Published: 26 March 2024

RESEARCH-ARTICLE

Predictive and Near-Optimal Sampling for View Materialization in Video Databases

YANCHAO XU, Zhejiang University, Hangzhou, Zhejiang, China

DONGXIANG ZHANG, Zhejiang University, Hangzhou, Zhejiang, China

SHUHAO ZHANG, Nanyang Technological University, Singapore City, Singapore

SAI WU, Zhejiang University, Hangzhou, Zhejiang, China

ZEXU FENG, Zhejiang University, Hangzhou, Zhejiang, China

GANG CHEN, Zhejiang University, Hangzhou, Zhejiang, China

Predictive and Near-Optimal Sampling for View Materialization in Video Databases

YANCHAO XU, Zhejiang University, China
DONGXIANG ZHANG*, Zhejiang University, China
SHUHAO ZHANG, Nanyang Technological University, Singapore
SAI WU, Zhejiang University, China
ZEXU FENG, Zhejiang University, China
GANG CHEN, Zhejiang University, China

Scalable video query optimization has re-emerged as an attractive research topic in recent years. The OTIF system, a video database with cutting-edge efficiency, has introduced a new paradigm of utilizing view materialization to facilitate online query processing. Specifically, it stores the results of multi-object tracking queries to answer common video queries with sub-second latency. However, the cost associated with view materialization in OTIF is prohibitively high for supporting large-scale video streams.

In this paper, we study efficient MOT-based view materialization in video databases. We first conduct a theoretical analysis and establish two types of optimality measures that serve as lower bounds for video frame sampling. In order to minimize the number of processed video frames, we propose a novel predictive sampling framework, namely LEAP, exhibits near-optimal sampling performance. Its efficacy relies on a data-driven motion manager that enables accurate trajectory prediction, a compact object detection model via knowledge distillation, and a robust cross-frame associator to connect moving objects in two frames with a large time gap.

Extensive experiments are conducted in 7 real datasets, with 7 baselines and a comprehensive query set, including selection, aggregation and top-k queries. The results show that with comparable query accuracy to OTIF, our LEAP can reduce the number of processed video frames by up to 9× and achieve 5× speedup in query processing time. Moreover, LEAP demonstrates impressive throughput when handling large-scale video streams, as it leverages a single NVIDIA RTX 3090ti GPU to support real-time MOT-based view materialization from 160 video streams simultaneously.

CCS Concepts: • **Information systems → Query optimization**.

Additional Key Words and Phrases: Video Analytics, Database Management System

ACM Reference Format:
Yanchao Xu, Dongxiang Zhang, Shuhao Zhang, Sai Wu, Zexu Feng, and Gang Chen. 2024. Predictive and Near-Optimal Sampling for View Materialization in Video Databases. *Proc. ACM Manag. Data* 2, 1 (SIGMOD), Article 19 (February 2024), 27 pages. https://doi.org/10.1145/3639274

*Dongxiang Zhang is the corresponding author.

Authors' addresses: Yanchao Xu, The State Key Laboratory of Blockchain and Data Security, Zhejiang University, Hangzhou, China, xuyanchao@zju.edu.cn; Dongxiang Zhang, Zhejiang University, Hangzhou, China, zhangdongxiang@zju.edu.cn; Shuhao Zhang, Nanyang Technological University, Singapore, Singapore, shuhao.zhang@ntu.edu.sg; Sai Wu, Zhejiang University, Hangzhou, China, wusai@zju.edu.cn; Zexu Feng, Zhejiang University, Hangzhou, China, fengzexu@zju.edu.cn; Gang Chen, Zhejiang University, Hangzhou, China, cg@zju.edu.cn.

ACM 2836-6573/2024/2-ART19
https://doi.org/10.1145/3639274

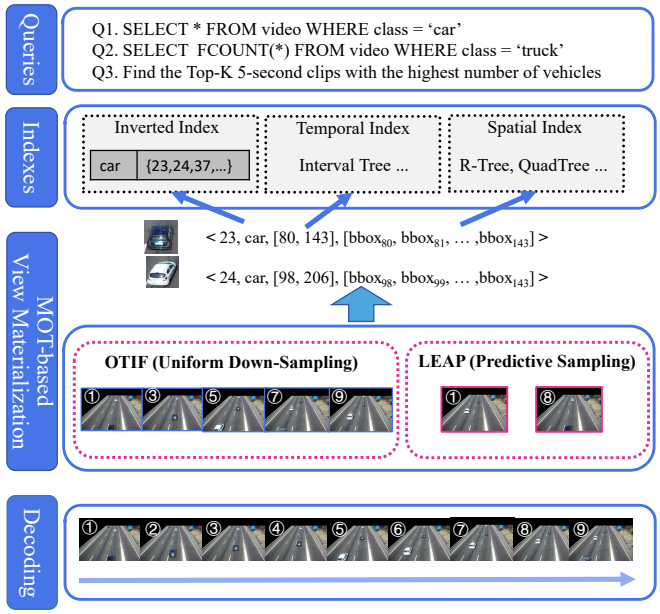


Fig. 1. Pipeline of MOT-based view materialization for video query processing.

1 INTRODUCTION

The proliferation of smart cities has resulted in a remarkable upsurge in the deployment of video cameras, which play a pivotal role in fortifying public safety and optimizing public transportation. These cameras produce continuous flows of live video streams that constitute a massive video database for analytical queries [44] and insight extraction [49, 51] through the utilization of increasingly accurate machine learning models. To address the efficiency issue, scalable video query optimization has re-emerged as an attractive research topic in recent years and a noticeable number of video database systems have been proposed.

Since the primary performance bottleneck originates from the computationally intensive inference overhead incurred by deep learning models, the prevailing approach is to construct fast proxy models to replace the time-consuming oracle model, while accepting a tolerable level of accuracy degradation. The idea has been widely embraced and implemented within systems like NoScope [22], FOCUS [16], TAHOMA [1], ABAE [23], Everest [28] and FiGO [5]. Alternatively, we can leverage the inherent temporal redundancy within successive video frames to reduce the number of processed frames, either by down-sampling [3, 4] or constructing a lightweight binary classifier to skip irrelevant frames [22, 40]. In FOCUS [16] and Video-zilla [17], frame clustering and inverted index are utilized to improve efficiency. The video frames are first ingested with cheap convolutional neural networks to extract class labels. Subsequently, semantically similar frames are clustered and inverted index is built to reduce the search space of video frame retrieval queries.

Among these systems, OTIF [4] has introduced a new paradigm in the realm of video query processing. As shown in Figure 1, it harnesses multi-object tracking (MOT) techniques to extract the trajectories of all moving objects portrayed in video footage. This step can be considered as a manner of view materialization, which stores the results of multi-object tracking queries to facilitate online processing of other queries. The output of MOT can be represented as $\langle oid, labels, interval, bboxes \rangle$, where *interval* refers to the starting and ending frames that contain the object and *bboxes* indicates

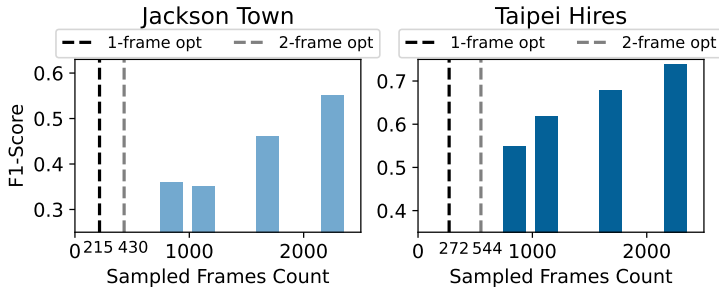


Fig. 2. Performance of OTIF under low sampling rates.

the object size and position at each relevant video frame. With the extracted trajectories, offline indexes (e.g., inverted index, temporal index and spatial index) can be constructed to answer selection, aggregation and top-k queries with sub-second latency. For instance, the retrieval of video frames containing both an ambulance and a firetruck can be effortlessly achieved through a simple intersection operation between the inverted lists corresponding to these two labels. Similarly, estimating traffic flow at a timestamp can be accomplished by leveraging stabbing queries over the interval tree to obtain the number of distinct vehicles.

These features render OTIF as the sole system capable of handling a diverse range of queries and exhibits cutting-edge efficiency. Nonetheless, its view materialization is too expensive to support large-scale cameras. Even though OTIF has adopted down-sampling to reduce the number of processed video frames for multi-object tracking, based on its experimental analysis, applying object detection on the sampled frames is the performance bottleneck and occupies more than 60% of the total execution time. Since the cost of object detection is positively correlated with the number of sampled video frames, a feasible solution to further enhance efficiency is devising a more robust materialization strategy that operates effectively under a reduced sampling rate. Therefore, in this paper, we are motivated to investigate the following two research questions.

RQ1: In the optimal scenario, what is the minimum number of sampled frames to cover all moving objects? Given a video clip, we can extract a collection of moving objects $\{o_1, o_2, \ldots, o_n\}$. Each object o_i is present within a consecutive sequence of frames spanning the interval $[s_i, e_i]$. Here, s_i represents the initial frame in which o_i is detected and frame e_i captures the final occurrence of o_i. To infer the minimum number of sampled frames that cover all moving objects, we formulate it as a minimum interval stabbing problem. We call this 1-frame optimality because each object is only required to appear in one video frame. However, with only one observation of o_i, it becomes exceedingly difficult to extract its trajectory from the video clip. Thus, we delve into an examination of 2-frame optimality and theoretically derive the minimum number of video frames in which each moving object is captured at least twice. These two optimality measures serve as theoretical lower bounds for video frame sampling and provide guidance for an MOT-based view materialization algorithm in assessing its optimization potential when operating at a low sampling rate.

RQ2: In practice, can we perform MOT-based view materialization with near-optimal number of frames?

In the scenario of an extremely low sampling rate, the query processing accuracy of OTIF deteriorates to an impractical level. The reason is that with sparse observations of target objects, it is particularly challenging to accurately capture the motion patterns of moving objects and robustly perform object association. As verified in Figure 2, we conduct selection queries on the target object “bus” using video datasets Jackson and Taipei. The number of frames for 1-frame

optimality and 2-frame optimality have been highlighted. Notably, when the number of sampled frames is significantly reduced, approaching the threshold of 2-frame optimality, the F_1-score of OTIF exhibits a drastic decline to 0.36 and 0.54 in these two datasets, respectively.

In this paper, we devise an MOT-based view materialization strategy called LEAP, which works well when the sampling rate is set in proximity to the 2-frame optimality. LEAP builds upon a predictive framework, wherein we progressively predict the next sampling frame F_{i+1} based on the objects that **first appear** in F_i. We set F_{i+1} as the frame capturing the **last occurrence** of these objects such that the sampling rate can get close to 2-frame optimality.

