Craw: A Unified and Efficient Querying Framework for Large-Scale Video Datasets

Ziqi Zhou
Nanchang University
Nanchang, China
ziqizhou@email.ncu.edu.cn

Hanjian Jiang
Nanchang University
Nanchang, China
jianghj0123@outlook.com

Zihao Zeng
Nanchang University
Nanchang, China
zihaozeng0021@gmail.com

Xupuzhe Shao
Nanchang University
Nanchang, China
xpzshao@email.ncu.edu.cn

Zichen Xu
Nanchang University
Nanchang, China
xuz@ncu.edu.cn

ABSTRACT

The ubiquitous deployment of cameras has led to explosive growth of video data, creating an urgent need to explore valuable content. Single-level queries are insufficient to extract comprehensive information, raising the demand for multi-level queries (existence, dynamic, similarity) within a single unified system. However, limited by the high complexity and redundancy of video, existing systems usually support single-level queries, while Vision-Language Models that support multi-level queries incur prohibitive computational overhead, making them infeasible for large-scale video datasets.

To address these issues, we propose Craw, a framework for efficient multi-level queries on large-scale video datasets. Specifically, Craw (1) designs the Video Semantic Unit to encapsulate video semantics, (2) develops a semantic-preserving video segmentation algorithm, and (3) constructs a hybrid index framework integrating an inverted index with a cluster index layer for efficient query execution. Experimental results show that Craw outperforms the state-of-the-art (SOTA) by reducing query latency up to two orders of magnitude, while effectively supporting multi-level queries.

PVLDB Reference Format:
Ziqi Zhou, Hanjian Jiang, Zihao Zeng, Xupuzhe Shao, and Zichen Xu. Craw: A Unified and Efficient Querying Framework for Large-Scale Video Datasets. PVLDB, 19(11): 2964 - 2977, 2026.
doi:10.14778/3836663.3836666

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/CrawZhou/Craw_Submission.

1 INTRODUCTION

The pervasive deployment of various cameras has expanded dramatically [50], driven by growing demands such as smart city development [44], transcending traditional monitoring purposes. According to the latest report, the global video surveillance market is projected to grow from $56.11 billion in 2025 to $88.06 billion by 2030 [32]. These cameras are widely used in diverse scenarios like traffic accident analysis [51], where practical requirements go beyond single-level query capabilities, thus increasingly demanding support for multi-level queries (existence, dynamic, similarity) [4, 22, 48]. For instance, traffic analysts may need to query specific objects (existence query), track dynamic events (e.g., two vehicles approaching each other), or optimize signal timing by analyzing traffic flow via similar video segments [47] (similarity query).

Existing techniques struggle to fully address this practical demand for multi-level queries. Vision-Language Models (VLMs) can support multi-level queries [46] but rely on prohibitive frame-level processing, making them infeasible for large-scale video datasets [24]. Additionally, unlike practical scenarios that require structured queries [4], VLMs take natural language as input, which may lead to hallucinations and fail to provide accurate, real-time responses. Meanwhile, most specialized systems only support a single query level, requiring multiple separate systems to meet practical demands like traffic analysis, thus limiting flexibility and efficiency. These limitations highlight the necessity of a unified querying framework that supports multi-level queries while maintaining low overhead.

Achieving efficient multi-level queries in large-scale datasets is non-trivial, posing several critical challenges: **(1) Designing appropriate semantic representation.** To support multi-level queries, the framework requires a proper representation method to encapsulate fine-grained semantics and organizes them in a structured manner [20, 31]. **(2) Maintaining semantic integrity.** Practical scenarios often require structured and accurate query results, and the preservation of video semantic integrity is the key to ensuring query accuracy. However, traditional methods such as fixed interval [49] and temporal difference [18, 38] often rely on frame-pixel differences and ignore high-level semantics (e.g., "two pedestrians side by side"), leading to fragmented segments that undermine semantic integrity and query accuracy. **(3) Enabling low overhead.** Efficient query performance is critical for large-scale video datasets. VLMs may incur heavy computational costs due to frame-level feature extraction and cross-modal alignment, while the multi-system deployment of specialized systems introduces cumulative operational overhead and results in low execution efficiency.

