RESEARCH-ARTICLE

MAST: Towards Efficient Analytical Query Processing on Point Cloud Data

JIANGNENG LI, Nanyang Technological University, Singapore City, Singapore

HAITAO YUAN, Nanyang Technological University, Singapore City, Singapore

GAO CONG, Nanyang Technological University, Singapore City, Singapore

HAN MAO KIAH, Nanyang Technological University, Singapore City, Singapore

SHUHAO ZHANG, Huazhong University of Science and Technology, Wuhan, Hubei, China

Proceedings of the ACM on Management of Data, Volume 3, Issue 1 (February 2025)
https://doi.org/10.1145/3709702
EISSN: 2836-6573

MAST: Towards Efficient Analytical Query Processing on Point Cloud Data

JIANGNENG LI, Nanyang Technological University, Singapore
HAITAO YUAN, Nanyang Technological University, Singapore
GAO CONG, Nanyang Technological University, Singapore
HAN MAO KIAH, Nanyang Technological University, Singapore
SHUHAO ZHANG, Huazhong University of Science and Technology, China

The proliferation of 3D scanning technology, particularly within autonomous driving, has led to an exponential increase in the volume of Point Cloud (PC) data. Given the rich semantic information contained in PC data, deep learning models are commonly employed for tasks such as object queries. However, current query systems that support PC data types do not process queries on semantic information. Consequently, there is a notable gap in research regarding the efficiency of invoking deep models for each PC data query, especially when dealing with large-scale models and datasets. To address this issue, this work aims to design an efficient approximate approach for supporting PC analysis queries, including PC retrieval and aggregate queries. In particular, we propose a novel framework that delivers approximate query results efficiently by sampling core PC frames within a constrained budget, thereby minimizing the reliance on deep learning models. This framework is underpinned by rigorous theoretical analysis, providing error-bound guarantees for the approximate results if the sampling policy is preferred. To achieve this, we incorporate a multi-agent reinforcement learning-based approach to optimize the sampling procedure, along with an innovative reward design leveraging spatio-temporal PC analysis. Furthermore, we exploit the spatio-temporal characteristics inherent in PC data to construct an index that accelerates the query process. Extensive experimental evaluations demonstrate that our proposed method, MAST, not only achieves accurate approximate query results but also maintains low query latency, ensuring high efficiency.

CCS Concepts: • **Information systems → Database query processing**.

Additional Key Words and Phrases: Point cloud data query processing, DB4AI

ACM Reference Format:
Jiangneng Li, Haitao Yuan, Gao Cong, Han Mao Kiah, and Shuhao Zhang. 2025. MAST: Towards Efficient Analytical Query Processing on Point Cloud Data . *Proc. ACM Manag. Data* 3, 1 (SIGMOD), Article 52 (February 2025), 27 pages. https://doi.org/10.1145/3709702

1 Introduction

With advancements in 3D scanning technology and its widespread application in fields such as autonomous driving and robotics, Point Cloud (PC) data [38] has increased significantly. For ease of understanding, PC data is defined as a set of 3-dimensional points $\mathsf{P} = \{(x, y, z) \in \mathbb{R}^3\}$ that capture environmental contexts. The inherent spatial features in PC data enable the extraction of detailed

Authors' Contact Information: Jiangneng Li, Nanyang Technological University, Singapore, jiangnen002@e.ntu.edu.sg; Haitao Yuan, Nanyang Technological University, Singapore, haitao.yuan@ntu.edu.sg; Gao Cong, Nanyang Technological University, Singapore, gaocong@ntu.edu.sg; Han Mao Kiah, Nanyang Technological University, Singapore, hmkiah@ntu.edu.sg; Shuhao Zhang, Huazhong University of Science and Technology, China, shuhao_zhang@hust.edu.cn.

ACM 2836-6573/2025/2-ART52
https://doi.org/10.1145/3709702

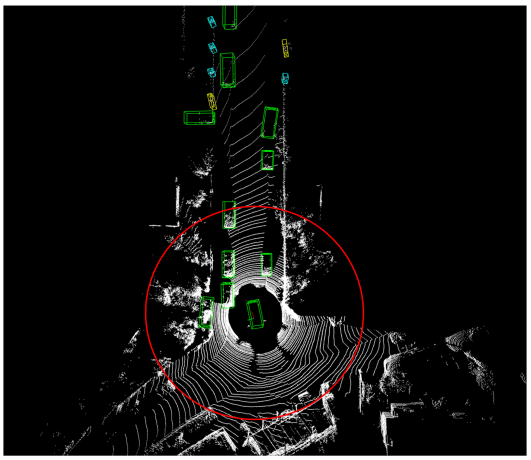


Fig. 1. A visualization of querying on PC data with objects detected, where the bounding boxes indicate objects, and the red circle indicates a spatial query filter.

3D semantic information, such as the precise localization of objects. For instance, PC data provides a more robust source of spatial information than 2D images in the autonomous driving scenario.

Given the rich semantic content of PC data, there are numerous requirements for analyzing and querying semantic information from these datasets. Current research has primarily focused on enhancing the accuracy of deep models in tasks such as 3D classification [35, 41], object detection [39, 40, 48], and semantic segmentation [16, 36], which often leverage deep learning techniques like deep neural networks. As shown in Fig. 1, querying PC data involves not only retrieving semantic objects identified through detection methods but also evaluating spatial predicates. However, to the best of our knowledge, no system supports such PC queries containing predicates on semantic information, which are extracted using deep learning models. Current database systems that support the PC data type, such as PostgreSQL pgPointcloud [34] and Ganos [46], treat the PC data as typical spatial objects to support queries with spatial predicates (e.g., retrieving PC data within a query region), but not queries with semantic predicates.

We proceed to introduce two kinds of such PC queries (i.e., the *PC retrieval query* and the *PC aggregate query*) in the following example.

Example 1.1. Given the vast volume of PC data generated by autonomous driving systems, automotive companies want to query the data to detect specific events. For example, a query might seek to identify high-risk collision scenarios, defined by the presence of three or more cars within a specified radius around the vehicle capturing the data. This scenario can be formalized as finding instances where the count of detected cars is three or more (i.e., $\texttt{Count}(\texttt{Car}) \geq 3$) and each car is within a certain distance (i.e., $\texttt{Distance}(\texttt{Car}, \texttt{center}) < \texttt{r}$) from the LiDAR-equipped vehicle. This retrieval query can be represented in an SQL as follows, where each frame `PC_frames.frame` denotes a 3D point set (formally defined in Section 2), and $\texttt{f}_\texttt{M}$ denotes the function that processes PC frames with the deep model `M` designed for the object detection task, which outputs the set of bounding boxes of objects:

```
- - - - - - - - - - - - PC Retrieval Query - - - - - - - - - - - -
  SELECT * FROM PC_frames.frame WHERE
  (   SELECT COUNT(*)
```

```
    FROM f_M(PC_frames.frame) AS cars
    WHERE Distance(cars.loc, frame.center) < r
) >= 3
```

As another example, the company may want to determine whether a driver tends to drive close to neighboring cars or maintain a safe distance. To evaluate the driving habit, the query would calculate the average number of neighboring cars during the vehicle's operation (i.e., Avg(Count(Cars))). This aggregate query is formulated as follows.

```
- - - - - - - - - - - - PC Aggregate Query - - - - - - - - - - -
SELECT Avg(count_car.num) FROM
(   SELECT COUNT(*)
    FROM f_M(PC_frames.frame) AS cars
    WHERE Distance(cars.loc, frame.center) < r
) AS count_car.num
```

To bridge the gap between analysis requirements and current PC database management systems, this paper aims to develop a processing framework for supporting analytical queries with deep learning models on PC data. Processing PC data with deep models incurs high computational costs, For instance, analyzing a single PC frame with an NVIDIA GeForce RTX 2080 Ti GPU takes approximately 0.1 seconds for object prediction. Therefore, due to the large volume of PC data and the computational intensity of deep model inference, it is impractical to answer the aforementioned queries by preprocessing all PC frames using deep models [26], and then analyzing the processed structured information through a data warehouse disregarding the raw PC data , even when amortized across multiple queries. Worse still, the efficiency of executing queries within PC database systems remains relatively unexplored, while advancements in the analysis of PC data have focused primarily on accuracy metrics. This raises a critical, yet underexplored aspect: *Considering the high computational cost of processing PC data with deep models, how do we optimize the efficiency of queries on PC data?*

One idea to address this challenge is to design lightweight models (referred to as proxy models) as replacements for the original costly model (referred to as the oracle model). However, proxy models are often specialized for particular tasks, such as detecting vehicles or pedestrians, and require expertise in techniques like model distillation to design. Consequently, creating a lightweight model that performs well across diverse queries is challenging. Inspired by the work [3], which proposes addressing video analytical queries with approximate results by sampling video frames and processing the sampled frames using deep learning models, we tackle the challenge by generating approximate query results, aiming to answer analytical queries on PC data both efficiently and effectively while reducing the deep model usage needed.

We focus on analytical queries involving both semantic predicates and spatial predicates for PC data. We introduce a query processing framework that reduces the need for deep model usage by processing only a subset of the PC data. By leveraging the inherent spatio-temporal features in PC data, our framework generates approximate query results for multiple queries based on the sampled and processed PC data. Unlike previous work on video data [3, 4], which are query-driven methods (optimizing the query performance with a query workload as the input), we propose a method that enables PC frame sampling for multiple queries without relying on the query workload as the input during the sampling process. Our framework has the advantage of the generality of model selection and the stable efficiency improvement proportional to the sampling budget (as to

be explained in Section 7.2). We develop a query processing framework that includes a sampling procedure to sample a fraction of PC data and construct an index storing the information provided by deep models, and a querying procedure that generates query results based on the index.

A key challenge in this context is optimizing the querying performance with spatio-temporal information utilized, as different data points may vary in their significance and contribution to the overall query efficiency [3, 21]. To tackle this, we model the sampling process as a decision-making problem. We introduce a hierarchical multi-agent framework that hierarchically selects PC frames. Further, we propose to utilize the spatio-temporal continuity to guide the sampling procedure. In particular, considering that PC data sequences inherently exhibit spatial and temporal characteristics, we design a reward based on the variance between the newly sampled PC and the predicted object mobility situation (i.e., location and velocity of objects) derived through spatio-temporal analysis. This approach aims to ensure that the sampled PC frames are of maximal importance. The experimental study (detailed in Section 7) shows the superiority of our framework.

We summarize our contribution as follows:

- We develop an efficient query processing framework for retrieval and aggregate queries with semantic and spatial filters on PC data, which relies on deep learning models to extract semantic information. Notably, our work is the first to focus on efficiently processing such queries on PC data.
- We design a novel query processing method named Multi-Agent framework with Spatio-Temporal Analysis (MAST). This method leverages the spatio-temporal feature of PC data to improve the query performance on sampled PC frames. We theoretically analyze the approximate performance bound of our framework.
- We conduct extensive experiments on real and synthetic datasets. The experimental results show the superiority of our framework and the proposed method MAST compared with the baselines.

The rest of this paper is organized as follows: In Section 2, we define the problem of PC query processing. In Section 3, we summarize the related work. In Section 4, we introduce the framework of efficient PC query processing. We then introduce our proposed processing method MAST (Section 5) and conduct theoretical analysis (Section 6). Finally, we present and analyze the experimental results in Section 7.

2 Preliminaries

We introduce the type of queries that we focus on in this paper, and introduce the formalized research problem that we study.

2.1 Query Definition

We first introduce two types of queries, namely *PC retrieval query* and *PC aggregate query*.

PC Retrieval Query: We define the PC retrieval query as the query that returns *id*s of PC frames satisfying all query predicates.