To predict the next sample frame F_{i+1}, we propose a data-driven motion manager for object trajectory forecasting. The motion management is initialized by trajectory clustering and updated to capture unseen trajectories or new travel patterns. It performs trajectory forecasting by assigning the target object to the cluster with the highest matching probability and uses the representative trajectory in the cluster to determine the location of the object in the subsequent frames. To connect the objects between two sampled frames F_i and F_{i+1}, we devise a robust object association strategy that takes into account spatial, temporal and visual clues simultaneously. Finally, since object detection is the most expensive operator, we employ knowledge distillation to acquire a compact detector network without performance degradation.

We conduct extensive experiments to compare LEAP with 7 baselines in 7 real datasets. The experiments cover a comprehensive query set, including selection, aggregation and top-k queries. The results show that LEAP can achieve near-optimal sampling performance. With comparable query accuracy to OTIF, our LEAP can reduce the number of processed video frames by up to 9× and achieve 5× speedup in terms of query processing efficiency. We also evaluate the throughput of parallel view materialization among large-scale video streams. LEAP demonstrates impressive throughput — it can leverage a single NVIDIA RTX 3090ti GPU to support view materialization from 160 video streams in real time. We release the LEAP source code at https://github.com/zju-xuyc/LEAP.

In summary, we study efficient MOT-based view materialization for video database under an extremely low sampling rate and present the following key contributions:

(1) We establish two types of optimality measures, serving as theoretical lower bounds of video frame sampling for MOT-based view materialization.
(2) To the best of our knowledge, we are the first to adopt a predictive sampling strategy in VDBMS, which works well when the sampling rate is set in proximity to the 2-frame optimality.
(3) We propose a data-driven motion manager that provides accurate trajectory prediction with very sparse observations. In addition, we devise a compact object detection model with higher efficiency and a resilient object association strategy to connect objects across frames with a large temporal gap.
(4) We perform comprehensive experiments to validate the superiority of LEAP over OTIF. LEAP requires 9× fewer video frames and achieves 5× speedup for video query processing with the same accuracy level.

As far as we know, LEAP is the first work aiming to parse videos with such large sampling intervals that far exceed what previous works could support. Larger sampling intervals introduce greater challenges into the entire system, and we have managed to address this through a comprehensive framework, thereby enabling real-time parsing of dozens of video streams simultaneously.

The remainder of this paper is organized as follows. We review the literature of video query optimization systems in Section 2. In Section 3, we undertake a theoretical analysis on the lower bound of video frame sampling and present an overview of our sampling scheme. The construction

and maintenance of a data-driven motion manager for trajectory prediction is proposed in Section 4. Techniques in terms of object detection and tracking are presented in Section 5. We conduct extensive experiments in Section 6 and conclude the paper in Section 7.

2 RELATED WORK

2.1 On-the-fly Video Query Optimization

NoScope [22] is a pioneering work that leverages smaller yet faster specialized networks to enhance model inference speed. Moreover, it takes advantage of redundancy in neighboring frames and trains a difference detector for irrelevant frame filtering. The idea of proxy model design to achieve better tradeoff between efficiency and accuracy has been widely adopted by subsequent works, including FOCUS [16], TAHOMA [1], ABAE [23], Everest [28], Dove [47] and FiGO [5]. For example, TAHOMA replaces the accurate-but-expensive CNN with cascades of fast image classifiers. It constructs numerous specialized candidate binary-classification CNN models and identifies a set of Pareto-optimal cascades with varying trade-offs between accuracy and throughput. Everest [28] trains a lightweight convolutional mixture density network (CMDN) to generate an approximate score distribution for each frame. Subsequently, it utilizes uncertain query processing to accelerate top-K analytics while providing probabilistic guarantees.

An alternative approach to enhance performance speed is the adoption of downsampling techniques to reduce the number of processed frames. MIRIS [3] and OTIF [4] adopt uniform downsampling to efficient object tracking. ExSample [36] extends the sampling strategy from uniform downsampling to importance sampling.

To reduce the manual efforts of tuning configuration parameters of video analytics, such as video resolution, sampling rate and the variants of deep learning models, automatic configuration of the video query processing engine also received significant attention. These works are similar to automatic database tuning in traditional RDBMS, such as CDBTune [53], QTune [29]. Zeus [6] adopts reinforcement learning to indicate the relevant video segments and the sampling rate to control the tradeoff between efficiency and accuracy. SMOL [25] considers input image size as a key factor. With smaller input image size, the image processing time and model inference cost can be reduced.

2.2 Index-Assisted Video Query Optimization

To support video frame selection queries for target objects, FOCUS [16] and Video-zilla [17] first ingest video frames with cheap convolutional neural networks to extract class labels. Subsequently, semantically similar frames are clustered and an inverted index is built to improve efficiency. When supporting aggregation queries, BlazeIt [21] consists of an ingestion step to randomly sample a subset of video frames and annotate them using the expensive CNN models. The images as well as their labels will be used to train a specialized NN to estimate the statistic, whose variance is further reduced via the technique of control variates [2]. In OTIF [4], the goal of the offline ingestion is to extract all the object tracks in an efficient and accurate fashion. Two optimization techniques, including segmentation proxy models and recurrent reduced-rate tracking, are developed for speed acceleration. With the extracted tracks, common selection and aggregation queries can be answered with sub-second latency.

2.3 Vehicle Trajectory Prediction

Vehicle trajectory prediction has been a well-studied problem. Mainstream approaches are learning-based and rely on past states of traffic participants in a given scene to estimate their future states.

For instance, [20, 38, 41] adopt traditional machine learning methods to predict potential future movements in short/medium-term action spaces.

Recent methods as mmTransformer [34], MFP [45], CoverNet [37], HOME [12] and DenseTNT [13] are deep learning based and leverage historical trajectory information of the target and scene information (road network maps etc.) for semantic feature encoding. Readers can refer to a recent survey [18] for more details.

These approaches often require sufficient historical states or abundant contextual information, which are not available in our problem setting. In this paper, we intend to predict the future trajectory of a moving object simply from its first observation in a sampled video frame.

2.4 Low-Frame Rate Multi-Object Tracking

The challenge of multi-object tracking under a low sampling rate has also been noticed in the computer vision community. CenterTrack [56] uses the center point of objects as the tracking point, jointly training detection and tracking. By inputting RGB images of consecutive frames and the predicted heatmap of the previous frame, it learns temporal and spatial information together for robust tracking. APPTracker [55] augments CenterTrack with an additional tracking head to identify unreliable displacement estimates at low frame rates. ColTrack [33] inserts an information refinement module between every two temporal encoders to better fuse temporal cues and refine features. A tracking object consistency loss is proposed to guide the fusion procedure.

3 OVERVIEW OF LEAP

In this section, we undertake a theoretical analysis on the lower bound of video frame sampling and present an overview of our sampling scheme, which comprises a predictive sampling strategy, a data-driven motion manager, a robust object tracker and a distilled object detector. Details of these components will be explained in the subsequent sections.

3.1 Optimal Sampling Strategy

Recall that each object o_i is captured by the surveillance camera within an interval $[s_i, e_i]$, where s_i represents the initial frame and e_i captures the final occurrence. To infer the minimum number of sampled frames that cover all moving objects, we can formulate it as a minimum interval stabbing problem.

Definition 1 (Minimum Interval Stabbing Problem [8]). *Given n intervals that we have $I = \{[s_1, e_1], [s_2, e_2], \ldots, [s_n, e_n]\}$, the minimum interval stabbing problem finds a point set $X = \{x_i, \ldots, x_k\}$ with the minimum size and for every interval $[s_i, e_i]$, there exists a point $x_j \in X$ such that $x_j \in [s_i, e_i]$.*

It has been proved that the problem can be optimally solved by the greedy heuristic based on *leftmost right endpoint* (LRE).

Theorem 1. *Let α be the number of sampled frames by LRE algorithm. It requires at least α frames to cover each moving object once [8].*

Example 1. *We use Figure 3 as an illustrative example to explain LRE algorithm. Suppose there are 11 objects to be extracted and each object o_i appears in video frames $[s_i, e_i]$. We sort them in ascending order of e_i. In the first iteration, e_1 is selected and the four intervals stabbed by $x = e_1$ (i.e., $[s_1, e_1]$, $[s_2, e_2]$, $[s_3, e_3]$ and $[s_4, e_4]$ in this example) are removed. In the second iteration, e_5 is selected and intervals for o_5, o_6 and o_7 are removed. Finally, e_8 is selected and we obtain the optimal point set $\{e_1, e_5, e_8\}$. When these three frames are sampled, all the moving objects can be covered.*

Algorithm 1: Leftmost Right Endpoint (LRE)

Input: Intervals $I = \{[s_1, e_1], [s_2, e_2], \ldots, [s_n, e_n]\}$
Output: Minimum stabbing point set X

```
Sort [s_i, e_i] in increasing order of right endpoint e_i;
while I is not empty do
    Take the first interval [s_t, e_t] from I ;
    Add its right endpoint e_t to the stabbing set X ;
    Remove all intervals in I that contain e_t ;
end
return X ;
```

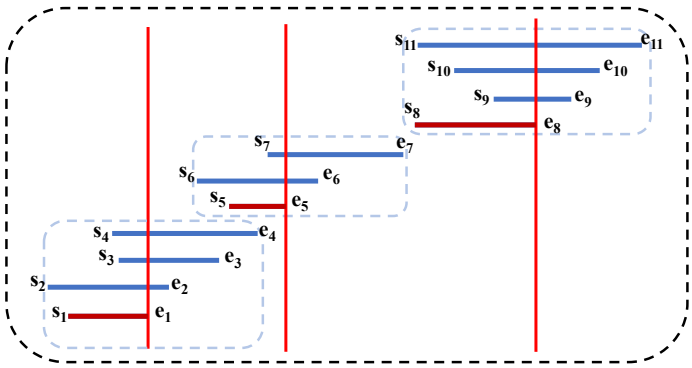


Fig. 3. An illustrative example for LRE algorithm.

In practice, with only one observation of o_i, it becomes exceedingly difficult to accurately extract its trajectory from the video clip. Thus, we are motivated to study the 2-frame optimalty. In the following, we theoretically prove the minimum number of video frames in which each moving object is captured at least twice.

Theorem 2. *Let α denote the number of sampled frames by LRE algorithm. The minimum number of frames to guarantee that each moving object appears at least two times is 2α.*

Proof. Without loss of generality, let $\{e_1, e_2, \ldots, e_\alpha\}$ be the sampled frames by the LRE algorithm and $e_1 < e_2 < \ldots < e_\alpha$. We first prove by contradiction that intervals $[s_i, e_i]$ and $[s_j, e_j]$ with $i < j$ are disjoint. Suppose these two intervals are overlapped, since $e_i < e_j$, we have $s_i < s_j < e_i < e_j$. However, when we process e_i in the LRE algorithm, all the intervals containing e_i will be removed. This leads to a contradiction that e_j occurs in the output of LRE algorithm. Therefore, to guarantee that these disjoint intervals $[s_1, e_1], [s_2, e_2], \ldots, [s_\alpha, e_\alpha]$ are stabbed twice, we need at least 2α sampled frames. □

Example 2. *We still use Figure 3 to explain the 2-frame optimality and how to sample the 2α frames. The sampled frames by the LRE algorithm are $\{e_1, e_5, e_8\}$ and their intervals $[s_1, e_1]$, $[s_5, e_5]$ and $[s_8, e_8]$ are disjoint. We need at least 6 sampled frames to guarantee that objects o_1, o_5 and o_8 are observed twice.*

To obtain these 6 frames, we can re-use the 3 sampled frames by LRE algorithm. The remaining frames are selected from the rightmost left endpoint among the group of intervals intersecting with $\{e_1, e_5, e_8\}$, respectively. For example, $[s_5, e_5]$, $[s_6, e_6]$ and $[s_7, e_7]$ intersect with e_5 and s_7 is the rightmost

left endpoint. So we can select it as the sampled frame. Similarly, s_3 and s_9 are selected and we obtain the 6 sampled frames $\{s_3, e_1, s_7, e_5, s_9, e_8\}$.

