.

E. Efficiency Analysis

TMerge seeks to bias the sampling towards track pairs with smaller scores. To evaluate its efficiency in achieving this goal, we define the *regret* R, which quantifies the deviation of the normalized distances of BBox pairs computed at each iteration from $\tilde{s}_{\text{min}}$, where $\tilde{s}_{\text{min}}$ denotes the normalized score of the track pair with the smallest score, i.e., $\tilde{s}_{\text{min}} = \min\{\tilde{s}_{i,j} \mid \forall p_{i,j} \in P_c\}$. Let $R(\tau_{\text{max}})$ denote the average regret produced by conducting τ_{max} iterations,

$$R(\tau_{\text{max}}) = \frac{1}{\tau_{\text{max}}} \sum_{\tau=1}^{\tau_{\text{max}}} \left(\tilde{d}_\tau - \tilde{s}_{\text{min}}\right),$$

where $\tilde{d}_\tau$ is the normalized distances computed at the τ-th iteration (Line 8 of Algorithm 2). Utilizing the regret guarantee for general stochastic bandits [28] (with no prior), we can derive a bound on the expectation of the average regret,

$$\mathbb{E}\left[\sum_{\tau=1}^{\tau_{\max}} \tilde{d}_\tau - \tau_{\max}\tilde{s}_{\min}\right] \leq O\left(\sqrt{|P_c|\tau_{\max}\log\tau_{\max}}\right)$$
$$\mathbb{E}[R(\tau_{\max})] \leq O\left(\sqrt{\frac{|P_c|\log\tau_{\max}}{\tau_{\max}}}\right). \tag{11}$$

This demonstrates that as $\tau_{\max}$ increases, the average regret of TMerge decreases, and the sampling is biased towards track pairs with lower scores.

For the baseline algorithm, as is evident in Figure 4, its complexity is proportional to the number of track pairs. In contrast, TMerge, given the hyper-parameter $\tau_{\max}$, performs a fixed amount of work. TMerge can be three orders of magnitude faster than the baseline, while achieving a comparable REC level, which will be detailed in our experiments.

F. GPU Acceleration

The algorithm as presented lends itself to native acceleration by a GPU. Instead of processing each track pair at a time, utilizing a GPU to assess the distance of two chosen BBoxes, a *batch* of track pairs can be processed jointly for the same purpose. Typically the size of the batch is constrained by the size of the GPU memory. That way the process can proceed very efficiently. We refer to this version of the algorithm as TMerge-**B** (**B**atch GPU Acceleration).

V. Experiments

A. Datasets

Numerous datasets were used for our evaluation. We report the results on the following three which are highly representative of all of our experiments.

1. Mot-17 [21] is a well-known dataset for multiple object tracking. It contains 7 different indoor and outdoor scenes of public places and 14 videos with pedestrians as the objects of interest. For the videos in the training dataset, we make use of its annotations of tracking to determine the ground truth polyonymous tracks; for the test dataset without annotations, we manually labeled the ground truth.
2. KITTI [29] is a real-world object tracking benchmark. We attempt to identify the polyonymous tracks of pedestrians created by the tracking algorithm and select 8 videos with enough instances of pedestrians from the test dataset[5] and manually label the ground truth.
3. PathTrack [25] is a large multiple object tracking dataset, which features thousands of person trajectories in hundreds of sequences. The sequences are trimmed from the source videos that are crawled from YouTube. We select 9 of the source YouTube videos, which are about two minutes in length on average, and manually labeled the ground truth.

[5]We observe that none of the videos in the training set has enough pedestrians to facilitate the evaluation of our proposal.

Using [30], by comparing the GT tracks to the tracks generated by the tracking algorithm, we can locate all polyonymous tracks. We evaluated different tracking algorithms, namely the algorithms SORT [3], DeepSORT [4], Tracktor [5], UMA [31] and CenterTrack [32] respectively on the three datasets. Overall, Tracktor has the best performance. In particular, it scored the best in terms of evaluation metrics (such as IDF1 [33] and MOTA [30]) and produced the least number of polyonymous tracks. Therefore, without specification, we use the tracking results produced by Tracktor in our experiments. Note that the tracking results produced by Tracktor serve as the input of our algorithm TMerge. The objective of TMerge is to identify the polyonymous tracks inside the tracking results produced by Tracktor. We will compare these tracking algorithms and explore the performance gains of TMerge on other tracking algorithms in §V-G.