PC Aggregate Query: The PC aggregate query returns the aggregate numerical result n aggregated by the extracted information from $\mathcal{D}$. In this work, we evaluate different aggregate operators. `Avg` returns the average number of filtered objects in each PC frame, `Count` returns the total number of satisfied PC frames, `Med` returns the median number of filtered objects of the PC frames, and `Min` and `Max` return the global minimum and maximum numbers of filtered objects, respectively. Other aggregate predicates can be supported with minimal effort by adding new operators.

For the query predicates, we focus on two types of filters that are of high importance during PC query processing, namely semantic predicate and spatial predicate. Our proposed framework is also applicable to other predicates by adding corresponding functions.

Semantic Predicate: We define the semantic filter as the numerical filter of objects: $|\texttt{Obj}|\ [\leq, \geq]\ \texttt{num}$. It filters on the number of satisfied objects in a PC frame, in which the PC frame can be regarded as a table with the object as the column. In the example given in Section 1, the semantic predicate filters on the result of the subquery in green brackets.
Spatial Predicate: The spatial filter specifies the spatial requirement of the considered objects. In this work, we consider the distance from the sensor (vehicle generating the PC) to the objects: $\texttt{Distance}(\texttt{Obj}, \texttt{center})\ [\leq, \geq]\ \texttt{r}$. Other spatial filters can be also supported by adding spatial operators.

2.2 Problem Definition

In this paper, we focus on optimizing the query processing on PC data which are captured periodically in batches within a database, i.e., a database of PC frames $\mathcal{D}$. Each PC frame $\texttt{P}$ contains spatial and temporal features: $\texttt{P} = (\mathcal{P}, t) \in \mathcal{D}$, in which $\mathcal{P}$ is a set of 3-dimensional spatial points and t is the timestamp indicating the generation time of $\texttt{P}$. The PC frames $\texttt{P}$ in $\mathcal{D}$ from the same LiDAR sensor can be constructed as a sequence of frames w.r.t. the timestamp t. A deep model $\texttt{M}$ is included which takes $\texttt{P}$ as the input and outputs the semantic information. In this paper, $\texttt{M}$ is the object detection model which predicts the objects in the PC frame and their corresponding locations, represented as the bounding boxes of objects: $\texttt{M}(\texttt{P}) : \mathcal{B}$, $b = (min, max, angle) \in \mathcal{B}$, where *min* and *max* are the minimum and maximum points of the bounding box, and *angle* is the rotation angle of the object. The objective of the research is to efficiently process queries and generate query results with maximized accuracy (e.g., F1 score measuring retrieval query, and aggregate accuracy measuring aggregate query) under a limited budget of $\texttt{M}$ usage. We define the problem as follows.

Problem 1. *Given a database $\mathcal{D}$, where PC data periodically arrive at the server, we aim to design a framework for processing PC data such that the query accuracy of queries is maximized, while the average query latency with a limited budget of resource usage, i.e., the usage budget of deep model* $\texttt{M}$*, is minimized.*

Remark. (1) We develop the framework to sample the PC data for all types of queries we consider, instead of optimizing a specific query. (2) Our PC query processing framework is dedicated to the situation where deep model computation is costly (since preprocessing all PC frames is not practical compared with the size of the requested queries), or limited GPU resources are available [21]. (3) Our framework is designed to meet the demands of analyzing periodically collected data on the server side, rather than the real-time decision making support on the vehicle side. (4) We design a general framework that would be suitable for any database system for PC data on the server side.

3 Related Work

Point cloud data management & analytics. There are industry practices [34, 46] supporting PC data, which treat the 3D points as basic spatial objects. Currently, there is no system supporting processing specialized queries on PC data. As for PC data analytics, three main tasks are exclusively studied [12], namely 3D classification [35, 41], object detection [39, 40, 48], and semantic segmentation [16, 36]. Among them, object detection is important in the autonomous driving scenario to analyze auto cars' driving actions. In particular, object detection takes point cloud frames as input and detects objects appearing in the PC frame, returning their corresponding labels (the category of the object) and the locations (normally a bounding box that contains the object).

The SOTA detection methods [39, 40, 48], e.g., PV-RCNN [39], have achieved above 86% accuracy of vehicle location prediction (the correctness of predicted bounding boxes). However, these studies focus on improving the model accuracy, and therefore have a low inference efficiency (which takes around 0.1 seconds to process one PC frame using the current GPU). Therefore, conducting

analytical queries on PC data can be inefficient. Different from current point cloud analysis studies, in this paper, we aim to provide a framework that can efficiently process point cloud data analysis with a budget of GPU resource usage.

Video data query processing. Our work is inspired by work on video data analysis [3, 7–10, 13, 17–19, 21, 25, 26, 42, 47, 50]. This line of research can be summarized into two categories [9]: (1) accelerating the query by providing approximate results through deep model inference efficiency optimization [3, 19, 20], and (2) treating the deep model as a preprocessing method, and analyzing the information extracted by deep models [26, 50], without the need to refer to the original data. In addition, there is some work [7, 8] that aims to enhance the video analytics by optimizing the query planner. In this paper, our focus on PC data analysis is most relevant to the first category of research, where the data is of significant volume and new data may come, making it impractical to preprocess all data prior to analysis.

The work in the first category can be further classified into two subcategories. The first category aims to generate lightweight models that provide faster analysis but coarser predictive performance [1, 19–21, 37]. For example, Kang et al. [19, 20, 21] propose a series of approaches that apply proxy models as efficient approximations to the oracle model to provide efficient query processing on batch and streaming video data. ABae [22] is designed to handle queries with expensive predicates. Anderson et al. [1] develop proxy models by optimizing data representation (e.g., resolution) to increase query efficiency. Russo et al. [37] accelerate the aggregate query on streaming data. The second category is to provide approximate results by reducing the deep model usage [3, 15, 23, 24]. For example, TASTI [23, 24] develops an index producing approximate results based on the fact that data with similar embeddings produce similar query results. Bang et al. [3] propose to sample a portion of the video database and efficiently provide approximate query results. He et al. [15] proposes to leverage commonsense knowledge models that estimate objects' conditional existence probability and choose a fraction of frames to be processed. Moll et al. [33] develop methods to adapt sampling budget regarding data and query.

There is also work on video analytics that analyzes spatio-temporal features of objects with video data [4, 9, 44]. Chen et al. [9] propose to conduct a defined query named STAR retrieval which retrieves objects' co-occurrence duration with spatio-temporal constraint. Different from our setting, STAR retrieval treats applying deep models as a preprocessing step. MIRIS [4] supports processing object tracking queries with constraints such as the velocity of objects on video data. Unlike our method, which supports optimizing query performance for multiple query types instead of a specific given query, MIRIS focuses on the retrieval query and is a query-driven method with query workload as the input. SketchQL [44] provides a framework where users can compose visual queries by drawing the trajectories of objects through the interface. Similar to [9], SketchQL focuses on a specific query with spatio-temporal features instead of supporting general retrieval and aggregate queries.

PC data contains exact 3D spatial information. In this paper, we present the first study to offer efficient query processing for Point Cloud data. We design a novel sampling mechanism and querying method that produces efficient and accurate query results.

There is some work [3, 4, 10, 33] that samples to improve query efficiency for video data. Zeus [10] assigns a higher sampling rate to the video segments containing actions while assigning a lower sampling rate to those without, to accelerate action retrieval queries. Zeus decides a sampling rate for each video segment, while our method provides frame-level sampling for the whole PC sequence. Seiden [3] optimizes queries on video data by selecting sampled video frames with temporal continuity leveraged. ExSample [33] applies sampling to help detect video segments likely to contain objects of interest. MIRIS [4] applies a filter and refined procedure to control the processed frame rate among video segments w.r.t. the given query. [3, 4, 33] are query-driven

methods, that optimize query performance given a specific query or a workload of queries, while our sampling method focuses on features of the input data and optimizes all types of queries considered without requiring a specific query workload as the input, which is similar to [10]. It allows the sampling procedure to be executed before the query workload is known, and enables different queries to be conducted with the same sampling results.

Reinforcement Learning for Database Management. Reinforcement learning (RL) methods are widely utilized in optimizing data management and analysis [6]. Marcus et al. [32] utilize the RL method to tune the knobs of a database system. Li et al. [28] and Gu et al. [11] apply the RL method to help construct the index structure. Bang et al. [3] apply the UCB [2] method to handle data sampling. Liang et al. [30] utilize RL for data partitioning. He et al. [14] apply a hierarchical bandit structure [49] to detect data clusters with most frames satisfying the UDF filter. [14] provides a static hierarchical MAB framework with all agents initialized before training, while our method generates new MAB agents during the sampling procedure. In this paper, we develop a multi-agent framework to help conduct the PC data sampling procedure. We carefully design the reward that captures the spatio-temporal difference between the system's predicted result and the sample result.

4 Framework Overview

We briefly introduce the key challenges of efficiently querying PC data and how our proposed solution, namely Multi-Agent framework with Spatio-Temporal analysis (MAST), addresses the challenges. The architecture of MAST is shown in Fig. 2.

Importance sampling: To conduct efficient queries, MAST selects a fraction of data points out of $\mathcal{D}$ to produce approximate query results. This raises a vital problem for MAST to detect important PC frames, i.e., which PC frames mostly affect the query performance. Addressing this problem requires that the sampling method analyze the relation between the features of PC frames and the query performance. To resolve this, we design the *Sampling Module* which provides the first sampling method that analyzes the spatio-temporal features of PC frames to help select PC frames. We develop a multi-agent sampling policy, and carefully design a reward. This reward reflects the difference between the real mobility information extracted from the sampled PC frame by the deep model and the estimated mobility information predicted by the mobility estimating module, named *ST-PC Analysis Module*. Therefore, MAST selects PC frames whose mobility situations differ the most from the estimated situations predicted by the currently sampled PC frames. This approach results in more accurate query results.

Accurate and efficient query processing: With sampled PC frames and the information extracted by the deep model, generating accurate query results becomes a vital issue. Current methods, such as linear prediction, do not consider the mobility situation of PC frames, and thus result in sub-optimal query performance on queries such as retrieval queries (as analyzed in Section 7). To address this issue, we propose predicting query results utilizing the spatio-temporal (ST) continuity of PC frames. Our method simulates the real object movement, and achieves outstanding performance on multiple query types. Furthermore, processing queries with ST-analysis will take more computation. To accelerate the query processing, we precompute and store the mobility prediction in the *Indexing module*, thus preventing repetitive computation and achieving computation cost reduction.

Pipeline. When a query arrives, (1) the sampling module selects the sampled PC frames from the database, and the sampled frames are input to the deep model to get the predicted results. The feedback (reward in the reinforcement learning scenario) is then provided by the deep model to the sampling module for policy fine-tuning; (2) The sampled frames together with the deep model results are stored in an index for query processing. The ST-PC analysis module provides mobility

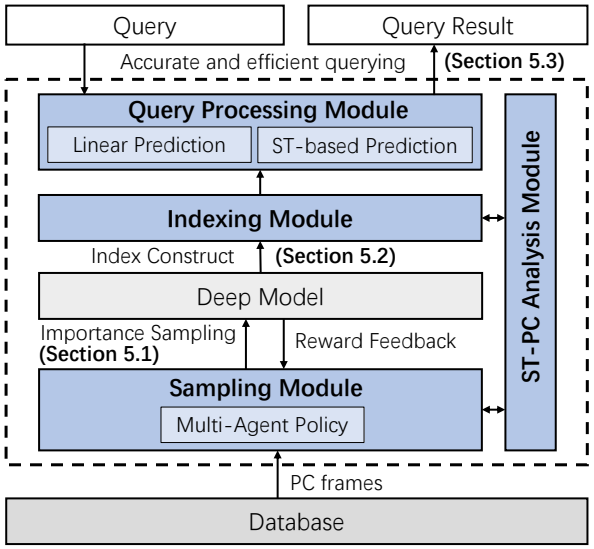


Fig. 2. The architecture of our PC query processing framework, including sampling module, deep model, indexing module, ST-PC analysis module, and query processing module. The framework is general for all PC database systems.

prediction for sampling as well as the indexing module; (3) After the sampling procedure is done, the query is then input to the query processing module. The processing module will generate the approximate query result.