These two optimality measures serve as theoretical lower bounds for video frame sampling and provide guidance for our LEAP algorithm when operating at a low sampling rate.

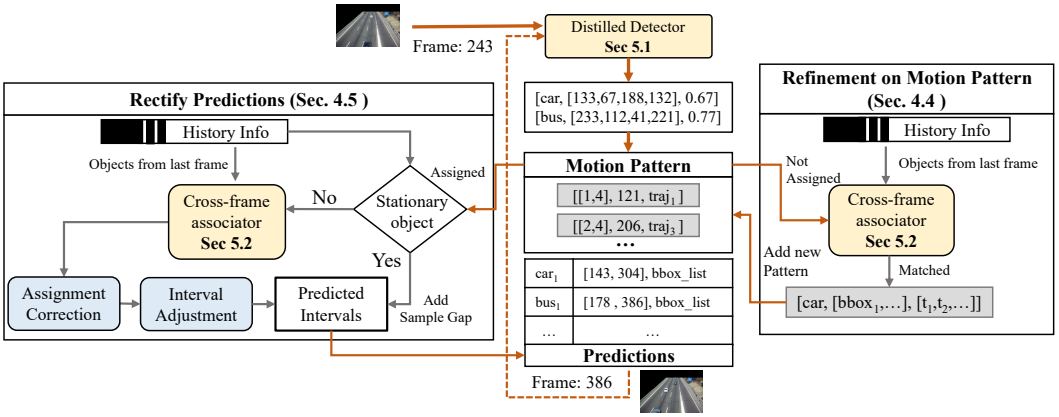


Fig. 4. Predictive sampling framework of LEAP.

3.2 Predictive Sampling Framework

To support efficient MOT-based view materialization, this paper presents a practical approach aimed at minimizing the number of processed video frames while maintaining comparable video query accuracy to existing video databases. In contrast to the uniform down-sampling technique employed in OTIF [4], we propose a novel predictive sampling scheme called LEAP, which works well when the sampling rate is set in proximity to the 2-frame optimality. The efficacy of LEAP can be attributed to two key elements:

(1) We investigate video query processing in the context of video big data generated by urban-scale surveillance cameras deployed in road networks for smart city applications. LEAP is able to leverage the inherent motion patterns of moving vehicles or pedestrians to enhance trajectory prediction accuracy.
(2) LEAP represents the first attempt to incorporate multi-object tracking between two frames with a substantial temporal gap. The tracking mechanism of LEAP extensively exploits spatial, temporal, and visual attributes to maximize the utilization of available information.

The predictive sampling framework of LEAP is depicted in Figure 4. Given an input video clip or live stream, LEAP is employed to extract trajectories of all moving objects. The framework consists of three primary modules. Firstly, the Motion Pattern Manager is responsible for acquiring motion patterns of moving objects through trajectory clustering. Given that surveillance cameras in real-world applications capture long videos of dynamic scenes, LEAP continually updates the acquired motion patterns according to the extracted trajectories. For each detected object in a sampled video frame, this module assigns the most probable trajectory as the predicted motion. Leveraging this assigned trajectory, we can estimate the future frame in which the object will exit the camera's field of view and determine the subsequent sampling frame.

Though the task of next frame prediction can be viewed as a regression problem, we did not adopt regression models for two reasons. First, our approach is data-driven and unsupervised, but

constructing a regression model requires training data acquisition and supervised training. Second and more importantly, owing to the component of motion pattern refinement, our approach is adaptive to dynamic traffic environments. In contrast, regression models have to be re-trained when there occurs data drift (e.g., traffic accidents or other scenarios not covered in the training data). This step incurs additional online training cost, which is unbearable compared with the total ingestion time.

Given that object detection is a frequently executed and computationally intensive operation within the multi-object tracking task, we have also explored potential approaches to reduce the associated overhead. To address this concern, we employ knowledge distillation techniques [15] to construct a compact object detector. Through this process, we can derive a computationally lightweight network that maintains a high level of accuracy. In the context of object tracking, the task of cross-frame object association poses significant challenges. This task can be seen as an entity linking problem involving two sets of detected objects, and it becomes particularly challenging due to the disparity of object size, instances of partial occlusion, and substantial temporal gap between two frames. To address these challenges, LEAP relies on the cross-frame association module, which fully exploits the spatial, temporal and visual information to provide a robust mechanism for object tracking. Details of these core modules will be presented in the subsequent sections. In Section 4, we will discuss the module of the motion pattern manager, followed by the object detection model distillation and cross-frame object association in Section 5.

Algorithm 2: Predictive Sampling Algorithm

```
Initialize motion patterns from the first m frames;
cur ← m; prev ← m; Trajs ← ∅;
while cur is not the last frame do
    Perform object detection on frame cur;
    if f > m then
        obj_pair ← perform object association between frames prev and cur;
        Update motion patterns based on obj_pair;
    end
    prev ← cur;
    for each detected object o_i do
        Trajs ← Trajs ⋃ T_o;
        if o_i is a new object then
            Assign the most probable trajectory T_o to o_i;
            Estimate the exit frame f_e from T_i;
            cur ← max(cur, f_e);
        end
    end
end
return Trajs;
```

The pseudo code for the predictive sampling scheme employed by LEAP is presented in Algorithm 2. LEAP incorporates a warm-up stage to initialize the motion patterns. In this stage, a limited number of video frames (4 minutes in our implementation) are processed, by employing existing approaches to extract the trajectories of moving objects. These trajectories are clustered via K-medoids clustering and each cluster is associated with a representative trajectory. With the

initialized motion manager, we progressively predict the next sampling frame based on the objects that first appear in the current frame. For each sampled frame *cur*, we perform object detection and associate the detected objects with those from the previously sampled video frame. If the trajectories derived from the matching pairs do not exist in the motion pattern manager, we perform motion pattern update to incorporate unseen trajectories. For each new object in frame *cur*, we estimate its exit frame according to the maintained travel patterns. The maximum interval is selected as the next frame to sample. More details will be presented in Section 4.

4 DATA-DRIVEN MOTION MANAGER

This section studies multi-object tracking under an extremely low sampling rate. Under such a situation, traditional tracking strategies encounter performance degradation [33, 55], which has also been verified in our empirical study for OTIF in Figure 2. To tackle the challenge, we introduce a pivotal component of our LEAP algorithm, the Data-driven Motion Manager, specifically designed to handle the challenges associated with sparse sampling situations. We show the overall process of this section in Figure 4.

4.1 The Concept of Motion Pattern

In real scenarios, moving objects are influenced by inherent patterns [52] and scene-specific elements (e.g., lane markings and traffic signals). Leveraging such scene-specific data for autonomous driving applications has been investigated in earlier research [43, 48]. As illustrated in Figure 5, potential trajectories can be selected from a series of action candidate sets, such as moving straight, turning left, or turning right. Vehicles may follow one of the actions to generate trajectories that can be clustered to reveal inherent motion patterns. Hence, we define a motion pattern as a representative trajectory in the cluster and denote it as a triplet $\langle traj_{id}, len, [bbox_1, bbox_2, ..., bbox_t]\rangle$, where $traj_{id}$ is an associated trajectory id, len indicates motion duration and each $bbox_i$ stores the information of bounding box in a specific frame.

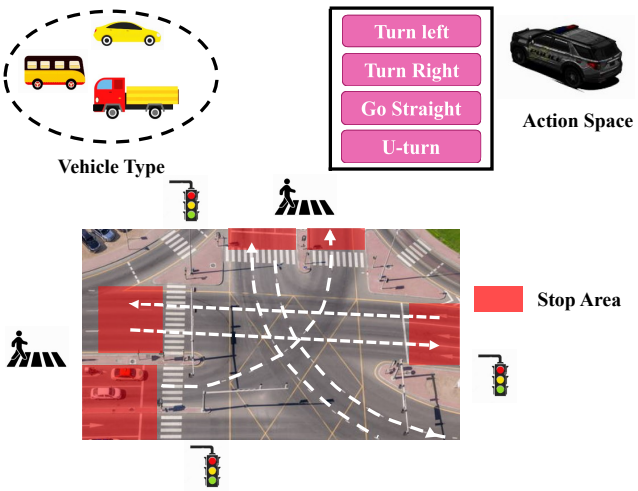


Fig. 5. An illustrated example of motion pattern.

4.2 Motion Pattern Initialization

LEAP requires a pre-processing stage to initialize the motion patterns. Given an input video stream or video clip, we use the first few minutes of the video frames and perform multi-object tracking *without down-sampling* using an existing algorithm [54]. Among output trajectories, we only maintain those with moderate length to perform trajectory clustering. This is because short trajectories may be generated by inaccurate object tracking with id switch and long trajectories may be caused by stationary objects.

Afterwards, we present our trajectory clustering algorithm to capture the inherent motion patterns and select a representative trajectory within each cluster. As to trajectory similarity, we adopt the Fréchet distance [10, 50], which takes into account the location and ordering of the points along the trajectories. The trajectories are clustered using K-Mediods [26] and we employ the Davies-Bouldin Index(DBI) [7] to determine the proper number of clusters.

With the clustered trajectories, we take an additional step to merge clusters with similar trends in motion behavior. As shown in Figure 6, we identified several areas where objects enter the frame based on the starting point of the original trajectories. We annotated each region with a unique identifier. Then we assign the two ends of each trajectory to these regions and obtain the corresponding identifier as $traj_{id}$ for each trajectory in the motion pattern. Through such merge step, We add the information of coarse-grained similarity between trajectories, which provides additional reference for subsequent pattern assignments.

Square Northeast Jackson Town

Fig. 6. Entering Area captured by LEAP.

4.3 Pattern Assignment on Moving Object

Given a detected object, we predict the object trajectory by assigning it to an existing pattern $\langle traj_{id}, len, [bbox_1, bbox_2, ..., bbox_t]\rangle$ and use the sequence of bounding boxes $[bbox_1, ..., bbox_t]$ as the output of trajectory prediction. In the following, we present our approach of motion pattern assignment.