We adopt diverse windowing strategies in our evaluation. For each video in MOT-17 and KITTI, we treat the entire video as a window, and apply Tracktor onto the entire video to produce tracking results. Then, we create P_c as the set of all the track pairs in the entire video, and identify the ground truth polyonymous tracks P_c^* produced by Tracktor in each video using [30]. For each video in PathTrack, we partition the entire video into overlapping windows following the procedure described in §II. The default window size is $L = 2000$; we will evaluate the impact of varying Ls in our experiments. Then, for each window W_c of each video in PathTrack, we create P_c and P_c^*. According to §III, we set $K = 5\%$.

B. Algorithms Compared

We present a comparison of the following algorithms:

1. BL: The baseline approach presented in Algorithm 1. This algorithm also can be accelerated utilizing GPUs in the same way as outlined in §IV-F. We refer to the accelerated version of this algorithm as BL-B.
2. PS: This algorithm conducts uniform (stratified) random sampling [34] on each track pair (i.e., a stratum) assessing distances over a fixed proportion (namely η) of BBox pairs. A small η will include fewer BBox pairs, resulting in reduced time overhead but lower *REC*; whereas on the contrary, a large η leads to higher *REC* but increased runtime overheads. PS can be accelerated by GPUs, giving rise to PS-B.
3. LCB: We adapt the Upper Confidence Bounds (UCB) [22] algorithm, an alternative sampling strategy for the Multi-Armed Bandits problem, to identify polyonymous track pairs by replacing UCB with Lower Confidence Bounds (LCB). Unlike TMerge, in each iteration, it calculates the LCB of the scores of all track pairs, selects the track pair with the smallest bound, randomly assesses a BBox pair distance from the selected track pair, and finally updates the lower confidence bounds of all track pairs. We refer to it as LCB and its GPU accelerated version as LCB-B. Since each iteration is dependent on the result of the previous iteration, LCB-B cannot benefit much from GPU acceleration.

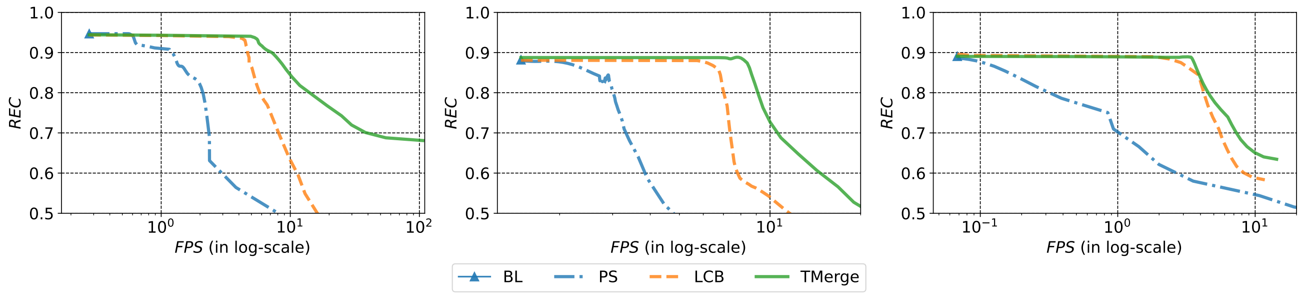

Fig. 5. *REC-FPS* curves on datasets MOT-17 (left), KITTI (middle) and PathTrack (right).

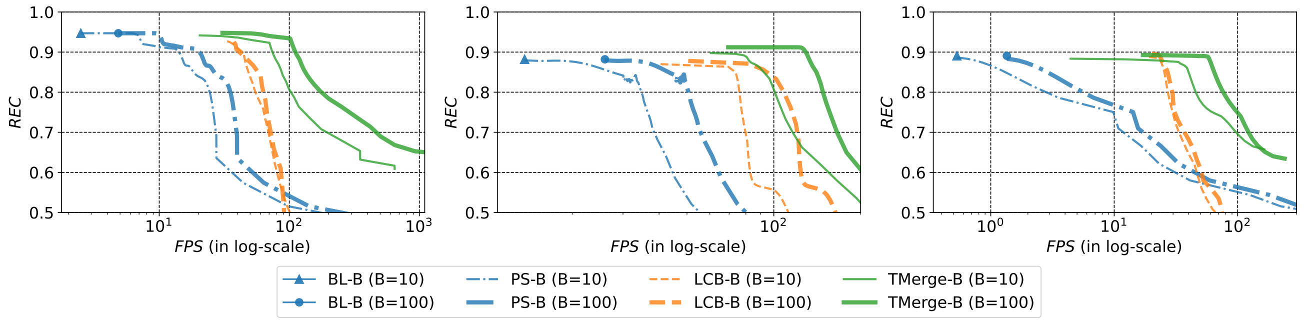

Fig. 6. *REC-FPS* curves of batched algorithms (with different batch sizes $\mathcal{B}$) on datasets MOT-17 (left), KITTI (middle) and PathTrack (right).