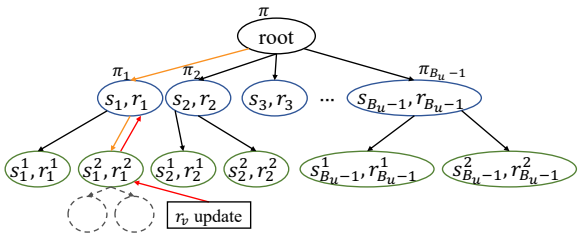
5 MAST: Efficient PC Query Processing

We proceed to introduce the detailed design of the modules of our framework Multi-Agent framework with Spatio-Temporal analysis (MAST). In particular, we design an effective multi-agent sampling method in the *Sampling Module*, and propose a spatio-temporal analysis-based method in the query processing module.

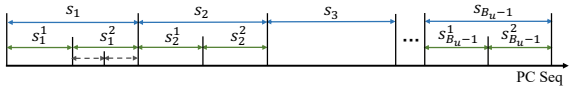
5.1 Multi-Agent Sampling

Motivation. In the sampling procedure, a simple way of sampling PC frames is to sample the PC frames randomly from $\mathcal{D}$. However, as described in recent research [3, 21], the importance of different data points varies in terms of contribution to sampling performance. Therefore, to achieve good query performance, an optimal sampling method that can sample frames with higher importance is necessary. Suppose the metric measuring the importance of PC frames is defined as their contributions to query accuracy, a sampling policy that can automatically explore the PC frames with potential high importance w.r.t. the metric is preferred. Therefore, we propose to utilize RL techniques to develop the sampling policy, which can detect important PC frames, without heavy heuristic designs, and balance the exploitation and exploration. Furthermore, the policy should be lightweight (without a deep model) to achieve high efficiency during sampling.

We proceed to introduce the sampling procedure. First, we explain how we develop the RL environment by modeling the problem as a decision-making process. Then, we describe reward computation based on analyzing the spatio-temporal features of PC frames. Finally, we introduce the overall sampling algorithm.



(a) The corresponding segment tree.



(b) The partitioned PC frame sequence.

Fig. 3. The segment tree for PC frame sampling.

Hierarchical Multi-Arm Bandit Problem Modeling. To utilize RL methods, we should first initialize the environment for the RL policy. We model the decision-making process as a multi-arm bandit problem [3], where the PC sequence is split into multiple segments (as the multiple arms) and the agent will select one segment from which a PC frame will be sampled.

1. Uniform sampling and coarse segmentation: Fig. 3b denotes the PC sequence. As the sampling procedure begin, we first uniformly sample the PC frames with a budget of $B_u = \beta \cdot B$ with $\beta \in (0, 1)$. Suppose the PC sequence contains $|\mathcal{D}|$ frames, $\mathcal{S}_u = \{\mathsf{P}_0, \mathsf{P}_{\frac{|\mathcal{D}|}{B_u}}, \ldots, \mathsf{P}_{|\mathcal{D}|}\}$ are sampled with an equal interval $|\mathcal{D}|/B_u$. The PC frame sequence is partitioned w.r.t. $\mathcal{S}_u$ into $B_u - 1$ segments with equal length, and we denote these segments with $Seg = \{s_i | i \in [1, B_u - 1]\}$ (segments colored in blue in Fig. 3b). The empirical study reveals that a lower frame density requires a higher B_u. In particular, each segment s_i contains two features: $s_i.start$ and $s_i.end$, which respectively indicate the id of the leftmost sampled PC frame and the rightmost sampled PC frame. Then, PC frames in $\mathcal{S}_u$ are input to `M` and processed. It provides the mobility situation of the beginning and end of each segment, as the initialized information. After Seg is initialized, an agent π is assigned which selects a segment each time from Seg, where the new sampled PC frame comes from.

2. Hierarchical fine-grained segmentation: After one segment s_i from Seg is selected by π, one simple way to get the new sampled PC frame is to randomly select one from s_i [3]. However, random sampling fails to explore important PC frames in s_i, and it will also provide unstable rewards, resulting in sub-optimal performance. To solve this issue, we propose further splitting s_i into shorter subsegments and assigning another agent to s_i to select PC frames. This forms a branching decision-making process [30], where the policy recursively selects segments from a coarse to a fine-grained degree.

Specifically, if a segment s_1 is selected and it has never been selected by π before, then the middle PC frame of s_1 with the ID of $id = \lfloor (s_1.start + s_1.end)/2 \rfloor$ will be selected and returned as the sampled PC. Then, s_1 is split into two subsegments by P_{id}: $s_1^1 = \langle s_1.start, id \rangle$ and $s_1^2 = \langle id, s_1.end \rangle$ (segments colored in green in Fig. 3b). We select the middle PC frame since currently there is no information about which PC frame within s_1 is the most important, and the lengths of the new split segments s_1^1 and s_1^2 are balanced. We then assign a new agent π_1 to s_1. When s_1 is re-selected by π in a future sampling step, π_1 is responsible for selecting between s_1^1 and s_1^2 (forming a 2-armed bandit problem), from which the new sampled PC frame is sampled. Therefore, we develop a multi-agent

sampling mechanism that hierarchically selects segments from coarse to fine-grained degree with multiple agents.

Note that each subsegment is divided into two in each branching step. The binary splitting allows for flexible depth control, where more frequently chosen subtrees will have a deeper level and more fine-grained segmentation, while less frequently chosen (i.e., less important) subtrees will have a shallower level. We empirically evaluate the effect of varying branching factors in Section 7.

3. Segment tree modeling: To model the hierarchical sampling procedure described above, we introduce a tree structure named *segment tree*. As illustrated in Fig. 3a, the segment tree is initialized with $B_u - 1$ segments (*Seg*) as the child nodes (representing the segments) of a root node (representing the whole sequence). Each child node stores (s_i, r_i) where s_i denotes the corresponding segment and r_i denotes the expected reward when selecting segment s_i. Consequently, each new sampling operation can be modeled as recursively selecting from the root node to a leaf node of the segment tree. Following this, the selected leaf node is then split into two new initialized child nodes.

As shown in Fig. 3a, during the sampling step, when a child node s_i is selected, two conditions are considered: (1) If the node s_i has never been selected before and is thus a leaf node, it returns the middle PC frame ID as the sampling result. The node then splits the corresponding segment into two sub-segments as described earlier, generating two child leaf nodes (s_i^1, r_i^1) and (s_i^2, r_i^2) with an initial reward of 0. An agent π_i is assigned to s_i to help select between (s_i^1, r_i^1) and (s_i^2, r_i^2) in future sampling steps. (2) If the node has been selected before (e.g., node s_1) and is a non-leaf node, it already has child nodes. Our algorithm then recursively applies its corresponding agent (e.g., π_1), selecting one of the two child nodes. The sampling step concludes when a leaf node of the tree is reached, and the frame in the middle of the corresponding subsequence will be selected as the sampled PC frame. The generated reward will then recursively update the agents on the path from the leaf node to the root node. We provide an example of one sampling step:

Example 5.1. The policy first selects a leaf node (s_1^2, r_1^2) (by following the orange path in Fig. 3a), and then a new reward r_v is computed. The expected rewards (r_1^2 and r_1) of nodes s_1^2 and s_1 are then recursively updated (following the red path in Fig. 3a) based on r_v w.r.t. Eq. 2. Two child nodes (in gray and dashed) are then initialized.

After $B - B_u$ sampling iterations, the segment tree keeps splitting and will have in total $(B_u - 1) + (B - B_u) = B - 1$ leaf nodes, each corresponding to a fine-grained segment. Each sampling step will traverse from the root node of the segment tree to a leaf node. To maintain efficiency, the tree is assigned a maximum depth, beyond which nodes will cease to produce deeper child nodes and will instead perform random sampling from the corresponding PC sequence segment. The default maximum depth is set to 10 (which is sufficient to produce enough subsegments).

4. RL agent design: The selection of a leaf node corresponds to a sequence of branch selections in the segment tree which requires an agent for each non-leaf node. To balance exploration and exploitation, we apply the Upper Confidence Bound (UCB) algorithm [2] as the algorithm of the agent to each non-leaf node. The UCB algorithm is a lightweight method that ensures high sampling efficiency without requiring a deep model. At each step, UCB selects an arm k with the maximum value v_k, computed as: $v_k = r_k + c \cdot \sqrt{\frac{2 \ln N}{N_k}}$, where r_k is the expected reward for selecting arm k. The parameter c balances the weight between exploration (selecting the less chosen arm) and exploitation (selecting the arm with the highest expected reward), and is set to 2 by default. A larger value of c increases the probability of selecting a less chosen arm. N_k and N represent the sampling frequency of arm k and the total sampling frequency, respectively.

Reward Design. To optimize the policy, the reward r_v generated w.r.t. the newly sampled PC frame P should reflect the importance of P in terms of contribution to query performance. Therefore, we

design a reward by estimating the mobility difference between actual mobility (extracted by M) and the estimated mobility (predicted by MAST). The rationale of the idea is that PC frames can be estimated accurately by MAST based on the currently sampled PC frames contribute less than the PC frames that have a very different mobility compared with the estimated ones. Therefore, we first describe the spatio-temporal point cloud (ST-PC) analysis method, which analyzes the mobility of objects in the PC frames, and then we present how to use mobility to design the reward.

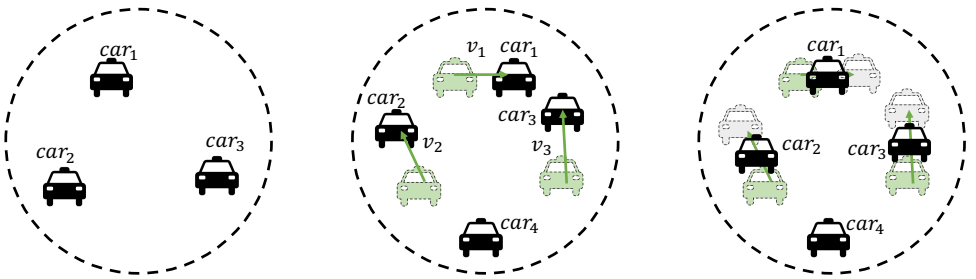
1. Spatio-Temporal PC-Analysis: Suppose two frames $\mathsf{P}_{t_1}, \mathsf{P}_{t_2}$ are sampled with a time interval $t = t_2 - t_1$, and the objects in P_{t_1} and P_{t_2} are predicted by the deep model, represented by the bounding boxes $\mathcal{B}_{t_1}$ and $\mathcal{B}_{t_2}$. W.l.o.g., we assume the objects are moving in a constant velocity from P_{t_1} to P_{t_2}, and we would like to analyze how the objects are acting, i.e., the speed and direction of the corresponding object. Therefore, we design an algorithm that tracks the objects between two frames and generates the simulated velocities that allow objects to move from the situation of P_{t_1} to P_{t_2}, namely spatio-temporal point cloud (ST-PC) analysis. To be specific, we first leverage the Hungarian algorithm [27] to do the pairwise matching between $\mathcal{B}_{t_1}$ and $\mathcal{B}_{t_2}$. Then, we treat the matched bounding boxes as the bounding boxes of identical objects and compute the velocity by dividing the time interval t by the distance between the two matched bounding boxes. Then, for the unmatched boxes, we develop a mechanism to control their appearance by adjusting the confidence of the boxes. We provide the pseudo-code of the spatio-temporal PC analysis algorithm in Alg. 1.