When an object first appears at time t in our observation, its center point is detected as $(x_c^{(t)}, y_c^{(t)})$. This positional information guides the assignment of candidate patterns from our motion pattern manager. For each trajectory candidate $traj$ in the pattern candidate set $\boldsymbol{P}$, we measure the distance d_{traj} between the observed point and the candidate trajectory as:

$$d_{traj} = \min \sqrt{(x_c^{(t)} - x_i)^2 + (y_c^{(t)} - y_i)^2} \tag{1}$$

Taking into account the inherent variability in object motion space, it may not be immediately possible to assign a unique pattern candidate. For trajectories belonging to the same $traj_{id}$, we select the one with the closest distance within each category, that satisfies the distance threshold, which is $d_{traj} < \lambda_d$, as the candidate.

Following this, we obtain an initial candidate pattern set $\boldsymbol{P}$ and store intermediate information $[obj_{id}, \boldsymbol{P}, img, bbox_t, d_{traj}]$ for the observed object, where obj_{id} is the allocated object id, img is the appearance and $bbox_t$ is the observed location at time t.

For each object, the pattern with the trajectory that is closest to the current observed location is chosen to generate the initial output. We take the trajectory $traj$ and moving interval len in the assigned closest pattern as our initial prediction and prepare for possible refinement in subsequent steps. To predict the motion interval of each target at current timestamp t, we apply predicted trajectory information $traj$ and the nearest neighbor point p_m at time m that matches the current detection to compute the predicted interval $[t - m, t + (len - m)]$. Then we select the next sampling frame based on intervals of the detected objects that disappear at the latest following our predictive sampling strategy.

4.4 Motion Pattern Refinement

We propose a refinement mechanism to handle dynamic traffic or unseen trajectories that are not maintained in the global motion patterns. When we perform object association between two frames, for any matching pair (obj_{cur}, obj_{prev}), if obj_{cur} cannot be allocated to any existing motion patterns, we need to construct a new motion pattern to cover the new trajectory generated by obj_{cur}. Otherwise, we know obj_{cur} can be assigned to an existing pattern with trajectory $traj$. In this case, if the positions of obj_{cur} and obj_{prev} match $traj$, we check the temporal consistency between the two sampled frames and $traj$. If the deviation exceeds a pre-defined threshold, we need to update len_t to match the temporal gap between obj_{cur} and obj_{prev}, reflecting traffic update.

4.5 Rectifying Predicted Output

To further enhance the accuracy of predicted output, we propose several post-processing strategies to rectify the parsed results. The whole process is shown in the left part of Figure 4.

Stationary Objects Analysis: We noted that long-term stationary vehicles bring two challenges. First, they add unnecessary computational overhead. Second, they may cause the propagation of prediction errors. As such, we propose a mechanism to figure out stationary targets between two adjacent frames. If two objects demonstrate the same object size and identical position between two neighboring sampled frames, we directly consider them as a matching pair, without the need for further processing. Consequently, we enlarge the predicted interval with the sample gap between the current and the last sampled frame.

Correction of Trajectory Assignment: At this stage, we have linked each object with a set of candidate trajectories and several corresponding observation points. To facilitate retrospective access to past information, LEAP additionally stores intermediate information during the parsing process as described in Section 4.3. This information encompasses historical position data $bbox_h$, timestamps t_h, appearance images img_h, as well as initially allocated candidate patterns $\boldsymbol{P}_h$. As these trajectories in $\boldsymbol{P}$ are obtained from past video streams, discrepancies with the actual trajectories are unavoidable. Therefore, we utilize the real-time observed data to refine the predicted trajectories.

Each observed object that is linked to a candidate trajectory set from the previous matching process will be assigned a unique candidate trajectory. If an object is detected only once, we select the trajectory in patterns with the closest distance to the observation point to generate its final prediction. For those detected multiple times, we choose the trajectory from the intersection of all assigned candidate patterns. If multiple trajectories remain after all observations, the one of the shortest average distances between actual observations is selected as the final predicted trajectory.

Interval Adjustment: Differences in object moving speeds, traffic light influences, and congestion may cause the object's predicted and actual travel times to diverge. In such cases, we correct the trajectory's interval time. Given a predicted trajectory $traj_i = (p_m, p_{m+1}, ..., p_n)$ from time point m

to n, and the actual observation point o_t with the nearest point position p_o at time t_p, and if $t_p \neq t$, then we adjust the predicted interval using an approximate scaling factor, $scale = \frac{t-m}{t_p-m}$:

$$interval = (t - (t_p - m) * scale, t + (n - t_p) * scale) \tag{2}$$

5 OBJECT DETECTION AND TRACKING

In this section, we present our techniques in terms of object detection and tracking. Our goal is to derive a compact object detection model with higher efficiency and a resilient object associator capable of linking objects across frames with a large temporal gap.

5.1 Object Detection Model Distillation

In OTIF, object detection is a frequent and expensive operator in the task of multi-object tracking. The operation is applied at least once in each sampled frame. To reduce the overhead of this component, a feasible solution is to construct a compact detector without sacrificing accuracy. Knowledge distillation [15] is commonly deemed as an effective model compression approach to derive a compact student model trained under the supervision of a larger teacher network. Our distilled object detection model is trained only once and does not require further re-training. This is because we use a large training dataset to ensure its generalization performance.

In this paper, we adopt [35] as the backbone distillation model for object detection, which transfers the knowledge of a more accurate teacher network to a computationally light-weight student network and can process hundreds of frames per second. It customizes Tiny-YOLO [39] as the architecture for the student network and formulates the distillation loss as objectness scaled function. The objectness is defined as the IoU (Intersection over Union) between the detected bounding box of a moving object and its ground truth. For those static objects in the background region, they are associated with low objectness values. The overall idea of the new distillation loss is to learn the bounding box coordinates and class probabilities only when objectness value of the teacher prediction is high.

Besides the distillation loss $\mathcal{L}_d$ used in [35] to enforce the consistency between the detection results from the teacher network and student network, we also take into account feature-level distillation loss. We adopt FPN (Feature Pyramid Network) [31] to extract multi-scale semantic information from the video frames. We incorporate an additional feature-level distillation loss to transfer the knowledge of these features from the teacher network and improve the performance of the student network:

$$\mathcal{L}_f = \frac{1}{CHW} \sum_{k=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} \left(F^T_{k,i,j} - f(F^S_{k,i,j}) \right)^2 \tag{3}$$

where, F^T is the visual feature generated by the teacher network, F^S is the feature from the student and f is the adaptation layer to reshape the F^S to the same dimension as F^T. H and W refer to the height and width of the features. C is the channel. With $\mathcal{L}_f$, the extended loss function in our distillation network is represented as:

$$\mathcal{L} = \mathcal{L}_d + \lambda \cdot \mathcal{L}_f \tag{4}$$

In our implementation, we adopt the popular detection model YOLOv5-L [19] as the teacher network, which has been pre-trained using the COCO dataset [32]. To improve the detection accuracy for the surveillance cameras deployed in road network, we also enhance the training dataset to cover more vehicle types, including PASCAL VOC [42], Stanford Car Dataset [9], Object365 [27], and VisDrone [57]. Among the training samples in these datasets, we retain the image samples whose labels are related to vehicles. The student network is trained to minimize the distillation loss in

Equation 4 as well as the detection loss, whose goal is to provide accurate object detection in the enhanced dataset.

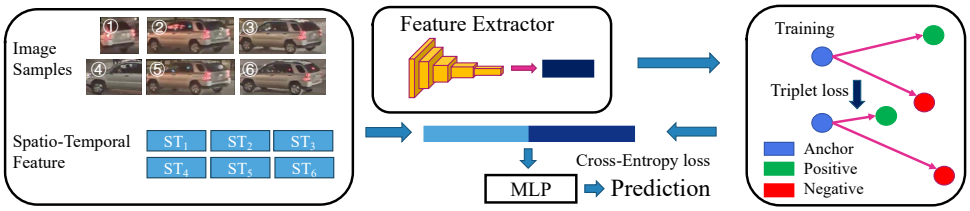


Fig. 7. The training procedure of object association model.

5.2 Cross-Frame Object Association

Given two video frames containing detected objects enclosed by bounding boxes, the cross-frame object association task aims to identify all matching pairs that refer to the same real-world entity from these two sets of detected objects. The component plays a vital role in generating correct output trajectories and is also beneficial for the trajectory initialization and update in the motion manager.

The task is particularly challenging when applied in the predictive sampling framework employed by LEAP. Recall that LEAP builds upon a predictive framework, wherein we progressively predict the next sampling frame F_{k+1} based on the objects that first appear in F_k. Thereby, we set F_{i+1} as the frame capturing the last occurrence of these objects. Consequently, we encounter a series of challenges. First, there is a notable disparity in object size for the same moving object. Smaller size of objects also imply more blurry images and less prominent outlook features. This can be observed in Figure 8, where an object is initially captured in the k-th sampled frame but appears significantly smaller in the subsequent $k + 1$ frame. Second, objects are prone to partial occlusions, as also demonstrated in the k-th and $k + 1$ frames of Figure 8. Finally, the temporal gap between two sampled frames in LEAP is considerably large. This poses a substantial obstacle to applying motion information as a clue for cross-frame object association.

Fig. 8. An example of cross-frame object association.

To tackle the aforementioned challenges, we propose a robust object association model based on [14, 30] that fully exploits the spatial, temporal and visual information. Two moving objects that refer to the same real-world entity have to be similar in visual appearance and consistent in spatial-temporal motion patterns. The overall framework of the model is illustrated in Figure 7. We collect data for different objects at different times and divide the corresponding features into vision-based and motion-based parts. For a specific object, the visual feature v is obtained through a pre-trained ResNet backbone network. We learn the appearance features in a supervised manner. Specifically, we choose the triplet loss as the loss function. Training samples are divided into anchor, positive and negative samples where the anchor and positive samples belong to the same object, while the negative samples belong to different objects.

Here we describe the training details of our cross-frame association module. Through the form of triplet loss [11, 46] in Equation 5, we draw a closer distance between positive samples and expand that of negative samples. In the triplet loss, v^a is the representation of anchor sample; v^p and v^n denote the corresponding representation of positive and negative samples, respectively. m is a margin parameter enforced between a pair of positive and negative samples. Lastly, $D(:,:)$ is the distance measure between a pair of object features.

$$\mathcal{L}_{triplet} = \sum_{i}^{N} [D(v_i^a, v_i^p) - D(v_i^a, v_i^n) + m] \tag{5}$$

While training the appearance features, we additionally apply spatio-temporal features which consist of position feature $bbox$ and timestamp t for each sample as a complement to the object features. For example, in each epoch during the training process, for each input pair to be matched, we will concatenate the appearance feature supervised by triplet loss during training to obtain v_c. Similarly, the spatio-temporal features are also concatenated together to get ST_c. Then, we apply a fully-connected layer for v_c and ST_c separately to generate transformed features. The transformed v_c and ST_c are combined into a final representation vector. This representation vector is then put into MLP (Multi-Layer Perceptron) module, which finally yields the predicted matching result. As the association task can also be viewed as a binary classification task, we use Cross-Entropy loss as the classification loss, y is the ground-truth label and $\hat{y}$ is the predicted possibility to be matched, the loss function can be described as:

$$\mathcal{L}_{ce} = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y})) \tag{6}$$

During the training phase, the two losses $\mathcal{L}_{triplet}$ and $\mathcal{L}_{ce}$ are optimized over the same set of input pairs. More specifically, $\mathcal{L}_{triplet}$ is used to optimize the appearance representation learning and $\mathcal{L}_{ce}$ is applied to determine whether the input pair belong to the same object. By jointly optimizing $\mathcal{L}_{triplet}$ and $\mathcal{L}_{ce}$, the association model is trained to effectively learn both appearance and spatio-temporal information to improve the accuracy of object association. The former loss focuses on enhancing the discriminative capability of appearance features, while the latter one combines spatio-temporal information with appearance features for the association task. This enables the association model to capture both visual and spatio-temporal clues to make accurate association decisions.