4. TMerge and its accelerated version TMerge-B. We set the default value of $\tau_{\max}$ to 10,000 and default $thr_S = 200$. $\tau_{\max}$ determines how many BBox pairs extracted and will be varied in the experiments. We will also evaluate the performance varying thr_S.

All algorithms utilise a ReID model to calculate the BBox pair distance. We utilize the state-of-the-art ReID model, OSNet [20] in our evaluation. [6] For algorithms that are accelerated by GPUs, we denote $\mathcal{B}$ the batch size, namely the number of track pairs that are jointly evaluated. Each experiment reports the average of the results of 10 independent trials for each algorithm compared. All algorithms were implemented in Python and run on a Linux server with Intel Xeon Gold 6244 3.60GHz CPU, with 64GB memory having an NVIDIA TITAN Xp GPU.

C. Metrics

We use *REC* to measure the end-to-end accuracy of the algorithms, where $REC = REC(\hat{P}^*_{c|K})$ (defined in §II Equation (3)). We use *Runtime* and *FPS* to evaluate the performance of the algorithms, where Runtime is in seconds and *FPS* expresses the average number of frames processed per second by the algorithms.

D. Track Merging Performance

By varying η in algorithm PS and varying $\tau_{\max}$ in algorithms LCB and TMerge, we plot Figure 5, the *REC-FPS* curves on the algorithms compared. Points closer to the top right corner represent better performance. From Figure 5, it can be observed that, at the same *REC*, TMerge provides 10x to 100x higher *FPS* than PS and BL. We choose two *REC* values and present the *FPS* of the algorithms under the *REC* values on MOT-17 in Table II. At *REC*=0.80, the speed of TMerge is 16 times faster than that of PS; at *REC*=0.93, the speed of TMerge is 19 times faster than that of PS and 23 times than BL. From Figure 5, we observe that the performance improvement of TMerge

[6]We retrain the ReID model by a loss function that combines triplet and softmax loss [35] to enable it to extract feature vectors from BBoxes, where the BBoxes with smaller Euclidean distance between their feature vectors are more likely to refer to the same object.

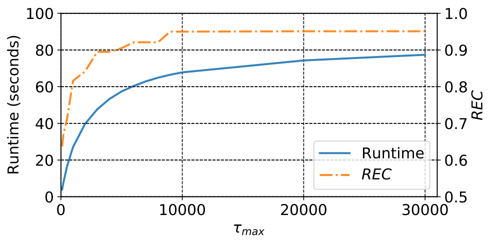

Fig. 7. Runtime and *REC* of TMerge-B varying $\tau_{\max}$.

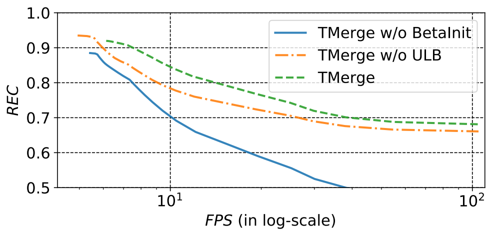

Fig. 8. *REC-FPS* curves of TMerge and TMerge without BetaInit or ULB.

TABLE II

THE *FPS* OF METHODS BL-B, PS-B, LCB-B AND TMERGE-B ON MOT-17 UNDER *REC*=0.80 AND *REC*=0.93.