Algorithm 1: Spatio-temporal PC Object Analysis

input : box predictions $\mathcal{B}_{t_1}, \mathcal{B}_{t_2}$ of frame 1 P_{t_1} and frame 2 P_{t_2} (two continued sampled frames with timestamp t_1, t_2, respectively)
output : Velocity $\mathcal{V}$ of P_{t_1}

Initialize M;
$\mathcal{V}$[velocity], $\mathcal{V}$[additional box] $\leftarrow \emptyset$;
for $b_i \in \mathcal{B}_{t_1}$ **do**
  for $b_j \in \mathcal{B}_{t_2}$ **do**
    /* compute the distance between two boxes */
    $M_{i,j} \leftarrow ||b_i.center - b_j.center||_2$;
/* Hungarian matching generate matching pairs */
Match = Hungarian_Match(M);
for $i, j \in$ Match **do**
  $v_i = M_{i,j}/(t_2 - t_1)$;
  $\mathcal{V}$[velocity].$append(v_i)$;
if $|\mathcal{B}_{t_1}| > |\mathcal{B}_{t_2}|$ **then**
  for $b_i \in unmatch(\mathcal{B}_{t_1})$ **do**
    $v_i \leftarrow 0$;
    $\mathcal{V}$[velocity].$append(v_i)$;
else if $|\mathcal{B}_{t_1}| < |\mathcal{B}_{t_2}|$ **then**
  for $b_j \in unmatch(\mathcal{B}_{t_2})$ **do**
    $v_j \leftarrow 0$;
    $\mathcal{V}$[additional box].$append(b_j)$;
    $\mathcal{V}$[velocity].$append(v_j)$;
return $\mathcal{V}$

In Alg. 1, we first initialize the list to store the velocity of objects in P_{t_1} (line 2). Then, we compute and store the pairwise distance between objects in P_{t_1} and P_{t_2} (lines 3–5). We match objects between two frames with the Hungarian Match algorithm (line 6), and regard the matched bounding boxes as the boxes of identical objects (we only match objects with the same category). We then compute the velocity of the object (lines 7–9). In the case that the number of objects is not equal between two PC frames, there are two conditions: if (1) $|\mathcal{B}_{t_1}| > |\mathcal{B}_{t_2}|$ (lines 10 − 13), we set the velocity of the unmatched $b_i \in |\mathcal{B}_{t_1}|$ as 0, and decrease the confidence score with time passes (i.e., it will not be considered when the confidence bellows the confidence threshold, during the bounding box prediction of unsampled PC frames); else if (2) $|\mathcal{B}_{t_1}| < |\mathcal{B}_{t_2}|$ (lines 14–18), we also set the velocity of unmatched boxes (termed additional boxes) as 0 and increase the confidence score with time passes.



(1) PC frame in t_1 (2) PC frame in t_2 (3) Predicted PC frame in $t, t \in (t_1, t_2)$

Fig. 4. An example of how spatio-temporal PC analysis is processed. The black cars are the actual objects in the corresponding PC frame. In (3), the green and gray cars are the cars that appeared in P_{t_1} and P_{t_2}, respectively. We compute the estimated locations of objects based on P_{t_1} and P_{t_2}.

Example 5.2. This example illustrates how to generate the estimated mobility of unsampled PC frames based on ST-PC analysis, as shown in Fig. 4. In Fig. 4(1), three cars are detected in the PC frame P_{t_1} generated in t_1, namely car_1, car_2, and car_3. In Fig. 4(2), four cars are detected in the PC frame P_{t_2} generated in t_2. After the Hungarian algorithm is conducted, car_1, car_2, and car_3 in P_{t_1} are paired to a car in P_{t_2}. The algorithm will regard the paired cars as the same car. The velocity is then computed, noted as v_1, v_2, and v_3 of car_1, car_2, and car_3. One object car_4 is not matched in P_{t_2} . Given the PC frame in t, $t \in (t_1, t_2)$, we generate the predicted object spatial situation: For the matched cars, we update the location by plus the velocity times the time gap: $Loc(car_i, t) = Loc(car_i, t_1) + v_i \cdot (t - t_1)$. Then for the unmatched car_4, we decide whether the confidence that car_4 is in P_t is high enough (above 0.5 by default). Note that the confidence increases if t is close to t_2. If the unmatched car is from P_{t_1}, the confidence increases if t is close to t_1.

2. Generating Reward w.r.t. ST-PC analysis: With Alg. 1, we proceed to describe how the reward r_v is generated. When a PC frame P_t is selected as the sampled PC by the policy, the deep model will predict the bounding boxes of objects in P_t, and the output is represented as $\mathcal{B}_t$. Note r_v describes the difference between the object detection result by the deep model and the objects predicted by the ST-PC analysis. Suppose that P_t is between two sampled frames P_{t_1} and P_{t_2}. By applying the ST-PC analysis, we predict the objects and their bounding boxes by inputting P_{t_1} and P_{t_2} into Alg. 1, and get the velocity $\mathcal{V}$ of objects in P_{t_1}. We then apply $\mathcal{V}$ to P_{t_1} and move the objects in P_{t_1} with time $t - t_1$ and get the object boxes set $\mathcal{B}_t^e$ as the estimated bounding boxes (as Example 5.2 shows). We generate r_v by evaluating the difference between $\mathcal{B}_t^e$ and $\mathcal{B}_t$.

To compute r_v, we first input $\mathcal{B}_t^e$ and $\mathcal{B}_t$ into the Hungarian matching algorithm to pair the boxes, resulting in the matched set M_{Hun}. We then compute the distance between the paired boxes. The reward r_v is computed as follows:

$$r_v = (1 - c_{var}) \cdot \frac{\sum_{b_i, b_j \in M_{Hun}} dist(b_i, b_j)}{d_{max} \cdot |M_{Hun}|} + c_{var} \cdot \Big(|\mathcal{B}_t^e| + |\mathcal{B}_t| - 2 \cdot |M_{Hun}| \Big), \tag{1}$$

where b_i and b_j are the matched box pairs from $\mathcal{B}_t^e$ and $\mathcal{B}_t$, $dist(b_i, b_j)$ is the Euclidean distance between b_i and b_j, $|M_{Hun}|$ is the number of matched pairs, and d_{max} is the maximum distance that a LiDAR sensor could capture. Additionally, for the unmatched bounding boxes, we incorporate another component of the reward based on the number of mismatched pairs. A weight c_{var} is applied to re-weight the two components of the reward.

Algorithm 2: Hierarchical Multi-Agent Sampling

input : A database of PC frames $\mathcal{D}$, an oracle deep model M, sampleing budget B, the ratios α and β
output : The set of sampled PC frames $\mathcal{D}_s$

1 $\mathcal{D}_s \leftarrow \emptyset$;
2 $B_u \leftarrow \beta \cdot B$;
3 $\mathcal{D}_s.append(\texttt{UniformSample}(\mathcal{D}, B_u))$;
4 $\texttt{Sort}(\mathcal{D}_s)$;
5 **for** i *from* 0 *to* $(len(\mathcal{D}_s) - 1)$ **do**
6 $\quad s_i \leftarrow (\mathcal{D}_s[i],\ \mathcal{D}_s[i+1])$;
7 $\quad Seg.append(s_i)$;
8 Initialize T with Seg;
9 $B_r \leftarrow (1 - \beta) \cdot B$;
10 **while** $B_r > 0$ **do**
11 $\quad s \leftarrow$ sampling a leaf node of T with its current policy ;
12 $\quad \texttt{P} \leftarrow \mathcal{D}[(s.start + s.end)/2]$;
13 $\quad \mathcal{D}_s.append(\texttt{P})$;
14 $\quad r_v = Reward(\texttt{P}, \texttt{T})$ using Eq.1;
15 $\quad$ split the leaf node s in T into two nodes s^1, s^2 with P;
16 $\quad \texttt{T}.recursive\_update(\alpha, r_v, s)$;
17 $\quad B_r \leftarrow B_r - 1$;
18 **return** $\mathcal{D}_s$;

Hierarchical Multi-Agent Sampling. The whole sampling procedure is summarized in Alg. 2. The sampling set $\mathcal{D}_s$ is initialized as $\emptyset$ (line 1). Uniform sampling is then conducted with a budget of B_u (lines 2–3). Then the segment set Seg is set up, and the segment tree T is initialized (lines 4–9). We then process the sampling procedure by recursively sampling PC frames w.r.t. the policy modeled by T (lines 10–17), including the sampling (lines 11–13), the reward computing (line 14), and the segment tree update (lines 15–16). Specifically, to account for incremental changes, we update the expected reward at each non-leaf node of the sampling path provided in Example 5.1 at time t as follows:

$$r_t = (1 - \alpha_r) \cdot r_{t-1} + \alpha_r \cdot r_v, \tag{2}$$

Algorithm 3: Indexing construction

```
   input  : Sampled PC list S
   output : Constructed Index
1  Initialize Index with S;
2  for i from 0 to Len(S) − 1 do
3      t_i, t_{i+1} = S[i].t, S[i + 1].t;
4      for t from t_i + 1 to t_{i+1} − 1 do
5          B_t^e ← Pred(S[i], S[i].V, t − t_i);
6          Index.add(B_t^e);
7  return Index;
```

where r_t is computed by re-balancing the historical reward r_{t-1} and the newly generated reward r_v with the ratio α_r.

5.2 Indexing

To reduce repeated computation, after the sampling procedure, the predicted results of the deep model are stored in the index module. Further, the ST-PC analysis module is applied and predicts the estimated object mobility situation for all unsampled PC frames. The estimated results are also stored in the index module for the predicting module computation.

As shown in Alg. 3, the indexing procedure predicts the bounding boxes of the un-sampled PC frames (lines 1–6). The bounding box prediction function Pred() in line 5 takes the i-th sampled PC frame $\mathcal{S}[i]$ and its corresponding velocity $\mathcal{V}$, and the time gap between the predicted PC frame P_t and $\mathcal{S}[i]$. As described in Example 5.2, the function will apply the velocity and update the location of the matched bounding boxes in $\mathcal{S}[i]$. For the unmatched objects, the confidence score of unmatched boxes in $\mathcal{S}[i]$ will drop with t increases, while the confidence score of unmatched boxes in $\mathcal{S}[i+1]$ will increase with t increases. The predicted bounding box $\mathcal{B}_t^e$ is then stored in *Index* (line 6).

5.3 Query Processing

In the query processing module, the framework leverages the sampling results stored in *Index*, and produces approximated query results for queries. Similar to the sampling procedure, we apply the proxy results of ST-PC analysis to help conduct the PC queries, including the retrieval and aggregate queries. With the sampled PC and their corresponding bounding boxes prediction, we provide multiple methods to conduct approximate queries, namely linear prediction [3], and ST-based prediction.

For the linear prediction, if the numerical result computed by the filters of a query q of two sampled PCs P_{t_1} and P_{t_1} are n_{t_1} and n_{t_2}, the linear prediction will predict the numerical result of P_t with $t \in [t_1, t_2]$ as $n_t = n_{t_1} + (n_{t_2} - n_{t_1}) \frac{t-t_1}{t_2-t_1}$.

Example 5.3. If a retrieval query requests to return all frames with more than 3 cars within 10 meters of the sensor. Suppose P_1 at time 1 has 1 satisfied car, while P_4 at time 4 has 5 cars satisfying the predicates. The linear prediction will predict that P_2 with number $\lfloor 1 + (5-1)\frac{2-1}{4-1} \rfloor = 2$, which does not meet the query predicate, therefore P_2 will not be put in the result PC set.

Linear prediction only computes the numerical shifting and does not consider the real mobility situation of objects in the PC data. Therefore, we leverage the predicted bounding boxes by ST-analysis in *Index* to help produce query results. To be specific, with the query q given, as described in Example 5.3, the framework computes a numerical number n which indicates the satisfied

number of objects, based on the predicates of q and the bounding box set $\mathcal{B}$ of a PC frame. Our framework processes the approximate retrieval query and aggregate query results as follows: For the retrieval query, the framework determines whether the numerical number satisfies the frame of the predicate (in this case, it determines whether n satisfies the constraint).

$$True \ / \ False \leftarrow Q_R(\mathcal{B}), \tag{3}$$

where Q_R is the retrieval process function.