The search space for object association between two frames F_i and F_{i+1} consists of all pairs of detected objects. Since our object association model requires visual feature extraction, which is computationally expensive, we also propose a filtering mechanism to reduce the number of candidate pairs for the association task.

Our overall idea is that if a detected object is highly consistent with its predicted motion pattern, we think there is no need to apply the object association model for further verification. Otherwise, the object has been deviated from the predicted trajectory. In this case, object association is performed to identify the true trajectory and then we can rectify the underlying motion patterns.

Based on this idea, we introduce the condition for motion consistency. Given a detected object obj_{cur} at frame F_{i+1}, we first perform motion pattern assignment in Section 4.3 to find its candidate motion patterns. Afterwards, we compare the similarity of assigned patterns of obj_{cur} with that of all detected objects in the previous frame F_i. If we can find a candidate pattern for an object obj_{prev} that is highly similar to that of obj_{cur}, we consider obj_{cur} and obj_{prev} belong to the same entity. The motion pattern similarity is defined as the intersection ratio of $traj_{id}$ in the motion pattern candidate set.

Table 1. Real video datasets.

Dataset	Abbrev.	Train	Test	Resolution	# Objects	Avg. Traj. Len	Target Object Selectivity	Scene
Jackson Town	JT	60min	60min	1920 × 1080	1311	13.5s	(car:0.95, truck: 0.12, bus:0.02)	Night Street
Taipei Hires	Taipei	60min	60min	1280 × 720	940	13.35s	(car:0.62, truck: 0.46, bus:0.41)	T-Junction
Caldot1	Cal1	60min	50min	720 × 480	394	3.41s	(car:0.14, truck: 0.06)	Mountain Roads
Caldot2	Cal2	60min	40min	720 × 480	1335	4.22s	(car:0.88, truck: 0.24)	Highway
Tokyo	Tokyo	60min	51min	960 × 540	1534	4.95s	(car:0.71, truck: 0.33, bus:0.21)	Downtown
Adventure Rentals	AR	60min	60min	1920 × 1080	4061	4.74s	(car:0.78, truck: 0.32, bus:0.01)	Highway
Square Northeast	SN	60min	60min	1920 × 1080	241	9.47s	(car:0.33, truck: 0.1)	Crossroads

Table 2. Details of video queries.

Query type	Query Template	Query Instance
Selection	Q_1-Q_3: Select frames that contain {objects}	[car,truck,bus]
Aggregation	Q_4-Q_6: Count the number of {objects} in each video frame	[car,truck,bus]
Top-K	Q_7-Q_9: Find Top-10 frames with largest number of {objects} in the whole video	[car,truck,bus]

Following the Intersection over Union (IoU) matching concept, we calculate the overlap of moving intervals as follows:

$$overlap = \begin{cases} \frac{|e_t^i - s_{t-1}^i|}{\max(e_t^i, e_{t-1}^i) - \min(s_t^i, s_{t-1}^i)}, & \text{if}(e_t^i > s_{t-1}^j \ \& \ s_t^i < e_{t-1}^j) \\ 0, & \text{Otherwise} \end{cases} \tag{7}$$

If the overlap is lower than a threshold $\lambda_{overlap}$, we employ the cross-frame association module for further evaluation on whether they belong to the same object.

6 EXPERIMENTS

In this section, we compare LEAP with state-of-the-art video query processing systems and evaluate the performance in terms of both efficiency and accuracy.

6.1 Experimental Setup

Video Datasets We use 7 real video datasets for performance evaluation. Five of them, namely Jackson Town, Tokyo, Taipei Hires, Caldot1 and Caldot2, are sourced from OTIF [4]. Jackson Town captures the downtown traffic during nighttime, whereas Tokyo portrays morning peak-hour scenarios. Taipei Hires showcases a T-junction with diverse vehicles. Caldot1 and Caldot2 consist of low-resolution videos captured by California DOT cameras along two highways. Detailed information regarding these datasets is available in Table 1.

Besides the aforementioned datasets, we also collected two new datasets from YouTube live streams, including Adventure Rentals[1] and Square Northeast[2]. As shown in Table 1, Adventure Rentals contains the fewest number of objects among all the datasets, whereas Square Northeast exhibits the highest object density.

Following the setting of OTIF, each dataset includes a training subset to for model training and a test subset for performance evaluation. In terms of the target query objects, we have also presented their selectivity in Table 1. Generally speaking, "car" is the most frequent class label. The label "truck" exhibits greater frequency than "bus", which could be absent in certain datasets.

Video Queries. We examine three types of popular video queries adopted in previous studies, including selection queries [3, 4, 21, 22], aggregation queries [3, 4, 21] and top-k queries [28]. Each

[1]https://www.youtube.com/watch?v=NOXb1637dcM
[2]https://www.youtube.com/watch?v=qYwYrTpoboc

query type encompasses three distinct instances. There are 9 queries in total, with their details presented in Table 2.

Comparison Approaches. Since MOT-based view materialization is the only approach to supporting all the queries outlined in Table 2, we consider OTIF [4] as the primary competitor. We also include MIRIS as a baseline, which was originally designed for real-time object tracking. MIRIS [3] can be viewed as a precedent work of OTIF which is developed by the same research team. CenterTrack [56] is also included as a low-frame rate computer vision-based method.

There also exist numerous video query optimization systems tailored for specific query types. Most existing database systems are tailored for specific types of queries. As summarized in Table 3, NoScope [22] is specifically designed to accelerate selection query processing and Everest [28] is tailored for top-k query optimization. TASTI [24] employs an auxiliary model to derive query-specific scores utilizing generated frame embeddings. BlazeIt [21] focus on optimizing counting the number of objects in a frame with proxy models. Thus, it can also be used to support selection queries Q_1-Q_3 and top-k queries Q_7-Q_9 as well. Our LEAP, together with OTIF, MIRIS and CenterTrack, belong to the category of MOT-based view materialization. They can leverage the output of object trajectories and classes to support the three types of queries simultaneously.

Table 3. Supported query types for different baselines

	Methods	Selection	Aggr.	Top-K
MOT-based View Materialization	LEAP	✓	✓	✓
	MIRIS	✓	✓	✓
	OTIF	✓	✓	✓
	CenterT	✓	✓	✓
Query-specific Proxy Model	BlazeIt	✓	✓	✓
	NoScope	✓		
	TASTI	✓		
	Everest			✓

Performance Metrics. For selection queries, the ground truth result set comprises video frames that contain the query object and we use F_1-score as the evaluation metric. For aggregation queries, we adopt mean absolute percentage error (MAPE) to measure the estimation error, which is defined as $MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_i}$. Regarding top-$k$ queries, we employ precision as the performance metric, which measures the fraction of correct results among the top k returned items.

Implementation Details In the implementation of LEAP, we use a 4-minute video clip in each dataset to initialize the motion pattern manager. As to trajectory clustering, we adopt k-medoids clustering with Fréchet distance measure. Our object association model is trained using the training data, which is also used to train the RNN-tracker of OTIF.

As to the comparison approaches, the implementations of OTIF, MIRIS, NoScope, TASTI, BlazeIt, CenterTrack and Everest have been open-sourced and we follow their instructions to obtain the experimental results. It is also worth noting that the original implementation of OTIF splits the input video clip into multiple segments (60 as the default setting) and maintain a pool of object detection services for parallel processing. To ensure a fair comparison, we disable the video segment split function and maintain only one object detector. Nonetheless, we are also interested in comparing OTIF and LEAP when parallel materialization is allowed. Thus, we add an experiment in Section 6.7 to evaluate their throughput when processing large-scale video streaming sources.

Experimental Environment. To make the video decoding cost consistent among different datasets, we uniformly convert the video clips into MP4 format with H.264 encoding. All our experiments

are conducted on a server installed with Ubuntu 22.04 and CUDA 11. The hardware settings include Intel i9-10980Xe CPU, 256GB of memory, 2TB SSD hard drive, 4TB HDD hard drive, and two NVIDIA RTX 3090ti GPUs.

6.2 Sampling Results for MOT-based View Materialization

Our first experiment examines the sampling results of the MOT-based view materialization approaches, including LEAP, OTIF and MIRIS, performance of sampling and assess their deviation from optimality. Since OTIF and MIRIS allow adjustable sampling rate, for fair comparison, we present the number of sampled frames required by these approaches to achieve comparable query performance. The sampling outcomes for each approach across all datasets are presented in Table 4. OTIF exhibits superior performance compared to MIRIS. This can be attributed to OTIF's utilization of selective local region object detection models in conjunction with a recurrent neural network (RNN) for object tracking. Compared with OTIF, our LEAP can further reduce the number of sampled frames by more than 7× in five out of the seven datasets. This compelling outcome serves to validate the effectiveness of our sampling scheme.

To measure the gap to the sampling optimality, we use Opt-1 and Opt-2 to denote the 1-frame optimality and 2-frame optimality, respectively. Remarkably, LEAP achieves comparable performance to Opt-2 in the majority of datasets. In Caldot2 and Adventure Rental, LEAP's sampled frame count is even lower than that of Opt-2. The reason is that these two datasets capture highway scenes where vehicle motion patterns are simple and relatively easy to predict. Consequently, the trajectories of certain vehicles can be accurately reconstructed with just a single sampled snapshot.

Table 4. Sampling results.

Dataset	Opt-1	Opt-2	LEAP	OTIF	MIRIS	CenterT
JT	214	428	495	3375	5548	3600
Taipei	272	544	719	3375	4473	3600
Cal1	114	228	279	2815	2784	8908
Cal2	294	588	479	4107	2550	6570
Tokyo	256	512	912	3875	3370	6198
AR	644	1288	945	6750	9521	7200
SN	96	192	370	3375	3476	3600

We select Q_1, Q_4, and Q_7 as three representative queries within each query type. We selected configurations on MIRIS and OTIF that have the closest accuracy to LEAP. The results presented in Table 5 demonstrate that LEAP offers comparable performance to OTIF among the selected queries when applying the sampling rates reported in Table 4. However, the MOT-based ingestion cost of LEAP is significantly lower than that of OTIF. In the Square Northeast dataset, LEAP is 7 times faster than OTIF. In the Jackson Town dataset, LEAP achieves a 5× speedup. CenterTrack achieves comparable performance with OTIF in the datasets of Jackson Town and Adventure Rentals. However, it is remarkably inferior to OTIF in Caldot1 and Caldot2. The reason is that, as shown in Table 4, CenterTrack sampled considerably more frames to achieve comparable performance.