Methods	*REC*=0.80	*REC*=0.93
BL	-	0.27
PS	1.16	0.32
LCB	5.68	4.45
TMerge	19.26	6.10

Methods	$\mathcal{B}$=10		$\mathcal{B}$=100	
	REC=0.80	*REC*=0.93	*REC*=0.80	*REC*=0.93
BL-B	-	2.50	-	4.85
PS-B	13.47	2.98	19.42	5.58
LCB-B	53.74	34.79	52.74	39.20
TMerge-B	108.84	70.40	254.77	106.20

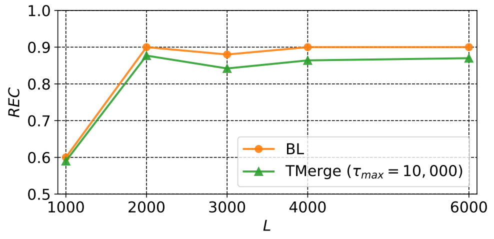

Fig. 9. *REC* of TMerge and BL varying *L*.

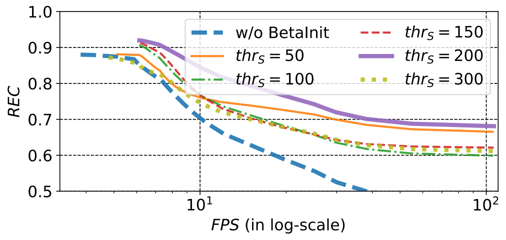

Fig. 10. *REC-FPS* curves of TMerge varying thr_S.

over LCB, which is also an efficient sampling approach, is less pronounced. However, LCB cannot benefit much from GPU acceleration, as demonstrated in Figure 6.

Figure 6 presents the *REC-FPS* curves on all algorithms amenable to acceleration varying the batch size $\mathcal{B}$. We also present in Table II the *FPS* of the algorithms with GPU acceleration under the *REC* values 0.80 and 0.93 on MOT-17. It is evident that TMerge-B is still 10x to 100x faster than PS-B and BL-B. Besides, it can be observed that GPU acceleration significantly improves TMerge-B, and at the same *REC* value, the *FPS* with $\mathcal{B}$=100 (denoted as TMerge-B$_{100}$) is significantly faster than that with $\mathcal{B}$=10 (denoted as TMerge-B$_{10}$), as expected. For example, at *REC*=0.80, TMerge-B$_{100}$ is 2.3 times faster than TMerge-B$_{10}$ and 13.2 times faster than TMerge; at *REC*=0.93, TMerge-B$_{100}$ is 1.5 times faster than TMerge-B$_{10}$ and 17.4 times faster than TMerge. For LCB-B, however, since each iteration is dependent on the result of the previous iteration, increasing $\mathcal{B}$ has little benefit for LCB-B. For example, at *REC*=0.80, the *FPS* of TMerge-B$_{100}$ is 4.8 times the *FPS* of LCB-B with $\mathcal{B}$=100; at *REC*=0.93, the *FPS* of TMerge-B$_{100}$ is 2.7 times the *FPS* of LCB-B with $\mathcal{B}$=100.

Figure 7 presents Runtime (in seconds) and *REC* of TMerge-B ($\mathcal{B} = 10$) as the number of iterations increases on all the videos in dataset MOT-17. For a small number of iterations, both *REC* and Runtime increase fairly fast. After that, improvement of *REC* exhibits diminishing returns, approaching the *REC* achieved by the Baseline algorithm. The reason is that, the bulk of the polyonymous track pairs have lower scores than the rest and the algorithm is highly successful in identifying them. Harder pairs, which scores not as low, require more iterations to identify. The Runtime exhibits a similar behaviour. As $\tau_{\max}$ grows, more feature vectors can be reused thus requiring less inferences using the ReID model. As a result, Runtime grows slowly in later iterations. As a comparison, algorithm BL-B takes 2,762 seconds to process all the videos in dataset MOT-17, which is 2 orders of magnitude slower than TMerge-B.

E. Ablation Study

To investigate how each of TMerge's components (BetaInit and ULB) contributes to its improved performance, we conduct an ablation study in which we start with TMerge and remove each component individually. We show the *REC-FPS* curves on MOT-17 dataset in Figure 8; similar trends were observed on other datasets. The blue curve (for TMerge without BetaInit) is at the lower left of the other two, which indicates that BetaInit has great impact on the efficiency of the algorithm. For example, under *REC*=0.89, BetaInit can improve TMerge's speed about 1.7 times. The orange curve (for TMerge without ULB) is also at lower left to the green curve (for TMerge), suggesting that ULB also contributes to the efficiency of the algorithm. Overall, BetaInit appears to have greater impact on the performance of the proposed approach.