Selecting an appropriate video semantic representation is foundational to addressing the three critical challenges outlined above, and ultimately determines the feasibility of multi-level queries. Existing methods can be mainly classified into three categories: key

Proceedings of the VLDB Endowment, Vol. 19, No. 11 ISSN 2150-8097.
doi:10.14778/3836663.3836666

frame (KF), full video (FV) and video segment (VS). These categories have inherent limitations and fail to support multi-level queries efficiently. KF [9, 17, 23] selects key frames from videos and extracts semantics from these frames as the "video semantic abbreviation". However, KF only captures static semantics at discrete moments, failing to capture dynamic semantics [21], making KF incapable of supporting dynamic queries. In contrast, FV [10, 16, 26] extracts holistic semantic information. Yet FV faces dual limitations: heavy reliance on manual processing and overly coarse granularity. Lacking the spatio-temporal information required for dynamic queries, FV only supports coarse-grained existence and similarity queries. VS [18, 49] serves as an intermediate granularity between KF and FV, balancing the efficiency of KF with the semantic coverage of FV. Nevertheless, current VS still falls short in maintaining semantic integrity, leading to fragmented semantics and low query accuracy. Therefore, we need an advanced video semantic representation.

To address these challenges, we propose Craw, a unified framework for efficient multi-level queries on large-scale video datasets. Craw adopts a lightweight design with one-time preprocessing to generate fine-grained, structured video semantic representation (i.e., VSU), employs a semantic-preserving video segmentation algorithm and a trajectory simplification algorithm to guarantee query accuracy and lower index construction overhead, and constructs a hybrid index framework, which integrates a cluster index layer and an inverted index and incorporates a Top-k strategy to accelerate query execution. Experimental results show that Craw achieves 14–34x query speedup over Video-zilla and outperforms the SOTA in both latency and accuracy.

Our contributions in this paper are as follows:

- We summarize the inherent limitations and key challenges of existing systems, clarify the research motivation for multi-level query demands, and introduce Craw, a novel unified framework for efficient multi-level queries on large-scale video datasets.
- We design the VSU to represent video semantics, and propose an innovative video segmentation algorithm that divides videos into semantically integral segments, preserving semantic integrity. We further design a trajectory simplification algorithm to reduce computational overhead while preserving query accuracy.
- We establish a hybrid index framework based on VSUs. This framework effectively reduces query execution overhead, maintains high query accuracy, and adapts to multi-level queries.
- We conduct comprehensive experiments on large-scale video datasets, comparing Craw with SOTA from multiple perspectives. Experimental results demonstrate that Craw achieves superior query performance and validates the effectiveness of all our proposed algorithms and the overall framework.

2 RELATED WORK

This section reviews related work and further elaborates on their limitations. Table 1 summarizes the comparison between Craw and some representative research.

Existence Query. Existence queries verify the presence of target objects and their relations. Recent works focus on two objectives. The first aims to reduce overhead: NoScope [23] skips redundant frames, Gecko [43] selects lightweight models via content-aware analysis, and Seiden [4] adopts temporally-guided sampling. However, these methods fail to capture inter-object relations. The second targets fine-grained queries (e.g., "a car adjacent to another car") using cross-modal techniques such as FIGO [9] and LOVO [28], but suffers from high preprocessing overhead. Craw bridges both objectives: VSUs capture fine-grained semantics and drastically reduce search space.

Table 1: Comparison of existing research and Craw. ✓̸ indicates indirect support for the corresponding query level.

Research	Existence Query	Dynamic Query	Similarity Query
FIGO [9]	✓	×	×
Video-zilla [18]	×	×	✓
MIRIS [6]	✓̸	✓	×
EQUI-VOCAL [49]	✓̸	✓	×
VLMs [45]	✓	✓	✓
Craw	✓	✓	✓

Dynamic Query. Dynamic queries rely on spatio-temporal information and thus require both semantic integrity and low-overhead execution. Current systems primarily adopt two approaches and select VS as representation, each with limitations. Trajectory-based approaches maintain semantic integrity, and aim to reduce query overhead. While methods like MIRIS [6] optimizes the sampling frame rate, whereas OTIF [7] generates reusable trajectories for subsequent queries, they still incur heavy preprocessing due to video high redundancy. Predefined event-based approaches enable structured queries but are constrained by fixed segmentation. For example, EQUI-VOCAL [49] enables declarative query synthesis, eliminating the need to manually specify events (e.g., "a car is to the right of another car") to improve efficiency. In contrast, Craw uses a semantic-preserving algorithm to maintain semantic integrity, and reduces redundant trajectory points to improve query efficiency.