For the aggregate query, the framework summarizes the numerical results and produces the query result. E.g., if the operator is average operator Avg, the framework will compute all numerical numbers of all frames and do the average computation: $\frac{\sum n_i}{|\mathcal{D}|}, \mathsf{P}_i \in \mathcal{D}$.

$$N \leftarrow Q_A(\mathcal{B}), \tag{4}$$

where Q_A is the aggregate process function. Different from the linear prediction which linearly predicts the numerical result for the unsampled PC, the ST prediction will apply the predicted bounding boxes $\mathcal{B}_t^e$ of PC frame by ST-PC analysis at time t, and compute the numerical number like the sampled PC frames by inputting $\mathcal{B}_t^e$ to Eq. 3 and 4. The index structure stores $\mathcal{B}_t^e$ in advance so there will be no repetitive computation.

6 Theoretical Analysis

In this section, we analyze the time complexity of our algorithm and bound the query result errors in query processing.

6.1 Complexity Analysis

We evaluate the time complexity of the sampling procedure, the indexing procedure, and the query procedure.

For the sampling procedure, the framework will select B PC frames with a budget of B. The deep model processes the sampled frames with a complexity of $O(B \cdot T_{\mathsf{M}})$ where T_{M} indicates the time used to process one PC frame with deep model M. The policy sampling time can be bounded by the depth of the segment tree $Dep_{\mathsf{T}} = \log(B)$, so we have the overall sampling complexity $O(B \cdot Dep_{\mathsf{T}}) = O(B \cdot \log(B))$. Since T_{M} is much larger than $\log(B)$, we have that the total time complexity is $O((B \cdot T_{\mathsf{M}}) + B \cdot \log(B)) = O(B \cdot T_{\mathsf{M}})$. In other words, the time complexity of the sampling procedure is proportional to the sampling budget.

The indexing procedure utilizes the deep model's output to predict approximate results for unsampled PC frames. Initially, the number of predictions is $|\mathcal{D}|$ since each frame in the database is analyzed to conduct approximate query processing. Assuming each prediction has a complexity of P, the overall complexity is $O(P \cdot |\mathcal{D}|)$.

In the query procedure, the complexity is proportional to the size of the given workload. Assuming the workload size is $|\mathcal{Q}|$ and each query's latency from the indexing is C, the overall complexity would be $O(C \cdot |\mathcal{Q}|)$. Linear prediction is faster as it only requires computing and storing linear numerical changes, whereas spatio-temporal (ST) prediction involves analyzing bounding boxes and obtaining results. In our empirical study (Section 7.2), linear prediction takes $0.03s$ to process one query, while ST prediction takes $0.07s$ per query, under default settings.

6.2 Aggregate Error Analysis

We proceed to analyze the error bound of our query processing framework. First, we assume that the sampling policy returns the preferred sampling result (as described in Section 7.2 (RQ7)). The preferred sampling result is the PC frame subset that best reflects the object mobility situation, simulating the ground truth with the sample budget.

Next, we study the aggregate operators Avg, Count, and Med operators. Specifically, we define the functions $f_{\texttt{Avg}}(\mathcal{S})$, $f_{\texttt{Cnt}}(\mathcal{S}, \theta)$, $f_{\texttt{Med}}(\mathcal{S})$ as the functions that take sampled PC frames $\mathcal{S}$ as input and output the average query result, count query result, and median query result, respectively, computed based on the query filters and $\mathcal{S}$. Then our objective in this section to bound the errors $|f_{\texttt{Avg}}(\mathcal{S}) - f_{\texttt{Avg}}(\mathcal{D})|$, $|f_{\texttt{Cnt}}(\mathcal{S}, \theta) - f_{\texttt{Cnt}}(\mathcal{D}, \theta)|$, and $|f_{\texttt{Med}}(\mathcal{S}) - f_{\texttt{Med}}(\mathcal{D})|$.

In what follows, we let $y(t)$ denote the function that gives the number of objects satisfying the query's predicates at time t. To aid our analysis, we further assume that this function $y(t)$ is Lipschitz-continuous (see the technical report [29] for a formal definition). Under this assumption, we provide the following error estimates of our sampling procedure for our operators of interest.

Theorem 6.1. *Suppose that $y(t)$ is Lipschitz-continuous with constant L_y. We have the following upper bounds on the errors.*

$$\left|f_{\texttt{Avg}}(\mathcal{S}) - f_{\texttt{Avg}}(\mathcal{D})\right| \leq L_y A_{\mathcal{S}}, \tag{5}$$

$$|f_{\texttt{Cnt}}(\mathcal{S}, \theta) - f_{\texttt{Cnt}}(\mathcal{D}, \theta)| \leq \frac{L_y - B_{\mathcal{S},y}}{L_y}, \tag{6}$$

$$|f_{\texttt{Med}}(\mathcal{S}) - f_{\texttt{Med}}(\mathcal{D})| \leq L_y C_{\mathcal{S}}, \tag{7}$$

Here, $A_{\mathcal{S}} \triangleq \frac{1}{4|\mathcal{D}|}\sum_{i=0}^{|\mathcal{S}|-2}(t_{i+1} - t_i)^2$, $B_{\mathcal{S},y} \triangleq \min_{0\leq i\leq|\mathcal{S}|-2}\left[|y(t_{i+1}) - y(t_{i+1})/(t_{i+1} - t_i)|\right]$, and $C_{\mathcal{S}} \triangleq \frac{1}{4}\max_{0\leq i\leq|\mathcal{S}|-2}(t_{i+1} - t_i)$.

The detailed proof of the theorem uses the Lipschitz continuity of $y(t)$. The proof is technical and hence is deferred to the technical report [29]. Here, we remark on the quantities of $A_{\mathcal{S}}$ and $C_{\mathcal{S}}$. Observe that $A_{\mathcal{S}}$ and $C_{\mathcal{S}}$ are dependent on the sample set $\mathcal{S}$ and query-independent. From our empirical studies $A_{\mathcal{S}} \approx 0.28|\mathcal{D}|/|\mathcal{S}|$ and $C_{\mathcal{S}} \approx 0.25|\mathcal{D}|/|\mathcal{S}|$. In other words, our error estimates (5) and (7) are proportional to $|\mathcal{D}|/|\mathcal{S}|$. This indicates that the accuracy of Avg and Med will increase when the sampling budget $\frac{|\mathcal{S}|}{|\mathcal{D}|}$ increases. The error bounds are possible to be applied to provide a specific confidence interval if the empirical value of L_y is provided. Then, the numerical bound could be computed based on the sample result and L_y.

7 Evaluation

The experimental study intends to answer the following questions:

- **RQ1**. How effective is our method MAST for both retrieval queries and aggregate queries compared to other methods?
- **RQ2**. How efficient is MAST compared to other methods in the sampling (deep model processing), indexing, and querying?
- **RQ3**. How does MAST perform with varying query selectivity?
- **RQ4**. How scalable is MAST with varying dataset sizes?
- **RQ5**. How does MAST perform with varying sampling budgets?
- **RQ6**. How does the choice of oracle models affect the performance of MAST?
- **RQ7**. How do the design choices of MAST affect the query performance?
- **RQ8.** What sampled PCs are preferred by MAST?

7.1 Experimental Setup

We proceed to introduce the experimental setups. The experimental parameters are given in Tbl. 1, where the sampling budget denotes the percentage of the dataset to be processed by the deep model. We next introduce the datasets, the query generation, the comparison methods, the oracle model variants, and the measuring metrics.

Table 1. Experimental parameters.

Parameters	Value
Dataset	**SemanticKITTI**, ONCE, SynLiDAR
Oracle Models	**PV-RCNN**, PointRCNN, SECOND
Sampling Budget	5%, **10**%, 15%, 20%, 25%

Table 2. Query Template Parameters

Query Parameters	Value
Object Comparison Operator	$\leq, \geq$
Object Num Threshold (#)	1, 3, 5, 7, 9
Spatial Comparison Operator	$\leq, \geq$
Spatial Distance Threshold (m)	2, 5, 10, 15, 20
Aggregate Operator	`Avg`, `Med`, `Count`, `Min`, `Max`

Dataset. Two real and one synthetic datasets are included in our evaluation, namely SemanticKITTI, ONCE, and SynLiDAR:

- **SemanticKITTI** [5]. The SemanticKITTI dataset contains 22 PC frame sequences that are scanned by a moving vehicle on different roads. Each PC sequence contains several thousand of frames with an FPS of 10 (two neighboring PC frames have a time gap of 0.1 seconds). We select the 5 sequences with the longest PC frame size.
- **ONCE** [31]. ONCE is the dataset that contains a total of one million PC frames. The dataset splits the frames into sequences, with an FPS of 2. We select the top-5 longest sequences.
- **SynLiDAR** [45]. It leverages the Unreal Engine 4 platform to synthesize point cloud data. SynLiDAR contains a long sequence with 45,076 PC frames, with an FPS of 10.

The size of each sequence is approximately 8GB in SemanticKITTI and ONCE, while 80GB in SynLiDAR. The FPS difference makes SemanticKITTI and SynLiDAR have a more intense frame situation while ONCE has a more sparse situation. The experimental results also show the performance difference between the datasets.

Query. We follow the query templates described in Tbl. 2 and design the query workload. We vary the parameters of the query templates and generate different retrieval and aggregate queries. The values of the parameters are selected so that the selectivity of generated retrieval queries would generally be uniformly distributed from 0.1% to 100%, which can be applied to evaluate query performance across different selectivities. During the evaluation, we omit the generated retrieval queries with a cardinality of 0. For the aggregate query, we consider five operators: `Avg`, `Med`, `Count`, `Min`, and `Max`.

Comparison Methods. We evaluate four methods. Note that no previous method is developed to address the problem in this work, and we adapt a SOTA method [3] for video processing.

- **Oracle**. This baseline is implemented which inputs all PC frames to the oracle model and generates the ground object prediction result, and the query result is generated correspondingly. We assume the result provided by Oracle is the true result. This method is applied to verify the efficiency of our system.
- **Seiden-PC**. We implement the method proposed by Bang et al. [3] and adapt it to fit PC data, with a limited deep model usage budget. It models the sampling procedure as a multi-arm bandit problem and solves the problem with an RL policy.

- **Seiden-PCST.** We replace the linear prediction method of Seiden-PC with our proposed query prediction method and generate the query results accordingly.
- **MAST.** Our proposed method. In the query procedure, it applies ST-based prediction when conducting retrieval query, `Count`, and `Med` aggregate queries, and applies linear prediction when conducting `Avg` aggregate queries.

Oracle Model Variants. We select three deep models as the oracle deep models in the experiments. We apply the pretrained models provided by OpenPCDet, MMLab [43].

- **PV-RCNN** [39]. It integrates 3D voxel Convolutional Neural Network (CNN) and PointNet-based set abstraction to learn discriminative point cloud features. PV-RCNN is one of the SOTA PC object detection methods.
- **PointRCNN** [40]. It utilizes a two-stage framework to do the object detection: (1) bottom-up 3D proposal generation (2) refining proposals in the canonical coordinates.
- **SECOND** [48]. The model investigates an improved sparse convolution method for Voxel-based 3D convolutional networks.

Metrics. (1) **Aggregate Accuracy.** We measure the accuracy of the query result compared with the ground truth: $Acc = \left|Res_{gt} - \left|Res_{gt} - Res_{pred}\right|\right| / Res_{gt}$. (2) **F1 Score.** To measure the performance of the retrieval query, we follow [3] and utilize F1 score as the metric: $F1 = 2 \cdot (\text{Precision} \cdot \text{Recall}) / (\text{Precision} + \text{Recall})$, where the precision and recall are computed as: $\text{Precision} = \frac{TP}{TP+FP}$ and $\text{Recall} = \frac{TP}{TP+FN}$. TP is the true positive, FP is the false positive, and FN is the false negative. The Aggregate Accuracy and F1 Score are computed by regarding the Oracle method as the ground truth.