F. Sensitivity Analysis

We investigated whether TMerge is sensitive to the hyperparameters by varying the BetaInit threshold thr_S and window length L. The study of parameter K is addressed in §III, and since its effect on TMerge follows the pattern depicted in Figure 3, it will not be elaborated further in this section.

(1) Influence of L. Figure 9 presents the *REC* achieved by algorithms BL and TMerge for varying values of the hyper-parameter window size L on PathTrack. According to the annotation made by the authors, for this dataset, the maximum length of tracks is around 1000 frames, i.e., $L_{\max}$=1000. Therefore, for L=1000 $< 2L_{\max}$, some polyonymous tracks may span more than two windows, making them hard to discover, impacting overall accuracy of both TMerge and BL. For other values of Ls such that $\geq 2L_{\max}$, the values of *REC* achieved by both BL and TMerge are close, which indicates that the algorithms are not sensitive to L.

(2) Influence of thr_S. Figure 10 presents the *REC-FPS* curves of TMerge on MOT-17, varying the hyper-parameter threshold thr_S for BetaInit (Algorithm 3). First, it can

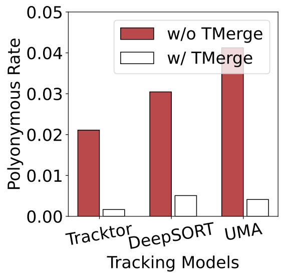

Fig. 11. Polyonymous Rates of Tracktor, DeepSORT and UMA on MOT-17 with and without TMerge. The lower the better.

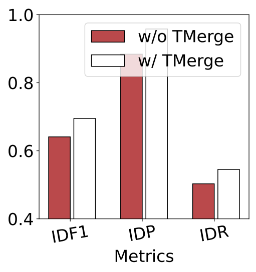

Fig. 12. MOT Metrics IDF1, IDP, and IDR of Tracktor on dataset MOT-17 with and without TMerge. The higher the better.

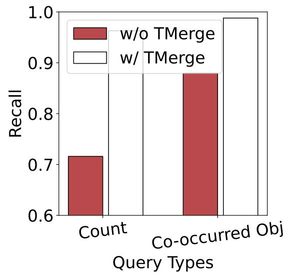

Fig. 13. Recall of the results on queries *Count* and *Co-occurred Objects* with and without TMerge on MOT-17. The higher the better.

be observed that the blue curve is lower than the others, indicating that TMerge without BetaInit performs worse than all others with BetaInit. Besides, BetaInit is sensitive to the values of thr_S. For example, in order to achieve the same *REC*=0.89, the *FPS* of TMerge with thr_S=200 is approximately 40% faster than that of thr_S=300. The optimal values of both L and thr_S can be both obtained by grid search on a period of labelled frame sequences.

G. Performance Gains on Different Tracking Algorithms

In this section, we compare three state-of-the-art tracking algorithms and explore the performance gains made by TMerge on them. Since the number of tracks found by each tracking algorithm varies, in order to depict the effect of the polyonymous track pairs on the tracking results, we define *Polyonymous Rate*, which is the ratio of the number of polyonymous track pairs to all track pairs. For the tracking algorithms without TMerge,

$$\text{Polyonymous Rate} = \frac{|P_c^*|}{|P_c|},$$

where P_c^* is the set of track pairs in P_c that are polyonymous (Equation (2)); for the tracking results corrected by TMerge,

$$\text{Polyonymous Rate}_{|\text{TMerge}} = \frac{|P_c^* \setminus \hat{P}_{c|K}^*|}{|P_c|},$$

where $\hat{P}_{c|K}^*$ is the set of track pairs estimated by Algorithm 2. Figure 11 demonstrates the Polyonymous Rates of Tracktor, DeepSORT and UMA on MOT-17 with and without TMerge; other datasets have similar behaviors. It can be observed that TMerge reduces the Polyonymous Rates of the tracking results by more than 10x. While existing tracking algorithms, such as [4], may alleviate track fragmentation caused by occlusion, it is evident that they are unable to eliminate polyonymous tracks without TMerge.

To directly evaluate the impact of the deployment of TMerge on the performance of tracking results, we plot Figure 12, which illustrates the performance gains of TMerge on Tracktor over multi-object tracking evaluation metrics [33]. The figure indicates that IDF1 [33], which evaluates the overall performance of the tracking results, has improved by 5 percentage points. In particular, both IDP and IDR have been improved. This indicates that TMerge corrects the polyonymous tracks generated by Tracktor that cannot be matched to any GT tracks, converting them from false negatives to true positives.