Similarity Query. Similarity queries retrieve semantically similar video fragments and often select VS to capture semantics. Current methods like Video-zilla [18] group frames with similar object distributions into semantic units and leverage cluster indexes to accelerate retrieval. However, Video-zilla only performs video segmentation from a pixel-level perspective, without exploiting the high-level semantics of video content. Meanwhile, its high computational cost in video segmentation and reliance on high-dimensional feature representations severely limit its overall efficiency. Craw reduces the segmentation costs and feature dimensionality, preserving semantic integrity while improving the query efficiency.

VLMs. While current VLMs have demonstrated the multi-level retrieval capabilities, they incur substantial overhead [42]. This stems from the reliance on frame-level feature extraction and fine-grained cross-modal alignment, making VLMs infeasible for large-scale video datasets. Thus, VLMs are usually limited to short videos [34]. Additionally, VLMs take natural language as input, which may lead to hallucinations and erroneous outputs. Compared to VLMs, Craw adopts lightweight preprocessing and leverages structured VSUs to support multi-level queries. Further, Craw uses structured queries via VSUs to avoid hallucinations effectively.

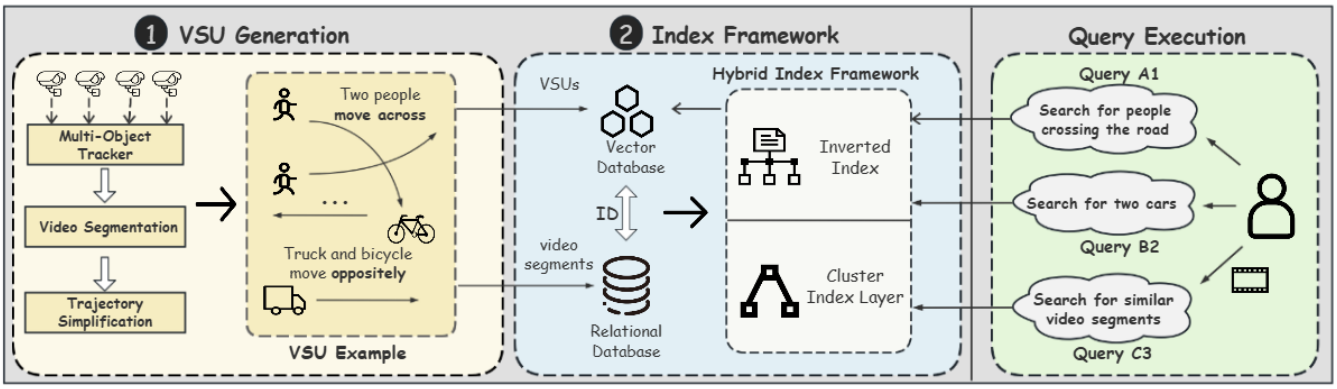


Figure 1: System overview of Craw, consisting of two core modules: VSU Generation and Index Framework, with the corresponding multi-level query execution pipeline.

3 OVERVIEW OF CRAW

In this section, we provide a system overview of Craw. As illustrated in Figure 1, Craw mainly comprises two key modules: *VSU Generation* (Sections 4–5) and *Index Framework* (Sections 6.1–6.2). We further elaborate on the *Query Execution* pipeline (Section 6.3).

In this paper, Craw is designed for comprehensive application scenarios like urban traffic monitoring and smart city management, where integrated multi-level query capabilities are in critical demand and which generate large-scale long videos captured by fixed cameras. Compared to non-static scenarios like on-board cameras, the latter have far less prominent need for multi-level queries and usually prioritize real-time object tracking or specific event detection like lane departure warning.