We also observe that despite the notable reduction in sampling rate in the Caldot1 dataset, the improvement in end-to-end processing time is not as substantial. To reveal the underlying reasons, we conduct a break-down analysis to investigate the time cost of each component. The results in Figure 9 show that the object detection cost is indeed the performance bottleneck in OTIF, but no

longer plays a dominant role in LEAP. Therefore, even though the time cost of object detection is positively correlated to the number of sampled frames, the total query processing time of LEAP is less sensitive.

Careful readers may find that for OTIF, the sum of the processing time for each component is higher than the time cost reported in Table 5. The reason for this discrepancy is that although we have disabled parallel object detectors in OTIF, its segmentation proxy model and object detector are deployed as two separate and parallel services, with the former running on CPU and the latter on GPU.

Table 5. Time cost and query performance given the sampling rates reported in Table 4. Better query results are highlighted in bold.

Method	Query	JT	Taipei	Cal1	Cal2	Tokyo	AR	SN
LEAP	$Q_1 \uparrow$	**0.91**	0.83	**0.58**	**0.75**	0.73	**0.89**	**0.57**
	$Q_4 \downarrow$	0.59	**0.39**	0.11	**0.32**	0.34	**0.44**	0.54
	$Q_7 \uparrow$	0.8	0.7	**0.9**	0.8	**1.0**	0.8	0.7
	Time	**36s**	**75s**	**21s**	**28s**	**68s**	**81s**	**26s**
OTIF	$Q_1 \uparrow$	**0.91**	**0.86**	0.54	0.71	**0.82**	0.88	0.47
	$Q_4 \downarrow$	0.52	0.62	0.11	**0.32**	**0.24**	0.67	**0.47**
	$Q_7 \uparrow$	0.8	0.9	**0.9**	**0.9**	0.9	**0.9**	0.7
	Time	231s	235s	58s	81s	163s	285s	189s
CenterT	$Q_1 \uparrow$	**0.91**	0.84	0.52	0.68	0.67	0.86	0.56
	$Q_4 \downarrow$	**0.50**	0.56	**0.09**	0.47	0.25	0.66	0.68
	$Q_7 \uparrow$	**0.9**	**1.0**	**0.9**	0.8	0.7	0.8	**0.9**
	Time	254s	163s	212s	155s	157s	350s	262s

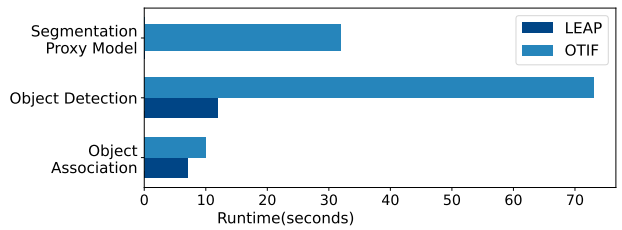
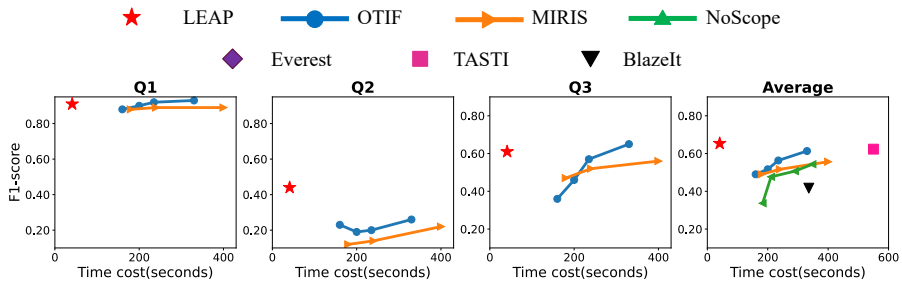


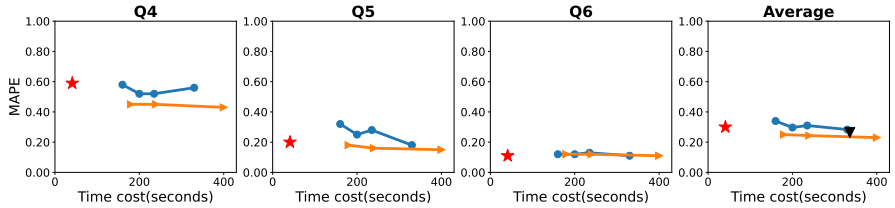
Fig. 9. Time cost breakdown for the procedure of multi-object tracking in the Caldot1 dataset.

6.3 Performance on Video Query Processing

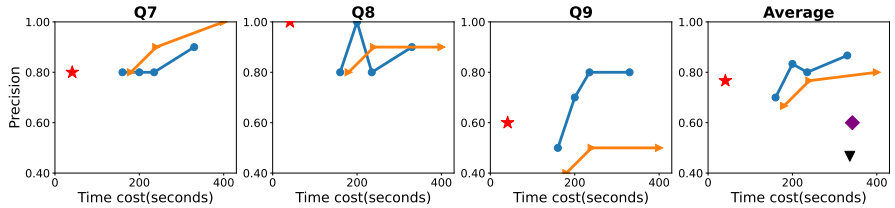
In this experiment, we evaluate the trade-off between running time and the quality of results obtained for the complete set of queries listed in Table 2. Note that OTIF, MIRIS and LEAP belong to the category of MOT-based view materialization. They can swiftly respond to online queries by harnessing offline video ingestion and index construction. We denote the total cost as $cost_p + m * cost_q$, where $cost_p$ is the time cost for offline view materialization and $cost_q$ represents incurred per individual online query, and m denotes the total number of queries. The time cost of these three approaches is primarily dominated by $cost_p$, and it becomes more advantageous for them when assessing the average processing cost across multiple queries. In contrast, the remaining baselines,



(a) Performance of selection queries $Q_1 - Q_3$ in Jackson Town.



(b) Performance of aggregation queries Q_4-Q_6 in Jackson Town.



(c) Performance of top-k queries $Q_7 - Q_9$ in Jackson Town.

Fig. 10. Performance of query processing from Q_1 to Q_9.

including NoScope, BlazeIt, TASTI, and Everest need to re-scan the videos for every incoming query.

Figure 10 presents a comparative analysis of LEAP against OTIF and MIRIS for individual queries. Additionally, we incorporate additional baselines for multi-query processing and examine the average time cost and query performance. Due to space limit, we only present the outcomes in Jackson Town as the representative dataset.

For selection queries Q_1, since "car" is the most prevalent object, all MOT-based approaches achieve high accuracy. However, in the case of the "truck" query, lower F_1-scores are observed due to misclassification of vans and pick-up trucks, being erroneously identified as "cars" by the object detection models. Regarding selection queries targeting frames containing a "bus", even though the selectivity is nearly zero in Jackson Town, the promising F_1-score can be attributed to distinguishable visual features between buses and cars. In the context of the multi-query scenario, we further incorporate NoScope, BlazeIt and TASTI as the baselines for video selection queries and report the average query processing time and F_1-scores for Q_1-Q_3. Our LEAP also clearly outperforms these baselines. For aggregation and top-k queries, while it is possible that LEAP may not dominate OTIF in terms of both query accuracy and processing time, LEAP still stands out as the most efficient approach across all query types.

In aggregation queries Q_4-Q_6, MIRIS can achieve lower estimation error than OTIF because OTIF adopts RNN for object tracking, which may fail to correctly capture the motion pattern when the object stays static for a long period, waiting for red traffic light. LEAP's accuracy could be inferior to OTIF in certain scenarios of aggregation and top-k queries because these two types of queries require accurate prediction of the time interval for each moving object captured by the camera. However, since LEAP leverages very sparse sampling rate to achieve high efficiency, it's difficult for LEAP to yield accurate interval prediction.

Overall, LEAP excels when handling selection queries. Even though its sampling rate is significantly lower than the baselines, its predictive sampling scheme is able to cover the moving objects with high recall. LEAP is also superior in handling object queries with lower selectivity (e.g., "bus" and "truck"). With more robust object tracking mechanism, LEAP incurs fewer number of object mismatches.

6.4 Cost of Pre-processing

Both OTIF and LEAP involve a pre-processing stage. In OTIF, a training data set is required to train its RNN-based tracker and jointly tune its segmentation proxy model, object detector, and the recurrent tracker. In LEAP, we need to train an object association model and distill a compact object detector. Additionally, our motion pattern manager requires an initialization overhead. This experiment aims to evaluate the pre-processing cost of OTIF and LEAP by considering the running time and the number of video frames used.

As depicted in Figure 11, the model training process for OTIF exceeds 7 hours, accompanied by an additional 50 minutes for joint parameter tuning. In LEAP, the training time for the object association model and the distilled object detector spans approximately 4 hours and 2 hours, respectively. Our hyper-parameters are tuned manually, without leveraging AutoML techniques to explore the parameter space. OTIF involves an automatic tuning component to find the optimal settings for the parameters of input resolution, sampling rate and segmentation model settings. The cost is reported in Figure 11. For the other hyper-parameters such as the confidence threshold of detector, OTIF treats them in the same way as LEAP. The initialization time for the motion pattern manager is a mere 15 minutes, which is negligible when compared to the total pre-processing time. It is worth noting that the pre-processing cost incurs a one-time investment. Subsequently, both OTIF and LEAP can be deployed on the target data source for continuous processing of video streams.

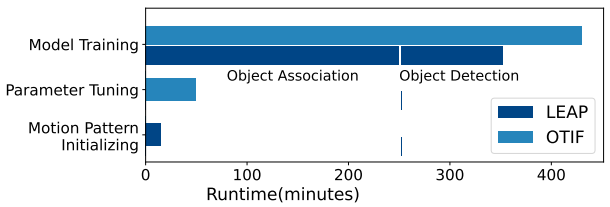


Fig. 11. Pre-processing cost for LEAP and OTIF in Caldot1.

Since LEAP requires the information from target video to initialize the motion pattern manager, we present in Table 6 the duration of the warm-up stage employed from the 5 datasets. In comparison, from the default training data settings in the source code of OTIF, we find that it requires the usage of a one-hour subset extracted from target video, in order to facilitate the training of the recurrent neural network (RNN) tracker and joint parameter tuning.

Table 6. The video length sampled for pre-processing.

Method	Jackson Town	Taipei	Caldot1	Caldot2	Tokyo
LEAP	6 min	4 min	4 min	4 min	4 min
OTIF	60 min	60 min	60 min	60 min	60 min

6.5 Accuracy of Cross-Frame Association

The accuracy of our proposed cross-frame association module, measured by the proportion of correct predictions, is shown in Table 7. The results are promising as the accuracy is located within $[0.84, 0.92]$ among the datasets, which verified the effectiveness to correctly link two detected objects from two sampled frames with a large temporal gap.