H. Benefits to Query Processing

This section discusses use cases that TMerge directly influences, namely processing queries that utilize tracking information over video feeds. TMerge is intended as a data pre-processing step after tracking algorithms have been applied but before downstream query processing takes place. We apply the framework for query processing proposed in [13], utilizing Mask R-CNN [36] and Tracktor for video pre-processing, and consider the following two types of video queries:

- *Count*: Count the number of objects (i.e., individual tracks) across more than a certain number (e.g., 200) of frames; Such queries are used to provide statistical information of objects in a certain time period for a given video; alternatively they are utilized to retrieve scenes that are related to the number of objects that remain visible in the scene for a certain time period (e.g., to find traffic congestion video clips from a long traffic video, or to identify cars/persons that are visible longer than a certain time period in a given video) [13].
- *Co-occurring Objects*: Identify video clips that are longer than a number of frames (e.g., 50 frames) and contain the same three objects appearing jointly. Such queries are used to retrieve video clips with certain co-occurrence patterns, possibly combined with other query conditions, e.g., to identify video clips where the same two persons and one vehicle with a coca-cola brand advertisement on the vehicle appear jointly [13].

We compare the results on the two queries with and without TMerge on MOT-17 and compute the Recall, demonstrated in Figure 13; other datasets have similar behaviors. It can be observed that, for the query *Count*, due to the polyonymous tracks, some tracks that meet the query condition are not detected, resulting in a low Recall (<75%). TMerge merges these polyonymous tracks, thereby enhancing Recall to more than 95%. Similarly, for the query *Co-occurring Objects*, TMerge increases the Recall from 88% to 95%. The Recall of such queries relies heavily on the accuracy of tracking results, which can be improved by reducing the number of polyonymous tracks. These are just two query scenarios where

TMerge can be applied; TMerge will benefit any query that relies on tracking results and thus have wide applicability.

VI. Related Work

Automated video analytics utilizing deep learning models as primitives is an area of increasing research interest in the community [8]–[11], [37]–[40]. Numerous recent works present query processing frameworks that include both frame content (object types, positions in frames, etc.) and temporal constraints (object tracking outputs, etc.) as query primitives. Kang et al. [6], [7] present video query processing techniques for specific types of aggregates. Chen et al. [12], [13] present a framework for processing temporal queries involving objects and their co-occurrences. Bastani et al. [41] utilize object tracking to answer count and spatial-constrained queries. Bastani and Madden [42] develop a method to obtain and apply tracking results more efficiently. A prerequisite for implementing these frameworks accurately is that deep learning algorithms (i.e., object tracking models) can correctly extract metadata that uniquely identify and track objects across video frames. In this work, we propose a general approach to improving the accuracy of tracking results by merging polyonymous tracks, leveraging many of the downstream query processing techniques used in previous work.

Multiple object tracking is an active research area featuring several proposals [31], [32], [43]. Occlusions (either object-to-object or object-to-scene occlusions) remain a significant challenge for object tracking [14]. Luo et al. [15] summarize various typical strategies to address occlusions. For example, the part-to-whole strategy, which is the most commonly used strategy in various scenarios (and which Tracktor [5] employs), assumes that a part of the object is still visible when an occlusion happens. Besides, some tracking methods handle occlusions by modeling the object's appearance and motion [44]. SORT [3] uses the location and size of bounding boxes to detect occlusions. Deep SORT [4] utilizes appearance information integrated based on a deep appearance descriptor to re-identify occluded targets. Some other approaches, such as [45] and [46], take motion prediction into account for tracking, deriving models that will be used to continuously predict the object location when a tracked object is occluded until the object reappears. However, the evaluation of the tracking results of the state-of-the-art tracking methods presented in [14], [15] (especially evaluations on metrics IDS and Frag) demonstrates that the occlusion handling tactics offered by these methods in the CV community do not fully eliminate polyonymous tracks.

Re-identification (ReID), aiming to determine whether a specific person or object exists in a target repository of data captured at different times or by other cameras, has been an active research area in computer vision in recent years [47], [48]. Research on ReID usually focuses either on constructing robust and discriminative features or identifying an improved similarity metric for comparing features [18], [49]. Some research (e.g., [17], [19]) applies a training framework with triplet loss functions to the person re-identification problems, utilizing deep convolution neural network (DCNN) models. In addition to person ReID, vehicle re-identification (V-ReID) has also become significantly popular due to its practical significance [50]. After the development of the early sensor based methods and hand feature based methods, certain V-ReID models based on DCNN have achieved great results [51], [52].