VSU Generation. This module is designed to process unstructured raw videos into structured VSUs for multi-level queries. The module first uses a multi-object tracker to capture continuous spatio-temporal trajectories across frames. Based on this trajectory information, this module leverages an adaptive video segmentation algorithm, converting raw videos into video segments with semantic integrity. To further improve efficiency, the module adopts a trajectory simplification algorithm that extracts key trajectory points. Each segment is converted into a VSU, containing trajectory, interaction and temporal matrices. The module encapsulates each segment as a query-friendly VSU, eliminating repetitive processing and decreasing computational costs.

Index Framework. To enable efficient multi-level query execution, this module constructs a hybrid index framework containing a cluster index layer and an inverted index layer. The inverted index layer acts as an effective locator and filter to drastically narrow the search space, while the cluster index layer improves efficiency for similarity queries using proposed metrics. This combination adapts to the diverse requirements of multi-level queries. Notably, while modern storage systems like LanceDB [25] and Deep Lake [13] support co-storage of metadata and vectors, they lack support for custom similarity measures during index construction, and thus cannot satisfy the demands of multi-level queries.

Query Execution. We further illustrate how Craw performs multi-level queries using the hybrid index and VSUs: ***Query A1***. For trajectory-based dynamic queries, Craw first uses the inverted index to filter irrelevant VSUs, drastically narrowing the search space, then performs matching on the remaining candidates and returns eligible video segments. ***Query B2***. For existence and predefined event-based dynamic queries, Craw directly invokes the inverted index for retrieval and returns results. ***Query C3***. For similarity queries, Craw extracts semantics from the input segment to form a single VSU without extra segmentation, performs fast matching via the cluster index layer, ranks candidates by similarity, and returns the most relevant video segments. For instance, when a traffic analyst issues the query "A bus stops in the station and two people are present" in a traffic junction, Craw first uses the inverted index to locate VSUs containing buses and people, then matches the spatio-temporal information in the remaining VSUs, and finally returns the eligible video segments.

4 VIDEO SEMANTIC UNIT

In this section, we first define a VSU and introduce its generation workflow (Section 4.1), which relies on the video segmentation algorithm in Section 5. We then propose a novel trajectory simplification algorithm (Section 4.2).

4.1 Defining Video Semantic Unit

Video is defined as a continuous time sequence of still frames [39], and its semantics describe the content of temporal dynamics and visual content of these frames [19]. To systematically characterize video semantics, we decompose it into a collection of multi-object events and define a VSU as the triple:

$$\text{VSU} = \{\mathcal{F}, \mathcal{I}, \mathcal{T}\}$$

where $\mathcal{F}$, $\mathcal{I}$ and $\mathcal{T}$ denote the trajectory matrix ($\mathcal{F}$), interaction matrix ($\mathcal{I}$), and temporal matrix ($\mathcal{T}$) respectively. Each matrix is a two-dimensional structure.

- $\mathcal{F}$: $\mathcal{F}$ describes the motion features of each object in the video segment, where each object's trajectory is represented by key trajectory points. These points are selected from the original trajectory matrix $\mathcal{F}'$ via the trajectory simplification algorithm (Section 4.2) to form the final $\mathcal{F}$ matrix.

• $\mathcal{I}$: $\mathcal{I}$ characterizes interaction relationships among objects in a VSU. We adopt an encoding scheme based on spatio-temporal relationships between object pairs (computed from $\mathcal{F}'$), such as the centroid distance and its temporal change. For example, code "0" denotes no specific relationship, and code "1" indicates that object A is moving away from B (i.e., the distance increases consistently over consecutive frames). Besides, these interaction relationships can be flexibly extended to support additional user-defined relations.

• $\mathcal{T}$: $\mathcal{T}$ captures the relative times of appearance and disappearance for each object within the video segment, with values normalized to the range $[0, 1]$ based on the total duration of the video segment.