Evaluation Platform. The experiments are conducted on an 80-core server with an Intel(R) Xeon(R) Gold 6248 CPU@2.50GHz 64.0GB RAM. One NVIDIA GeForce RTX 2080 Ti Rev. A with a memory of 11,264MB is leveraged for deep model inference. The experiments are conducted within a Docker container. We release our codebase. [1].

7.2 Experimental Results

We proceed to present the experimental results for each research question.

Table 3. Retrieval query performance comparison between MAST and baselines in terms of the averaged F1 Score.

Dataset	|Seq|	Retrieval F1 Score		
		Seiden-PC	Seiden-PCST	MAST
SemanticKITTI	4,541	0.770	0.780	**0.845**
	4,661	0.823	0.844	**0.854**
	4,071	0.789	0.814	**0.844**
	4,981	0.774	0.801	**0.852**
	3,281	0.798	0.811	**0.882**
ONCE	2,741	0.735	0.743	**0.764**
	3,862	0.806	0.810	**0.828**
	2,983	0.774	0.826	**0.841**
	4,638	0.770	**0.781**	0.777
	5,264	0.736	0.753	**0.798**
SynLiDAR	45,076	0.828	0.842	**0.911**

RQ1. Effectiveness Evaluation. In order to evaluate the effectiveness performance of MAST, we evaluate MAST on two real datasets and one synthetic dataset. The performance of different methods, including Seiden-PC, Seiden-PCST, and MAST, is shown in Tbl. 3 (reporting the retrieval

[1] https://github.com/gravesprite/MAST

query performance), Tbl. 4, and Tbl. 5 (reporting the aggregate query performance). F1 Score and Accuracy are computed w.r.t. the result of the Oracle method, defaulting to the PV-RCNN.

The retrieval query performance is reported in Tbl. 3. It shows that MAST outperforms Seiden-PC and Seiden-PCST in all sequences of SemanticKITTI. MAST gains an average increase of 7.68% in the F1 Score compared with Seiden-PC. We also notice that Seiden-PCST consistently outperforms Seiden-PC on the 5 sequences, which means that ST-PC analysis can effectively improve the quality of detecting PC frames satisfying the query predicate, and thus enhance the query performance. As for the ONCE dataset, MAST outperforms other methods in the majority of sequences (4 out of 5). MAST gains an increase of 4.16% in F1 Score compared with Seiden-PC. Apart from the similar conclusions compared with SemanticKITTI, in the ONCE dataset, the performance gain of MAST is relatively subtle compared with SemanticKITTI dataset in some of the sequences (e.g., Seiden-PCST outperforms MAST in sequence 4). This is mainly because the ONCE dataset collects PC frames with a sparser density (2 frames/second), compared with SemanticKITTI (10 frames/second). The spatio-temporal correlation is weak. Under the long sequence with 45,076 frames in SynLiDAR data, MAST outperforms other methods, increasing F1-Score by 10.0% compared with Seiden-PC, and 8.19% compared with Seiden-PCST. This result is consistent with results in SemanticKITTI since SynLiDAR has the same FPS of 10.

In conclusion, ST-PC analysis performs better on retrieval queries under the dataset with a higher frame density. When PC frames are captured frequently, MAST with ST-PC analysis will gain vital performance improvement on retrieval queries.

Table 4. Performance comparison of aggregate queries with three aggregate operators, including `Count`, `Avg`, and `Med`, in terms of average aggregate accuracy. The PC sequences are consistent with retrieval query performance evaluation.

Dataset	|Seq|	Aggregate Accuracy (%) `Count`			Aggregate Accuracy (%) `Avg`			Aggregate Accuracy (%) `Med`		
		Seiden-PC	Seiden-PCST	MAST	Seiden-PC	Seiden-PCST	MAST	Seiden-PC	Seiden-PCST	MAST
SemanticKITTI	4,541	78.764	89.191	**91.661**	95.107	95.858	**99.086**	100.000	88.333	**100.000**
	4,661	83.299	92.010	**92.736**	94.702	92.714	**97.565**	85.000	90.000	**95.000**
	4,071	79.133	90.785	**93.475**	97.644	96.195	**98.092**	85.500	**98.333**	94.667
	4,981	78.310	89.320	**91.666**	96.739	96.323	**98.904**	95.000	95.083	**96.333**
	3,281	77.643	89.781	**95.308**	98.298	96.829	**99.068**	86.571	98.036	**99.167**
ONCE	2,741	79.091	83.653	**83.899**	91.825	90.170	**94.451**	89.167	79.167	**89.167**
	3,862	80.760	87.585	**91.148**	93.273	94.295	**94.806**	85.000	**100.000**	95.000
	2,983	83.835	89.181	**89.368**	97.631	95.285	**97.846**	70.000	100.000	**100.000**
	4,638	80.173	**89.760**	89.375	96.765	95.789	**97.980**	86.667	**96.667**	93.333
	5,264	75.246	81.118	**86.052**	96.033	96.491	**97.363**	90.000	100.000	**100.000**
SynLiDAR	45,076	82.515	89.887	**90.849**	95.533	89.549	**99.480**	**100.000**	98.571	98.571

The aggregate query performance on `Count`, `Avg` and `Med` operators is reported in Tbl. 4. The performance shows that ST-based prediction can strongly increase the performance of `Count` and `Med` operators. Specifically, MAST outperforms Seiden-PC on the `Count` query, achieves an increase of aggregate accuracy up to 22.75% on SemanticKITTI, and achieves an increase up to 14.36% on the ONCE dataset. MAST outperforms Seiden-PC on `Avg` and gains accuracy increases up to 4.18% on SemanticKITTI, and up to 2.85% on ONCE. Further, MAST outperforms Seiden-PC on `Med` and gains accuracy increases up to 14.54% on SemanticKITTI, and up to 42.85% on ONCE. As for SynLiDAR, MAST outperforms other methods on `Count` (10%) and `Avg` (4%) compared with Seiden-PC. Seiden-PC performs best on `Med`, while Seiden-PCST and MAST have similar performance to Seiden-PC.

The results reported in Tbl. 4 show that the methods with ST-based prediction (Seiden-PCST, MAST) can gain vital performance improvements compared with linear prediction (Seiden-PC) on aggregate queries with operators such as `Count` and `Med`. This comes from the nature where such operators require to analyze every PC frame to generate the final aggregate result. On the

other hand, the linear prediction may attain good performance on the Avg operator (Seiden-PC outperforms Seiden-PCST in 7 out of 10 sequences). Similar to the retrieval query, Seiden-PCST achieves the best performance on some of the sequences in the ONCE dataset. This is because of the sparse PC nature of the ONCE dataset. The empirical results also guide us in assigning suitable prediction methods for different aggregate operators.

Table 5. Evaluation on Min and Max Aggregate Operators.

Dataset	|Seq|	Aggregate Accuracy (%) Min			Aggregate Accuracy (%) Max		
		Seiden-PC	Seiden-PCST	MAST	Seiden-PC	Seiden-PCST	MAST
Semantic-KITTI	4,541	80.000	80.000	**100.000**	**93.937**	92.548	91.745
	4,661	100.000	100.000	**100.000**	94.856	**94.856**	90.396
	4,071	70.000	100.000	**100.000**	**93.750**	92.083	83.686
	4,981	80.000	80.000	**90.000**	**97.500**	90.599	93.917
	3,281	100.000	100.000	**100.000**	96.167	**98.667**	96.833
ONCE	2,741	100.000	100.000	100.000	89.802	90.913	**91.291**
	3,862	100.000	100.000	100.000	88.762	**100.000**	79.214
	2,983	100.000	100.000	100.000	79.602	79.602	**89.571**
	4,638	100.000	100.000	100.000	97.143	97.143	**97.143**
	5,264	100.000	100.000	100.000	82.361	**98.333**	82.361
SynLiDAR	45,076	70.000	70.000	**100.000**	90.000	100.000	**100.000**

We also evaluate Min and Max aggregate operators, which return the global minimum and maximum numerical results of the PC sequence, respectively. The results are shown in Tbl. 5. MAST outperforms Seiden-PC and Seiden-PCST in finding accurate Min results across three datasets, while Seiden-PC outperforms Seiden-PCST and MAST on Max operators on some of SemanticKITTI sequences. The results reveal that Seiden-PC can perform well in finding global maximum. This is because of the design of Seiden-PC's reward. We observe that the PC sequence typically has a sharp slope around the global maximum. The reward of Seiden-PC evaluates the variance of sampled frames, and thus will sample more PC frames from the area with a sharp slope.

RQ2. Evaluation of time cost. We evaluate the time cost during the query processing procedure. During the query processing, the query workload generated w.r.t. the query template in Tbl. 2

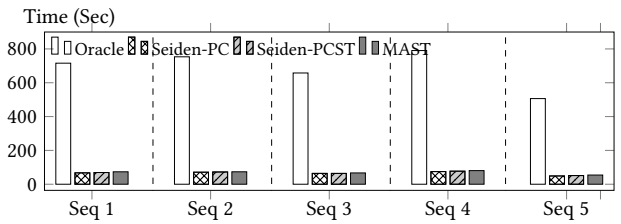


Fig. 5. Time cost evaluation of the sampling procedure (deep model processing) with the default budget set as 10%.

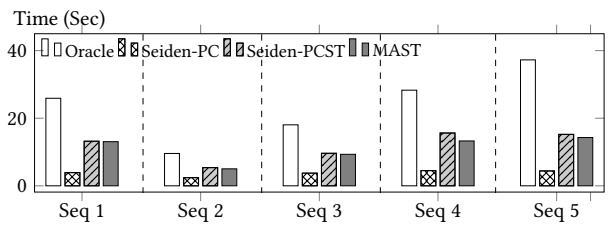


Fig. 6. Time cost evaluation of the query procedure.

contains 30 aggregate queries and 100 retrieval queries. The results of time cost during query processing on the SemanticKITTI dataset are shown in Fig. 5 and 6, where the x-axis denotes different PC sequences of the SemanticKITTI dataset. We evaluate the whole processing procedure between Oracle, Seiden-PC, Seiden-PCST, and MAST methods. We report the different parts of the process: the sampling procedure (reported in Fig. 5) and the query time (reported in Fig. 6). As for the indexing procedure, the Seiden-PC, Seiden-PCST, and MAST have a similar time cost which takes approximately 0.5 seconds.

In Fig. 5, the Oracle method takes from 506.3 to 790.4 seconds to leverage deep model and produce exact prediction, while the other methods that we implement in our PC query processing framework significantly save time cost (saving 90% of time cost by Oracle method, which is proportional to the sampling budget). Specifically, MAST takes from 54.3 to 80.5 seconds to finish the sampling procedure, Seiden-PC (resp. Seiden-PCST) takes from 48.8 (resp. 51.2) to 74.7 (resp. 77.1) seconds. MAST and Seiden-PCST take more time on sampling compared with Seiden-PC since the ST analysis includes more computation. The fact reveals that our framework could conduct efficient query analytics with a time reduction asymptotically proportional to the sampling budget.

During the query procedure, in Fig. 6, the Oracle method applies all predicted results to produce accurate results for each query, which takes from 9.5 to 37.2 seconds to process all queries. Seiden-PC takes from 2.4 to 4.5 seconds, Seiden-PCST takes from 5.3 to 15.6 seconds, and MAST takes from 5.0 to 14.2 seconds. On average, the linear prediction takes approximately 0.03 seconds to process a query while the ST prediction takes approximately 0.07 seconds to process a query. The indexing procedure has made the ST prediction 2× faster (by preventing repeated computation). Two prediction methods are in the same order of magnitude on time complexity, consistent with Section 6.1.