VII. Conclusions

Object tracking algorithms facilitate video query frameworks involving temporal-spatial constraints; however errors incurred by the tracking algorithm, such as track fragmentation, diminish their utility for video query processing.

In this paper, we propose the TMerge algorithm that utilizes sampling, to effectively identify polyonymous tracks. The algorithm constitutes a pre-processing step that can be adapted by video analytics systems to yield more accurate metadata extraction. Extensive experiments conducted on real datasets demonstrate that TMerge can provide a upto 100x speedup over other applicable approaches and is able to identify 95% of all polyonymous tracks by examining only a small number of track pairs. We believe that the area of quality driven data ingestion on the output of deep learning models is an important research direction.

References

[1] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[3] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proceedings of the IEEE International Conference on Image Processing*, 2016, pp. 3464–3468.

[4] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proceedings of the IEEE International Conference on Image Processing*, 2017, pp. 3645–3649.

[5] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking without bells and whistles," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 941–951.

[6] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "Noscope: Optimizing deep cnn-based queries over video streams at scale." *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1586–1597, 2017.

[7] D. Kang, P. Bailis, and M. Zaharia, "Blazeit: Optimizing declarative aggregation and limit queries for neural network-based video analytics," *Proceedings of the VLDB Endowment*, vol. 13, no. 4, pp. 533–546, 2019.

[8] I. Xarchakos and N. Koudas, "Svq: Streaming video queries," in *Proceedings of the International Conference on Management of Data*, 2019, pp. 2013–2016.

[9] N. Koudas, R. Li, and I. Xarchakos, "Video monitoring queries," in *IEEE International Conference on Data Engineering*, 2020, pp. 1285–1296.

[10] D. Chao, N. Koudas, and I. Xarchakos, "Svq++: Querying for object interactions in video streams," in *Proceedings of the International Conference on Management of Data*, 2020, pp. 2769–2772.

[11] Y. Xarchakos and N. Koudas, "Querying for interactions," in *IEEE International Conference on Data Engineering*, 2021, pp. 2153–2158.

[12] Y. Chen, X. Yu, and N. Koudas, "Tqvs: Temporal queries over video streams in action," in *Proceedings of the International Conference on Management of Data*, 2020, pp. 2737–2740.

[13] Y. Chen, X. Yu, N. Koudas, and Z. Yu, "Evaluating temporal queries over video feeds," in *Proceedings of the International Conference on Management of Data*, 2021, pp. 287–299.

[14] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera, "Deep learning in video multi-object tracking: A survey," *Neurocomputing*, vol. 381, pp. 61–88, 2020.

[15] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim, "Multiple object tracking: A literature review," *Artificial Intelligence*, vol. 293, p. 103448, 2021.

[16] C. Zhao, P. Mei, S. Xu, Y. Li, and Y. Feng, "Performance evaluation of visual object detection and tracking algorithms used in remote photoplethysmography," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[17] S. Ding, L. Lin, G. Wang, and H. Chao, "Deep feature learning with relative distance comparison for person re-identification," *Pattern Recognition*, vol. 48, no. 10, pp. 2993–3003, 2015.

[18] Z. Zhong, L. Zheng, D. Cao, and S. Li, "Re-ranking person re-identification with k-reciprocal encoding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1318–1327.

[19] M. Ye, J. Shen, G. Lin, T. Xiang, L. Shao, and S. C. Hoi, "Deep learning for person re-identification: A survey and outlook," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 6, pp. 2872–2893, 2021.

[20] K. Zhou, Y. Yang, A. Cavallaro, and T. Xiang, "Omni-scale feature learning for person re-identification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3702–3712.

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

[22] A. Slivkins *et al.*, "Introduction to multi-armed bandits," *Foundations and Trends® in Machine Learning*, vol. 12, no. 1-2, pp. 1–286, 2019.

[23] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," in *Annual Conference on Neural Information Processing Systems*, 2011, pp. 2249–2257.

[24] M. Zamprogno, M. Passon, N. Martinel, G. Serra, G. Lancioni, C. Micheloni, C. Tasso, and G. L. Foresti, "Video-based convolutional attention for person re-identification," in *International Conference on Image Analysis and Processing*, 2019, pp. 3–14.