Given a video segment (Figure 2 "VSU Example"). We assume that two people appear in the first half, and a bicycle and a truck appear in the second half. Their temporal matrices $\mathcal{T}$ are thus normalized to $[0, 0.5]$ for people and $[0.5, 1]$ for the bicycle and truck, respectively. Since the people and the two vehicles do not overlap in time, their corresponding entries in $\mathcal{I}$ are encoded as "0"; the bicycle is moving away from the truck, determined from the trajectory information (i.e., $\mathcal{F}'$), so this pair is encoded as "1".

VSU Generation Workflow. Craw first uses a multi-object tracking mechanism to extract object trajectories, like combining the YOLO series [35] for object detection with the BOT-SORT algorithm [1] for tracking. Each object is represented by the midpoint of the four-point 2-D bounding box coordinates in the original frame, and the class is determined by the highest-confidence classification output from the detector. Based on the tracking results, Craw then utilizes the video segmentation algorithm (Section 5) to partition the full video into semantic-preserving segments. For each segment, we assume there are m objects, each with at most n trajectory points indicating the object's presence throughout the segment, and derive an $m \times (2n + 1)$ matrix $\mathcal{F}'$, where each row corresponds to one object and contains the object class along with n pairs of (x, y) trajectory coordinates. This structure preserves spatio-temporal information consistency and serves as the foundation for constructing the $m \times m$ matrix $\mathcal{I}$ and the $m \times 2$ matrix $\mathcal{T}$. However, when some object is not shown at all times, $\mathcal{F}'$ may contain missing entries. To resolve this, Craw can fill missing trajectory entries with placeholder values outside the valid coordinate range, preserving row structure integrity and ensuring dimensional consistency for subsequent VSU matrix computation.

4.2 Trajectory Simplification Algorithm

The trajectory simplification algorithm selects critical keypoints to represent the entire trajectory, eliminating redundancy from dense sequences while preserving the overall motion pattern of objects. These points are identified as salient based on their geometric attributes (e.g., curvature, angular change, distance proportion), which effectively characterize the object's motion trend. This selective retention represents the entire trajectory path within an admissible error bound [29], thereby accelerating index construction and reducing query execution latency.

Given the matrix $\mathcal{F}' = [F'(O_1), F'(O_2), \ldots, F'(O_m)]^\top$, the algorithm first extracts the object class from each row and cleanses placeholder values, yielding the cleaned matrix $\mathcal{F}''$. For each $\mathcal{F}''(O_i)$, we first assign unique sequential indices to its coordinate points, serving as the spatial-temporal identifier for each point, enabling precise localization of points (Algorithm 1, function Preprecess). We then partition the indices into N segments to select N keypoints P (Algorithm 1, function AnalyzeTriangles). These keypoints form $N - 2$ triangles for further analysis using three core parameters: Q (distance proportion threshold), θ (angular change), and D (recursion depth, anti-infinite loop). Additionally, the values of N, θ, and D can be selected based on trajectory complexity and scene precision requirements. For high-precision scenes (e.g., indoor surveillance), we use a larger N with smaller θ and moderate D; for low-precision scenarios (e.g., highway monitoring), we adopt a smaller N with larger θ and smaller D to balance query accuracy and computational efficiency.

Algorithm 1 Trajectory Simplification Algorithm

Require: $\mathcal{F}'$, Q, θ, D, N
Ensure: Simplified trajectory matrix $\mathcal{F}$
$L_{\max} \leftarrow 0$; $\mathcal{F} \leftarrow \varnothing$
for each $F'(O_i) \in \mathcal{F}'$ **do**
  $(\text{cls}, \mathcal{F}''(O_i)) \leftarrow \textsc{Preprocess}(F'(O_i))$
  $K \leftarrow \textsc{AnalyzeTriangles}(\mathcal{F}''(O_i), Q, \theta, D, N)$
  $K \leftarrow \textsc{UniformPad}(K, L_{\max})$
  $L_{\max} \leftarrow \max(L_{\max}, |K|)$
  $\mathcal{F} \leftarrow \mathcal{F} \cup ([\text{cls}] + \textsc{Flatten}(K))$
end for
return $\mathcal{F}$
function $\textsc{Preprocess}(F'(O_i))$
  cls $\leftarrow$ ExtractClass$(F'(O_i))$
  $\mathcal{F}''(O_i) \leftarrow$ CleanPlaceholder$(F'(O_i))$
  $\mathcal{F}''(O_i) = \{(P, i) \mid P \in \mathcal{F}''(O_i), i = 0, \ldots, \lfloor |\mathcal{F}''(O_i)|/2 \rfloor - 1\}$
  return $(\text{cls}, \mathcal{F}''(O_i))$
end function
function $\textsc{AnalyzeTriangles}(\mathcal{F}''(O_i), Q, \theta, D, N)$
  $P \leftarrow \{\mathcal{F}''(O_i)_{\lfloor k(|\mathcal{F}''(O_i)|-1)/(N-1) \rfloor} \mid k = 0, \ldots, N-1\}$
  $K \leftarrow \bigcup_{t=0}^{N-3} \textsc{AnalyzeDelta}(P_t, P_{t+1}, P_{t+2}, \mathcal{F}''(O_i), Q, \theta, D)$
  return $\textsc{Sort}(K)$
end function