In the overall procedure, Oracle takes from 515.8 to 827.6 seconds to finish the query processing, while MAST takes from 59.8 to 95.2 seconds, which achieves approximately 10× improvement. In conclusion, our framework provides efficient query processing compared with the Oracle method.

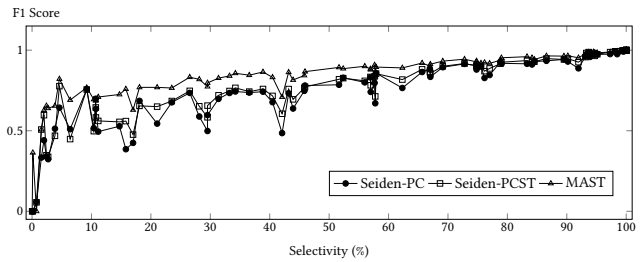


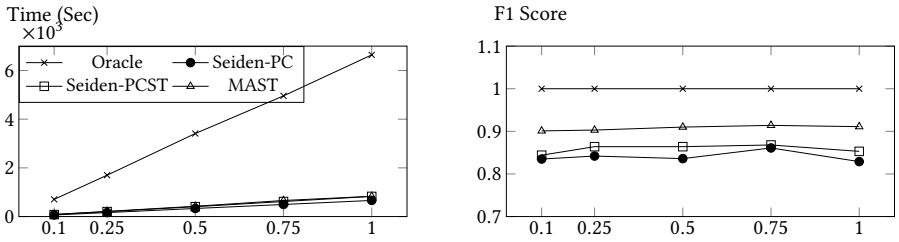
Fig. 7. The performance of retrieval queries by varying selectivity (from 0.02% to 100%).

RQ3. Performance varying retrieval selectivity. We process the retrieval queries on the sequence 0 of SemanticKITTI. And we sort the performance of each query in the workload w.r.t. the selectivity. The result is shown in Fig.7. In this figure, the selectivities of retrieval query are varying from 0.02% to 100%. In the left part of the figure when the selectivity is small, both Seiden-PC and Seiden-PCST have difficulty finding the key frames that could satisfy the filter predicates (e.g., in the retrieval query with predicates $\texttt{dist} \leq 15$ & $\texttt{Count} \geq 9$ with the selectivity of 0.132%, MAST found 2 out of 6 satisfied frames, while Seiden-PC and Seiden-PCST found none of the satisfied frames). This means that MAST has an advantage in finding sparse satisfied frames because of the ability to analyze the mobility condition, which reveals the reason for the outstanding performance compared with Seiden-PC and Seiden-PCST. Another fact is that the F1 Score tends to be higher with the increase in selectivity. Under the selectivity range of $[10\%, 65\%]$, where most of the retrieval

queries fall into, MAST consistently outperforms Seiden-PC and Seiden-PCST and provides F1 Score above 0.8 to most of the queries, while Seiden-PC and Seiden-PCST tend to provide F1 Score of approximately 0.7. When the selectivity tends to be high (from 80% to 100%), Seiden-PC and Seiden-PCST perform asymptotically similar to MAST and can accurately find frames (with F1 Score above 0.95).

In conclusion, MAST achieves good performance on queries with small selectivity with the power of mobility analysis, while in the high selectivity situation, all of the three methods can achieve satisfactory performance.

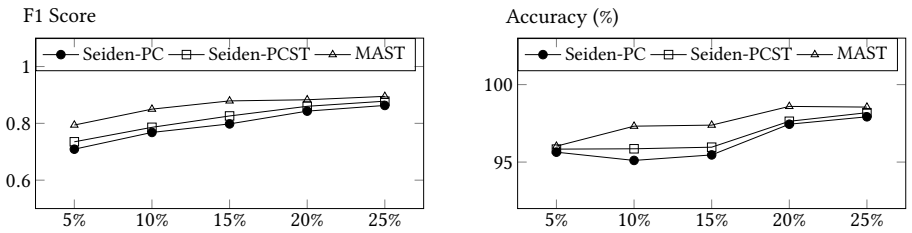
RQ4. Scalability evaluation. We evaluate the scalability of our framework on the SynLiDAR dataset, using subsets ranging from 10% of the dataset (4,507 frames) to 100% (45,076 frames). The results are shown in Fig. 8. As shown in Fig. 8a, our framework maintains its efficiency across different scales. Further, MAST also consistently provides stable performance, as shown in Fig. 8b. Also, the data are batched and fed to our framework in different percentages, which indicates the ability to maintain performance when handling new data arrival situations.



(a) Time cost. (b) Retrieval performance.

Fig. 8. The scalability evaluation of varying dataset percentage from 10% to 100%.

RQ5. Evaluation with varying sampling budgets. We vary the sampling budget for the deep model process from 5% to 25%, and the performance changes of retrieval query and aggregate query are shown in Fig. 9a and Fig. 9b, respectively.



(a) Retrieval queries. (b) Avg aggregate queries.

Fig. 9. The performance of retrieval and Avg aggregate queries of varying sampling budgets from 0.5% to 25%.

The figures show that the performance of both retrieval query and aggregate query increases with the increase in budget. Further, as the budget increases, the F1 Score demonstrates notable enhancements: Seiden-PC improves from 0.709 to 0.863, Seiden-PCST advances from 0.735 to 0.878, and MAST increases from 0.794 to 0.895. For aggregate accuracy, Seiden-PC increases from 95.639% to 97.922%, Seiden-PCST improves from 95.836% to 98.187%, and MAST increases from 96.027% to 98.549%. Other operators perform similarly with the budget increases. We notice that

MAST outperforms Seiden-PC and Seiden-PCST when the budget is low (e.g., 5%), while when the budget increases to above 20%, the performance gain becomes less significant. This means when the sampling budget becomes higher, a simpler sampling and prediction can also achieve a good performance.

Another fact is that the performance of the Avg aggregate query is satisfactory even when the sampling budget is low. It shows the sparse sample tolerance of our framework on the Avg operator.

RQ6. Evaluation using other deep models. The previous experiments are conducted using the default deep model PV-RCNN. This experiment is to evaluate the performance of MAST using other deep models. We select two other deep models namely PointRCNN [40], and SECOND [48] as the

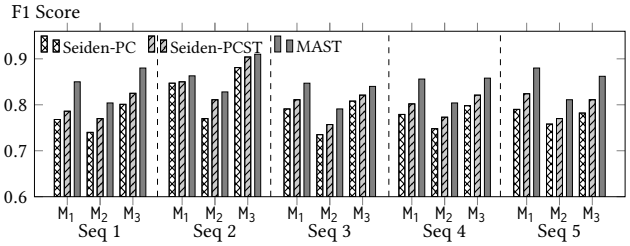
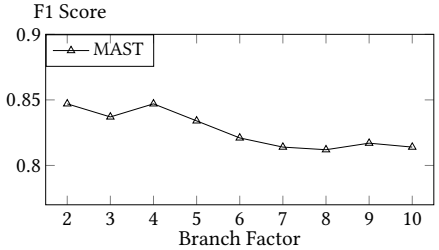
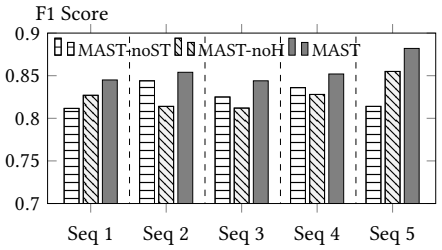


Fig. 10. The performance of retrieval queries with varying deep models, where M1, M2, M3 refers to PV-RCNN, PointRCNN, and SECOND, respectively.

Oracle models. The results on SemanticKITTI are shown in Fig. 10 and the results on other datasets are qualitatively similar. The bottom x-axis is the sequence id. The top x-axis is different models (M_1 for PV-RCNN, M_2 for PointRCNN, and M_3 for SECOND). The result shows MAST consistently outperforms Seiden-PC and Seiden-PCST with different deep models, which shows the generality of MAST. Moreover, we notice that MAST can gain the best performance when utilizing the SECOND model. One reason that MAST works the best on the SECOND model could be that the SECOND model tends to predict objects that are safe to be predicted (objects with high confidence), which alleviates the difficulty in capturing the spatio-temporal feature and produces similar results compared with the Oracle method.



(a) Varying branching factors.



(b) Evaluation of MAST variants.

Fig. 11. Evaluation of design choices of MAST.

RQ7. Evaluation of design choices. We proceed to evaluate the effect of different design choices of MAST, including the branching factor selection during the hierarchical fine-grained segmentation of MAST, and the effect of different components of MAST.

(1) Segment tree construction: We evaluate the performance of retrieval queries when varying the branching factor (the number of arms during the hierarchical segmentation) of the segment tree

from 2 to 10. The results are shown in Fig. 11a. The results show that the retrieval performance decreases when the branching factor increases. This is because when the branching factor increases, the ability to flexibly explore different subsegments with different granularity degrees decreases, since the maximum depth of a subtree will decrease when the branching factor increases.
(2) Ablation study: We evaluate the performance of multiple variants of MAST, including MAST-noST (without ST-based reward and prediction), and MAST-noH (without the hierarchical structure). Note that MAST-noH applies ST-based reward during sampling, and it is not equivalent to Seiden-PCST. We evaluate the performance of retrieval queries on the default setting. The results are shown in Fig. 11b. In this figure, the performance of both MAST-noST and MAST-noH declines compared to MAST, yet they still outperform Seiden-PC. The results show that the ST-based prediction and the hierarchical segmentation jointly enhance the query performance.

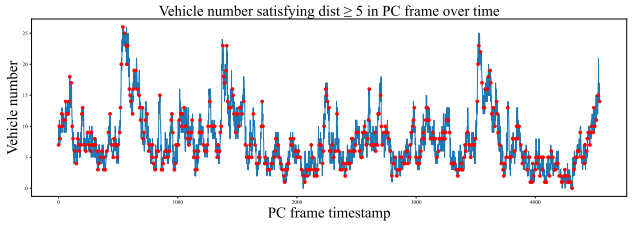


Fig. 12. Object number change over time in SemanticKITTI. The sampled PC frames by MAST are plotted in red dots.

RQ8. Preferred sampling result study. We study the question about what kind of sampling is preferred. We first select the predicate filter as `dist` ≥ 5, and the satisfied vehicle number over time, noted as $y(t)$, is shown in Fig. 12. The sampled PC frames by the MAST method are presented in red dots. The figure shows that the sampling will include majority of the local minima (suppose denoted as t_0, with the slope equals to 0 and there exists $[t_0-\delta, t_0+\delta]$ that for $\forall t \in [t_0-\delta, t_0+\delta],\ y(t_0) \leq y(t)$) and local maxima into the sampled set. As a result, the approximated function $y^a(t)$ based on the sampled set will be close to $y(t)$ and accurately indicate the up and down trends. We give a detailed theoretical analysis of the correctness of the aggregate query results in this situation.

8 Conclusions and Future Work

In this paper, we study efficient Point Cloud query processing. We design a framework that leverages the spatio-temporal feature of PC data to guide the sampling and querying procedures. Further, we explore how reinforcement learning techniques can enhance the sampling process, thereby increasing the accuracy of approximate query results. Experimental results show that our framework provides accurate and efficient PC query processing. In future work, we aim to extend our framework to support more complex queries, including join queries, multi-step operations, and intricate spatial and semantic filters, to facilitate more sophisticated data analysis. Additionally, we plan to integrate multimodal data sources, such as images, textual descriptions, and sensor data, to enrich the application scenarios of our system.

Acknowledgments

This research is supported in part by Singapore MOE AcRF Tier-2 grants MOE-T2EP20221-0015 and MOE-T2EP20223-0004, and MOE AcRF Tier-1 grant RT6/23. We also thank Jianyang for the insightful suggestions provided on the theoretical proof of this paper.