Table 7. Accuracy of cross-frame association.

Dataset	JT	Taipei	AR	SN	cal1	cal2	Tokyo
Accuracy	0.84	0.87	0.92	0.86	0.84	0.85	0.86

6.6 Ablation Study

6.6.1 Effect of Trajectory Clustering. In the ablation study of trajectory clustering, we construct a baseline by removing this component and the motion pattern assignment is performed on all extracted trajectories in the pre-processing stage. The results in Table 8 show that without clustering, we can observe considerable query performance degradation in certain queries, such as Q_4 in Jackson Town, where motion pattern assignment may be affected by outlier trajectories.

Table 8. Ablation study for trajectory clustering.

Dataset	Method	$Q_1 \uparrow$	$Q_4 \downarrow$	$Q_7 \uparrow$
Jackson Town	LEAP	**0.91**	**0.59**	0.8
	w/o clustering	0.90	0.66	0.8
Adventure Rental	LEAP	**0.88**	0.47	0.7
	w/o clustering	0.87	0.47	0.7
Taipei Hires	LEAP	**0.83**	**0.39**	0.7
	w/o clustering	0.80	0.43	0.7

6.6.2 Effect of Distilled Object Detector. To validate the usefulness of the distilled object detector, we conduct an ablation study, the findings of which are presented in Table 9. We compare the full version of LEAP with its counterpart that lacks the knowledge distillation component. The results demonstrate that the incorporation of the distilled model contributes to an enhanced trade-off between query performance and efficiency. The distilled object detector brings efficiency improvement by $3 \sim 9$ seconds, which results in considerable time saving in the module of object detection. For example, it takes around 10 seconds in Caldot1 to perform object detection with the distilled model (as shown in Figure 9). If we remove this component, the detection cost increases by 3 seconds. This implies that object distillation can reduce the detection cost by $3/13 \approx 23\%$.

Table 9. Ablation study of distilled object detector.

Dataset	Method	$Q_1 \uparrow$	$Q_4 \downarrow$	$Q_7 \uparrow$	Time
JT	LEAP	0.91	**0.59**	**0.8**	**36s**
	w/o distill	**0.92**	0.61	0.7	41s
AR	LEAP	0.88	0.47	0.7	**72s**
	w/o distill	**0.89**	**0.44**	**0.8**	81s
Taipei	LEAP	0.83	0.39	0.7	**75s**
	w/o distill	**0.86**	**0.38**	**0.8**	82s
Cal1	LEAP	**0.58**	0.11	**0.9**	**21s**
	LEAP w/o distill	**0.58**	**0.10**	**0.9**	24s

We also evaluate the effect of the two losses in our model distillation. Originally, YOLOv5 was adopted as the student network and we report the detection accuracy when distilled with different losses. The results in Table 10 show that the detection accuracy of YOLOv5s+$\mathcal{L}_d$+$\mathcal{L}_f$ is considerably better than the vanilla YOLOv5 without distillation. When we remove either loss in the distillation, the detection accuracy also declines, showing that both losses play a positive effect.

Table 10. Ablation study on different distillation losses.

Model	mAP@50	mAP@50-95
YOLOv5s+$\mathcal{L}_d$+$\mathcal{L}_f$	0.754	0.553
YOLOv5s+$\mathcal{L}_d$	0.745	0.542
YOLOv5s+$\mathcal{L}_f$	0.749	0.547
YOLOv5s (vanilla)	0.724	0.522

6.7 Throughput for Parallel Materialization

In the aforementioned experiments, we deliberately disabled the parallel setting in the OTIF system to ensure a fair comparison between the different approaches. However, in this particular experiment, we enable parallel processing for both OTIF and LEAP, allowing us to fully leverage the available computational resources. The primary objective here is to evaluate the throughput of parallel view materialization.

To facilitate our analysis, we have gathered a dataset consisting of 200 video clips, each with a uniform duration of 20 minutes. These video clips serve as individual data sources, representing live video streams. Our main focus is to determine the maximum number of video streams that a single GPU card can process in real time. Upon examining the results depicted in Figure 12, we observe that to keep pace with the speed of video data generation, OTIF can handle fewer than 40 video streams with one GPU card. In other words, to process a staggering total of 1000 video streams in real time, OTIF requires more than 25 GPU cards. In contrast, LEAP demonstrates a significantly higher throughput. With just one GPU card, LEAP can effectively process 160 video streams in real time. This equals to a 4× reduction in hardware expenditure when dealing with large-scale video streams.

7 CONCLUSION

In this paper, we studied MOT-based view materialization in video database and established two types of optimality measures, serving as theoretical lower bounds of sampling. We proposed a

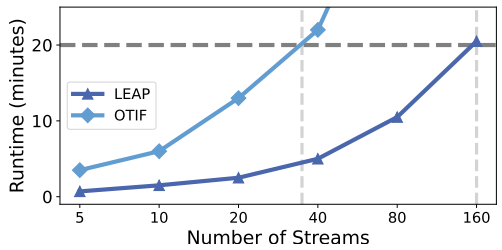


Fig. 12. Comparison of processing time of different numbers of input streams.

predictive sampling framework which incorporates a data-driven motion manager and a robust object association strategy to connect the detected objects in the sampled frames. We conducted comprehensive experiments that validated the superiority of LEAP over OTIF. With comparable query performance, LEAP requires up to 9× fewer sampled video frames in most datasets. With one GPU card, it can perform MOT-based view materialization upon 160 video streams in real time. Despite its outstanding performance, LEAP still has great room for improvement in terms of generality to handle dynamic update of camera orientation or more complex motion patterns, especially from pedestrians. We consider them as our future research directions.

ACKNOWLEDGMENTS

This work was sponsored by the National Key Research and Development Project of China (2022YFF0902000), the Key Research Program of Zhejiang Province(Grant No.2023C01037) and CCF-Huawei Populus Grove Fund.

REFERENCES

[1] Michael R. Anderson, Michael J. Cafarella, Germán Ros, and Thomas F. Wenisch. 2019. Physical Representation-Based Predicate Optimization for a Visual Analytics Database. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*. IEEE, 1466–1477. https://doi.org/10.1109/ICDE.2019.00132

[2] Jack Baker, Paul Fearnhead, Emily B. Fox, and Christopher Nemeth. 2019. Control variates for stochastic gradient MCMC. *Stat. Comput.* 29, 3 (2019), 599–615. https://doi.org/10.1007/s11222-018-9826-2

[3] Favyen Bastani, Songtao He, Arjun Balasingam, Karthik Gopalakrishnan, Mohammad Alizadeh, Hari Balakrishnan, Michael J. Cafarella, Tim Kraska, and Sam Madden. 2020. MIRIS: Fast Object Track Queries in Video. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 1907–1921. https://doi.org/10.1145/3318464.3389692

[4] Favyen Bastani and Samuel Madden. 2022. OTIF: Efficient Tracker Pre-processing over Large Video Datasets. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 2091–2104. https://doi.org/10.1145/3514221.3517835

[5] Jiashen Cao, Karan Sarkar, Ramyad Hadidi, Joy Arulraj, and Hyesoon Kim. 2022. FiGO: Fine-Grained Query Optimization in Video Analytics. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 559–572. https://doi.org/10.1145/3514221.3517857

[6] Pramod Chunduri, Jaeho Bang, Yao Lu, and Joy Arulraj. 2022. Zeus: Efficiently Localizing Actions in Videos using Reinforcement Learning. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 545–558. https://doi.org/10.1145/3514221.3526181

[7] David L. Davies and Donald W. Bouldin. 1979. A Cluster Separation Measure. *IEEE Trans. Pattern Anal. Mach. Intell.* 1, 2 (1979), 224–227. https://doi.org/10.1109/TPAMI.1979.4766909

[8] Jeff Erickson. 2019. *Algorithms*. http://jeffe.cs.illinois.edu/teaching/algorithms/

[9] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. 2015. The Pascal Visual Object Classes Challenge: A Retrospective. *Int. J. Comput. Vis.* 111, 1 (2015), 98–136. https://doi.org/10.1007/s11263-014-0733-5

[10] M Maurice Fréchet. 1906. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo (1884-1940)* 22, 1 (1906), 1–72.

[11] Adhiraj Ghosh, Kuruparan Shanmugalingam, and Wen-Yan Lin. 2021. Relation Preserving Triplet Mining for Stabilizing the Triplet Loss in Vehicle Re-identification. *CoRR* abs/2110.07933 (2021). arXiv:2110.07933 https://arxiv.org/abs/2110.07933

[12] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. 2021. HOME: Heatmap Output for future Motion Estimation. In *24th IEEE International Intelligent Transportation Systems Conference, ITSC 2021, Indianapolis, IN, USA, September 19-22, 2021*. IEEE, 500–507. https://doi.org/10.1109/ITSC48978.2021.9564944

[13] Junru Gu, Chen Sun, and Hang Zhao. 2021. DenseTNT: End-to-end Trajectory Prediction from Dense Goal Sets. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 15283–15292. https://doi.org/10.1109/ICCV48922.2021.01502

[14] Lingxiao He, Xingyu Liao, Wu Liu, Xinchen Liu, Peng Cheng, and Tao Mei. 2020. FastReID: A Pytorch Toolbox for General Instance Re-identification. *CoRR* abs/2006.02631 (2020). arXiv:2006.02631 https://arxiv.org/abs/2006.02631

[15] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. *CoRR* abs/1503.02531 (2015). arXiv:1503.02531 http://arxiv.org/abs/1503.02531

[16] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodík, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B. Gibbons, and Onur Mutlu. 2018. Focus: Querying Large Video Datasets with Low Latency and Low Cost. In *13th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2018, Carlsbad, CA, USA, October 8-10, 2018*, Andrea C. Arpaci-Dusseau and Geoff Voelker (Eds.). USENIX Association, 269–286. https://www.usenix.org/conference/osdi18/presentation/hsieh

[17] Bo Hu, Peizhen Guo, and Wenjun Hu. 2022. Video-zilla: An Indexing Layer for Large-Scale Video Analytics. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 1905–1919. https://doi.org/10.1145/3514221.3517840

[18] Yanjun Huang, Jiatong Du, Ziru Yang, Zewei Zhou, Lin Zhang, and Hong Chen. 2022. A Survey on Trajectory-Prediction Methods for Autonomous Driving. *IEEE Trans. Intell. Veh.* 7, 3 (2022), 652–674. https://doi.org/10.1109/TIV.2022.3167103

[19] Glenn Jocher. 2020. *YOLOv5 by Ultralytics*. https://doi.org/10.5281/zenodo.3908559

[20] Joshua Mason Joseph, Finale Doshi-Velez, Albert S. Huang, and Nicholas Roy. 2011. A Bayesian nonparametric approach to modeling motion patterns. *Auton. Robots* 31, 4 (2011), 383–400. https://doi.org/10.1007/s10514-011-9248-x

[21] Daniel Kang, Peter Bailis, and Matei Zaharia. 2019. BlazeIt: Optimizing Declarative Aggregation and Limit Queries for Neural Network-Based Video Analytics. *Proc. VLDB Endow.* 13, 4 (2019), 533–546. https://doi.org/10.14778/3372716.3372725

[22] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: Optimizing Deep CNN-Based Queries over Video Streams at Scale. *Proc. VLDB Endow.* 10, 11 (2017), 1586–1597. https://doi.org/10.14778/3137628.3137664

[23] Daniel Kang, John Guibas, Peter Bailis, Tatsunori Hashimoto, Yi Sun, and Matei Zaharia. 2021. Accelerating Approximate Aggregation Queries with Expensive Predicates. *Proc. VLDB Endow.* 14, 11 (2021), 2341–2354. https://doi.org/10.14778/3476249.3476285

[24] Daniel Kang, John Guibas, Peter D. Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2022. TASTI: Semantic Indexes for Machine Learning-based Queries over Unstructured Data. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 1934–1947. https://doi.org/10.1145/3514221.3517897

[25] Daniel Kang, Ankit Mathur, Teja Veeramacheneni, Peter Bailis, and Matei Zaharia. 2020. Jointly Optimizing Preprocessing and Inference for DNN-based Visual Analytics. *Proc. VLDB Endow.* 14, 2 (2020), 87–100. https://doi.org/10.14778/3425879.3425881

[26] Leonard Kaufman and Peter J Rousseeuw. 2009. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons.