[25] S. Manen, M. Gygli, D. Dai, and L. Van Gool, "Pathtrack: Fast trajectory annotation with path supervision," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 290–299.

[26] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise Reduction in Speech Processing*, 2009, pp. 1–4.

[27] W. Hoeffding, "Probability inequalities for sums of bounded random variables," in *The Collected Works of Wassily Hoeffding*, 1994, pp. 409–426.

[28] S. Agrawal and N. Goyal, "Analysis of thompson sampling for the multi-armed bandit problem," in *Conference on Learning Theory*, 2012, pp. 39–1.

[29] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[30] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the clear mot metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, pp. 1–10, 2008.

[31] J. Yin, W. Wang, Q. Meng, R. Yang, and J. Shen, "A unified object motion and affinity model for online multi-object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6768–6777.

[32] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," in *European Conference on Computer Vision*, 2020, pp. 474–490.

[33] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European Conference on Computer Vision*, 2016, pp. 17–35.

[34] W. Madow, "Elementary sampling theory," *Technometrics*, vol. 10, pp. 621–622, 08 1968.

[35] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[36] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969.

[37] D. Kang, J. Guibas, P. D. Bailis, T. Hashimoto, and M. Zaharia, "Tasti: Semantic indexes for machine learning-based queries over unstructured data," in *Proceedings of the International Conference on Management of Data*, 2022, pp. 1934–1947.

[38] P. Chunduri, J. Bang, Y. Lu, and J. Arulraj, "Zeus: Efficiently localizing actions in videos using reinforcement learning," in *Proceedings of the International Conference on Management of Data*, 2022, pp. 545–558.

[39] J. Cao, K. Sarkar, R. Hadidi, J. Arulraj, and H. Kim, "Figo: Fine-grained query optimization in video analytics," in *Proceedings of the International Conference on Management of Data*, 2022, pp. 559–572.

[40] D. Chao, N. Koudas, and X. Yu, "Marshalling model inference in video streams," in *2023 IEEE 39th International Conference on Data Engineering*, 2023.

[41] F. Bastani, S. He, A. Balasingam, K. Gopalakrishnan, M. Alizadeh, H. Balakrishnan, M. Cafarella, T. Kraska, and S. Madden, "Miris: Fast object track queries in video," in *Proceedings of the International Conference on Management of Data*, 2020, pp. 1907–1921.

[42] F. Bastani and S. Madden, "OTIF: efficient tracker pre-processing over large video datasets," in *Proceedings of the International Conference on Management of Data*, 2022, pp. 2091–2104.

[43] Y. Yuan, J. Chu, L. Leng, J. Miao, and B.-G. Kim, "A scale-adaptive object-tracking algorithm with occlusion detection," *EURASIP Journal on Image and Video Processing*, vol. 2020, no. 1, pp. 1–15, 2020.

[44] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys*, vol. 38, no. 4, pp. 13–es, 2006.

[45] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 300–311.

[46] Y. Liu, R. Li, Y. Cheng, R. T. Tan, and X. Sui, "Object tracking using spatio-temporal networks for future prediction location," in *European Conference on Computer Vision*, 2020, pp. 1–17.

[47] S. D. Khan and H. Ullah, "A survey of advances in vision-based vehicle re-identification," *Computer Vision and Image Understanding*, vol. 182, pp. 50–63, 2019.

[48] M. Liu, J. Zhao, Y. Zhou, H. Zhu, R. Yao, and Y. Chen, "Survey for person re-identification based on coarse-to-fine feature learning," *Multimedia Tools and Applications*, pp. 1–35, 2022.

[49] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng, "Person re-identification by multi-channel parts-based cnn with improved triplet loss function," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1335–1344.

[50] S. D. Khan and H. Ullah, "A survey of advances in vision-based vehicle re-identification," *Computer Vision and Image Understanding*, vol. 182, pp. 50–63, 2019.

[51] Y. Zhang, D. Liu, and Z.-J. Zha, "Improving triplet-wise training of convolutional neural network for vehicle re-identification," in *IEEE International Conference on Multimedia and Expo*, 2017, pp. 1386–1391.

[52] X. Liu, W. Liu, T. Mei, and H. Ma, "Provid: Progressive and multi-modal vehicle reidentification for large-scale urban surveillance," *IEEE Transactions on Multimedia*, vol. 20, no. 3, pp. 645–658, 2017.