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] Peter Auer. 2002. Using Confidence Bounds for Exploitation-Exploration Trade-offs. *J. Mach. Learn. Res.* 3 (2002), 397–422. https://jmlr.org/papers/v3/auer02a.html

[3] Jaeho Bang, Gaurav Tarlok Kakkar, Pramod Chunduri, Subrata Mitra, and Joy Arulraj. 2023. SEIDEN: Revisiting Query Processing in Video Database Systems. *Proc. VLDB Endow.* 16, 9 (2023), 2289–2301. https://doi.org/10.14778/3598581.3598599

[4] 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*. ACM, 1907–1921. https://doi.org/10.1145/3318464.3389692

[5] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jürgen Gall. 2019. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 9296–9306. https://doi.org/10.1109/ICCV.2019.00939

[6] Qingpeng Cai, Can Cui, Yiyuan Xiong, Wei Wang, Zhongle Xie, and Meihui Zhang. 2023. A Survey on Deep Reinforcement Learning for Data Processing and Analytics. *IEEE Trans. Knowl. Data Eng.* 35, 5 (2023), 4446–4465. https://doi.org/10.1109/TKDE.2022.3155196

[7] Jiashen Cao, Ramyad Hadidi, Joy Arulraj, and Hyesoon Kim. 2021. THIA: Accelerating Video Analytics using Early Inference and Fine-Grained Query Planning. *CoRR* abs/2102.08481 (2021). arXiv:2102.08481 https://arxiv.org/abs/2102.08481

[8] 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*. ACM, 559–572. https://doi.org/10.1145/3514221.3517857

[9] Yueting Chen, Nick Koudas, Xiaohui Yu, and Ziqiang Yu. 2022. Spatial and Temporal Constrained Ranked Retrieval over Videos. *Proc. VLDB Endow.* 15, 11 (2022), 3226–3239. https://doi.org/10.14778/3551793.3551865

[10] 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*. ACM, 545–558. https://doi.org/10.1145/3514221.3526181

[11] Tu Gu, Kaiyu Feng, Gao Cong, Cheng Long, Zheng Wang, and Sheng Wang. 2023. The RLR-Tree: A Reinforcement Learning Based R-Tree for Spatial Data. *Proc. ACM Manag. Data* 1, 1 (2023), 63:1–63:26. https://doi.org/10.1145/3588917

[12] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. 2021. Deep Learning for 3D Point Clouds: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* 43, 12 (2021), 4338–4364. https://doi.org/10.1109/TPAMI.2020.3005434

[13] Brandon Haynes, Amrita Mazumdar, Armin Alaghi, Magdalena Balazinska, Luis Ceze, and Alvin Cheung. 2018. LightDB: A DBMS for Virtual Reality Video. *Proc. VLDB Endow.* 11, 10 (2018), 1192–1205. https://doi.org/10.14778/3231751.3231768

[14] Wenjia He, Michael R. Anderson, Maxwell Strome, and Michael J. Cafarella. 2020. A Method for Optimizing Opaque Filter Queries. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. ACM, 1257–1272. https://doi.org/10.1145/3318464.3389766

[15] Wenjia He, Ibrahim Sabek, Yuze Lou, and Michael J. Cafarella. 2024. Optimizing Video Selection LIMIT Queries With Commonsense Knowledge. *Proc. VLDB Endow.* 17, 7 (2024), 1751–1764. https://doi.org/10.14778/3654621.3654639

[16] Jing Huang and Suya You. 2016. Point cloud labeling using 3D Convolutional Neural Network. In *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016*. IEEE, 2670–2675. https://doi.org/10.1109/ICPR.2016.7900038

[17] Gaurav Tarlok Kakkar, Jiashen Cao, Pramod Chunduri, Zhuangdi Xu, Suryatej Reddy Vyalla, Prashanth Dintyala, Anirudh Prabakaran, Jaeho Bang, Aubhro Sengupta, Kaushik Ravichandran, Ishwarya Sivakumar, Aryan Rajoria, Ashmita Raju, Tushar Aggarwal, Abdullah Shah, Sanjana Garg, Shashank Suman, Myna Prasanna Kalluraya, Subrata Mitra, Ali Payani, Yao Lu, Umakishore Ramachandran, and Joy Arulraj. 2023. EVA: An End-to-End Exploratory Video Analytics System. In *Proceedings of the Seventh Workshop on Data Management for End-to-End Machine Learning, DEEM 2023, Seattle, WA, USA, 18 June 2023*. ACM, 8:1–8:5. https://doi.org/10.1145/3595360.3595858

[18] Gaurav Tarlok Kakkar, Aryan Rajoria, Myna Prasanna Kalluraya, Ashmita Raju, Jiashen Cao, Kexin Rong, and Joy Arulraj. 2023. Interactive Demonstration of EVA. *Proc. VLDB Endow.* 16, 12 (2023), 4082–4085. https://doi.org/10.14778/3611540.3611626

[19] 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

[20] 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

[21] Daniel Kang, Edward Gan, Peter Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2020. Approximate Selection with Guarantees using Proxies. *Proc. VLDB Endow.* 13, 11 (2020), 1990–2003. http://www.vldb.org/pvldb/vol13/p1990-kang.pdf

[22] 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

[23] Daniel Kang, John Guibas, Peter Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2020. Semantic indexes for machine learning-based queries over unstructured data. *arXiv preprint arXiv:2009.04540* (2020).

[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*. ACM, 1934–1947. https://doi.org/10.1145/3514221.3517897

[25] Daniel Kang, Francisco Romero, Peter D. Bailis, Christos Kozyrakis, and Matei Zaharia. 2022. VIVA: An End-to-End System for Interactive Video Analytics. In *12th Conference on Innovative Data Systems Research, CIDR 2022, Chaminade, CA, USA, January 9-12, 2022*. www.cidrdb.org. https://www.cidrdb.org/cidr2022/papers/p75-kang.pdf

[26] Ferdinand Kossmann, Ziniu Wu, Eugenie Lai, Nesime Tatbul, Lei Cao, Tim Kraska, and Sam Madden. 2023. Extract-Transform-Load for Video Streams. *Proc. VLDB Endow.* 16, 9 (2023), 2302–2315. https://doi.org/10.14778/3598581.3598600

[27] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.

[28] Jiangneng Li, Zheng Wang, Gao Cong, Cheng Long, Han Mao Kiah, and Bin Cui. 2023. Towards Designing and Learning Piecewise Space-Filling Curves. *Proc. VLDB Endow.* 16, 9, 2158–2171.

[29] Jiangneng Li, Haitao Yuan, Gao Cong, Han Mao Kiah, and Shuhao Zhang. 2025. MAST: Towards Efficient Analytical Query Processing on Point Cloud Data [Technical Report]. https://github.com/gravesprite/MAST/blob/main/MAST_technical_report.pdf

[30] Eric Liang, Hang Zhu, Xin Jin, and Ion Stoica. 2019. Neural packet classification. In *Proceedings of the ACM Special Interest Group on Data Communication, SIGCOMM 2019, Beijing, China, August 19-23, 2019*. ACM, 256–269. https://doi.org/10.1145/3341302.3342221

[31] Jiageng Mao, Minzhe Niu, Chenhan Jiang, Hanxue Liang, Jingheng Chen, Xiaodan Liang, Yamin Li, Chaoqiang Ye, Wei Zhang, Zhenguo Li, Jie Yu, Chunjing Xu, and Hang Xu. 2021. One Million Scenes for Autonomous Driving: ONCE Dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

[32] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. 2019. Neo: A Learned Query Optimizer. *Proc. VLDB Endow.* 12, 11 (2019), 1705–1718. https://doi.org/10.14778/3342263.3342644

[33] 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

[34] n.d. 2023. PostgreSQL pgPointcloud. https://pgpointcloud.github.io/pointcloud/. [Online; accessed 27-Jan-2023].

[35] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 77–85. https://doi.org/10.1109/CVPR.2017.16

[36] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. (2017), 5099–5108. https://proceedings.neurips.cc/paper/2017/hash/d8bf84be3800d12f74d8b05e9b89836f-Abstract.html

[37] Matthew Russo, Tatsunori Hashimoto, Daniel Kang, Yi Sun, and Matei Zaharia. 2023. Accelerating Aggregation Queries on Unstructured Streams of Data. *Proc. VLDB Endow.* 16, 11 (2023), 2897–2910. https://doi.org/10.14778/3611479.3611496

[38] Radu Bogdan Rusu and Steve Cousins. 2011. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*. IEEE. https://doi.org/10.1109/ICRA.2011.5980567

[39] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. 2020. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 10526–10535. https://doi.org/10.1109/CVPR42600.2020.01054

[40] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 2019. PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 770–779. https://doi.org/10.1109/CVPR.2019.00086

[41] Kaleem Siddiqi, Juan Zhang, Diego Macrini, Ali Shokoufandeh, Sylvain Bouix, and Sven J. Dickinson. 2008. Retrieving articulated 3-D models using medial surfaces. *Mach. Vis. Appl.* 19, 4 (2008), 261–275. https://doi.org/10.1007/S00138-007-0097-8

[42] Abhijit Suprem, Joy Arulraj, Calton Pu, and João Eduardo Ferreira. 2020. ODIN: Automated Drift Detection and Recovery in Video Analytics. *Proc. VLDB Endow.* 13, 11 (2020), 2453–2465. http://www.vldb.org/pvldb/vol13/p2453-suprem.pdf

[43] OpenPCDet Development Team. 2020. OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds. https://github.com/open-mmlab/OpenPCDet.

[44] Renzhi Wu, Pramod Chunduri, Dristi J. Shah, Ashmitha Julius Aravind, Ali Payani, Xu Chu, Joy Arulraj, and Kexin Rong. 2024. SketchQL Demonstration: Zero-shot Video Moment Querying with Sketches. *Proc. VLDB Endow.* 17, 12 (2024), 4429–4432. https://doi.org/10.14778/3685800.3685892

[45] Aoran Xiao, Jiaxing Huang, Dayan Guan, Fangneng Zhan, and Shijian Lu. 2022. Transfer Learning from Synthetic to Real LiDAR Point Cloud for Semantic Segmentation. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*. AAAI Press, 2795–2803. https://doi.org/10.1609/AAAI.V36I3.20183

[46] Jiong Xie, Zhen Chen, Jianwei Liu, Fang Wang, Feifei Li, Zhida Chen, Yinpei Liu, Songlu Cai, Zhenhua Fan, Fei Xiao, and Yue Chen. 2022. Ganos: A Multidimensional, Dynamic, and Scene-Oriented Cloud-Native Spatial Database Engine. *Proc. VLDB Endow.* 15, 12 (2022), 3483–3495. https://doi.org/10.14778/3554821.3554838

[47] Zhuangdi Xu, Gaurav Tarlok Kakkar, Joy Arulraj, and Umakishore Ramachandran. 2022. EVA: A Symbolic Approach to Accelerating Exploratory Video Analytics with Materialized Views. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*. ACM, 602–616. https://doi.org/10.1145/3514221.3526142

[48] Yan Yan, Yuxing Mao, and Bo Li. 2018. SECOND: Sparsely Embedded Convolutional Detection. *Sensors* 18, 10 (2018), 3337. https://doi.org/10.3390/S18103337

[49] Yisong Yue, Sue Ann Hong, and Carlos Guestrin. 2012. Hierarchical Exploration for Accelerating Contextual Bandits. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress. http://icml.cc/2012/papers/933.pdf

[50] Dongxiang Zhang, Teng Ma, Junnan Hu, Yijun Bei, Kian-Lee Tan, and Gang Chen. 2023. Co-movement Pattern Mining from Videos. *Proc. VLDB Endow.* 17, 3 (2023), 604–616. https://doi.org/10.14778/3632093.3632119

Received July 2024; revised September 2024; accepted November 2024