[27] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 2013. 3D Object Representations for Fine-Grained Categorization. In *2013 IEEE International Conference on Computer Vision Workshops, ICCV Workshops 2013, Sydney, Australia, December 1-8, 2013*. IEEE Computer Society, 554–561. https://doi.org/10.1109/ICCVW.2013.77

[28] Ziliang Lai, Chenxia Han, Chris Liu, Pengfei Zhang, Eric Lo, and Ben Kao. 2021. Top-K Deep Video Analytics: A Probabilistic Approach. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 1037–1050. https://doi.org/10.1145/3448016.3452786

[29] Guoliang Li, Xuanhe Zhou, Shifu Li, and Bo Gao. 2019. QTune: A Query-Aware Database Tuning System with Deep Reinforcement Learning. *Proc. VLDB Endow.* 12, 12 (2019), 2118–2130. https://doi.org/10.14778/3352063.3352129

[30] Zepeng Li, Dongxiang Zhang, Yanyan Shen, and Gang Chen. 2023. Human-in-the-Loop Vehicle ReID. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, Brian Williams, Yiling Chen, and Jennifer Neville (Eds.). AAAI Press, 6048–6055. https://doi.org/10.1609/AAAI.V37I5.25747

[31] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. 2017. Feature Pyramid Networks for Object Detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 936–944. https://doi.org/10.1109/CVPR.2017.106

[32] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V (Lecture Notes in Computer Science, Vol. 8693)*, David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). Springer, 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

[33] Yiheng Liu, Junta Wu, and Yi Fu. 2023. Collaborative Tracking Learning for Frame-Rate-Insensitive Multi-Object Tracking. *CoRR* abs/2308.05911 (2023). https://doi.org/10.48550/arXiv.2308.05911 arXiv:2308.05911

[34] Yicheng Liu, Jinghuai Zhang, Liangji Fang, Qinhong Jiang, and Bolei Zhou. 2021. Multimodal Motion Prediction With Stacked Transformers. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 7577–7586. https://doi.org/10.1109/CVPR46437.2021.00749

[35] Rakesh Mehta and Cemalettin Öztürk. 2018. Object Detection at 200 Frames per Second. In *Computer Vision - ECCV 2018 Workshops - Munich, Germany, September 8-14, 2018, Proceedings, Part V (Lecture Notes in Computer Science, Vol. 11133)*, Laura Leal-Taixé and Stefan Roth (Eds.). Springer, 659–675. https://doi.org/10.1007/978-3-030-11021-5_41

[36] Oscar R. Moll, Favyen Bastani, Sam Madden, Mike Stonebraker, Vijay Gadepally, and Tim Kraska. 2022. ExSample: Efficient Searches on Video Repositories through Adaptive Sampling. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*. IEEE, 2956–2968. https://doi.org/10.1109/ICDE53745.2022.00266

[37] Tung Phan-Minh, Elena Corina Grigore, Freddy A. Boulton, Oscar Beijbom, and Eric M. Wolff. 2020. CoverNet: Multimodal Behavior Prediction Using Trajectory Sets. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 14062–14071. https://doi.org/10.1109/CVPR42600.2020.01408

[38] Shaojie Qiao, Dayong Shen, Xiaoteng Wang, Nan Han, and William Zhu. 2015. A Self-Adaptive Parameter Selection Trajectory Prediction Approach via Hidden Markov Models. *IEEE Trans. Intell. Transp. Syst.* 16, 1 (2015), 284–296. https://doi.org/10.1109/TITS.2014.2331758

[39] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, Faster, Stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 6517–6525. https://doi.org/10.1109/CVPR.2017.690

[40] Francisco Romero, Johann Hauswald, Aditi Partap, Daniel Kang, Matei Zaharia, and Christos Kozyrakis. 2022. Optimizing Video Analytics with Declarative Model Relationships. *Proc. VLDB Endow.* 16, 3 (2022), 447–460. https://www.vldb.org/pvldb/vol16/p447-romero.pdf

[41] Mao Shan, Stewart Worrall, and Eduardo Mario Nebot. 2011. Long term vehicle motion prediction and tracking in large environments. In *14th International IEEE Conference on Intelligent Transportation Systems, ITSC 2011, Washington, DC, USA, October 5-7, 2011*. IEEE, 1978–1983. https://doi.org/10.1109/ITSC.2011.6082922

[42] Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. 2019. Objects365: A Large-Scale, High-Quality Dataset for Object Detection. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 8429–8438. https://doi.org/10.1109/ICCV.2019.00852

[43] Wenbo Shao, Yanchao Xu, Jun Li, Chen Lv, Weida Wang, and Hong Wang. 2023. How Does Traffic Environment Quantitatively Affect the Autonomous Driving Prediction? *CoRR* abs/2301.04414 (2023). https://doi.org/10.48550/arXiv.2301.04414 arXiv:2301.04414

[44] Tao Sun, Jianqiu Xu, and Caiping Hu. 2022. An Efficient Algorithm of Star Subgraph Queries on Urban Traffic Knowledge Graph. *Data Sci. Eng.* 7, 4 (2022), 383–401. https://doi.org/10.1007/S41019-022-00198-0

[45] Yichuan Charlie Tang and Ruslan Salakhutdinov. 2019. Multiple Futures Prediction. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 15398–15408. https://proceedings.neurips.cc/paper/2019/hash/86a1fa88adb5c33bd7a68ac2f9f3f96b-Abstract.html

[46] Kilian Q. Weinberger and Lawrence K. Saul. 2009. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *J. Mach. Learn. Res.* 10 (2009), 207–244. https://doi.org/10.5555/1577069.1577078

[47] Ziyang Xiao, Dongxiang Zhang, Zepeng Li, Sai Wu, Kian-Lee Tan, and Gang Chen. 2023. DoveDB: A Declarative and Low-Latency Video Database. *Proc. VLDB Endow.* 16, 12 (2023), 3906–3909. https://doi.org/10.14778/3611540.3611582

[48] Yanchao Xu, Wenbo Shao, Jun Li, Kai Yang, Weida Wang, Hua Huang, Chen Lv, and Hong Wang. 2022. SIND: A Drone Dataset at Signalized Intersection in China. In *25th IEEE International Conference on Intelligent Transportation Systems, ITSC 2022, Macau, China, October 8-12, 2022*. IEEE, 2471–2478. https://doi.org/10.1109/ITSC55140.2022.9921959

[49] Hao Yu, Xi Guo, Xiao Luo, Weihao Bian, and Taohong Zhang. 2023. Construct Trip Graphs by Using Taxi Trajectory Data. *Data Sci. Eng.* 8, 1 (2023), 1–22. https://doi.org/10.1007/S41019-023-00205-Y

[50] Dongxiang Zhang, Zhihao Chang, Sai Wu, Ye Yuan, Kian-Lee Tan, and Gang Chen. 2022. Continuous Trajectory Similarity Search for Online Outlier Detection. *IEEE Trans. Knowl. Data Eng.* 34, 10 (2022), 4690–4704. https://doi.org/10.1109/TKDE.2020.3046670

[51] Dongxiang Zhang, Zhihao Chang, Dingyu Yang, Dongsheng Li, Kian-Lee Tan, Ke Chen, and Gang Chen. 2023. SQUID: subtrajectory query in trillion-scale GPS database. *VLDB J.* 32, 4 (2023), 887–904. https://doi.org/10.1007/S00778-022-00777-7

[52] Dongxiang Zhang, Teng Ma, Junnan Hu, Yijun Bei, Kian-Lee Tan, and Gang Chen. 2023. Co-movement Pattern Mining from Videos. *CoRR* abs/2308.05370 (2023). https://doi.org/10.48550/ARXIV.2308.05370 arXiv:2308.05370

[53] Ji Zhang, Yu Liu, Ke Zhou, Guoliang Li, Zhili Xiao, Bin Cheng, Jiashu Xing, Yangtao Wang, Tianheng Cheng, Li Liu, Minwei Ran, and Zekang Li. 2019. An End-to-End Automatic Cloud Database Tuning System Using Deep Reinforcement Learning. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, Peter A. Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska (Eds.). ACM, 415–432. https://doi.org/10.1145/3299869.3300085

[54] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. 2022. ByteTrack: Multi-object Tracking by Associating Every Detection Box. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXII (Lecture Notes in Computer Science, Vol. 13682)*, Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer, 1–21. https://doi.org/10.1007/978-3-031-20047-2_1

[55] Tao Zhou, Wenhan Luo, Zhiguo Shi, Jiming Chen, and Qi Ye. 2022. APPTracker: Improving Tracking Multiple Objects in Low-Frame-Rate Videos. In *MM '22: The 30th ACM International Conference on Multimedia, Lisboa, Portugal, October 10 - 14, 2022*, João Magalhães, Alberto Del Bimbo, Shin'ichi Satoh, Nicu Sebe, Xavier Alameda-Pineda, Qin Jin, Vincent Oria, and Laura Toni (Eds.). ACM, 6664–6674. https://doi.org/10.1145/3503161.3548162

[56] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. 2020. Tracking Objects as Points. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part IV (Lecture Notes in Computer Science, Vol. 12349)*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer, 474–490. https://doi.org/10.1007/978-3-030-58548-8_28

[57] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Heng Fan, Qinghua Hu, and Haibin Ling. 2022. Detection and Tracking Meet Drones Challenge. *IEEE Trans. Pattern Anal. Mach. Intell.* 44, 11 (2022), 7380–7399. https://doi.org/10.1109/TPAMI.2021.3119563

Received July 2023; revised October 2023; accepted November 2023