Each triangle defined by vertices P_t, P_{t+1}, P_{t+2} is analyzed to determine whether its vertices should be retained (Algorithm 1, function AnalyzeTriangles). According to Euclidean distance, the analysis involves calculating the total distance d_{total} of the whole trajectory, and then calculating d_{13} between P_t and P_{t+2}, gaining the distance proportion $\rho = \frac{d_{13}}{d_{total}}$ and the angle α among P_t, P_{t+1}, P_{t+2}. If $\rho \le Q$ or $\alpha \ge \theta$ or $D = 0$, the vertices P_t, P_{t+1}, P_{t+2} are all retained. Otherwise, a midpoint is selected based on the distance between P_t and P_{t+2}. The algorithm recursively analyzes the new triangles, until meeting the conditions or reaching maximum recursion depth.

After extracting the trajectory keypoints K for each object, we sort them by their original index values. To ensure trajectory uniformity, we pad each trajectory's keypoints K to match the maximum number of keypoints L_{max} found among all trajectories (Algorithm 1, function UniformPad). Concretely, the Euclidean distance between each pair of consecutive keypoints in K is calculated, and a new keypoint is inserted into the keypoint segment with the largest distance, at the index midpoint of that segment. This process is repeated until the trajectory contains exactly $L_{\max}$ keypoints. After supplementation, dimensional consistency is ensured.

The final output of the process is the simplified trajectory matrix $\mathcal{F}$. The complete algorithm is summarized in Algorithm 1. Notably,

instead of directly saving the unsimplified $\mathcal{F}'$ into the VSU and then simplifying it, we first generate $\mathcal{I}$ from $\mathcal{F}'$, and subsequently simplify $\mathcal{F}'$ into $\mathcal{F}$, yielding the complete VSU.

5 VIDEO SEGMENTATION ALGORITHM

In this section, we first present the problem definition (Section 5.1). We then describe the process of the video segmentation algorithm (Section 5.2), a prerequisite for VSU generation.

5.1 Problem Definition

Existing methods often fail to preserve semantic integrity in scenes with high similarity, leading to fragmented segments. To address this issue, we first define *Events* and *Semantic Integrity* as follows:

Definition 1. ***Events***. A video can be regarded as a collection of continuous events. We define an event as the temporal dynamic changes of one or multiple objects in the scene, where the event duration is quantified by the occurrence duration of the involved objects (i.e., the time span from an object entering to exiting the video frame). For example, in a traffic scene, the duration of the event "a car passing through an intersection" approximately equals the car's frame entry-to-exit time span. Notably, each event corresponds to a VSU for segmentation here, and we only take the temporal dimension here.

Definition 2. ***Semantic Integrity***. We define semantic integrity as the ability of VSUs to fully encapsulate the integral semantic information of events. This reflects the time concentration degree of objects' occurrence durations, which is critical to avoiding event fragmentation. Since events are defined as the temporal dynamic changes of objects, the concentration of objects' occurrence durations determines the complete representation of events. To evaluate semantic integrity, we propose the Duration Integrity Percentage (DIP). Let $\sum_i T_i$ denote the total occurrence duration of a single object, and $\max_i T_i$ denote its longest continuous occurrence duration across all segments. The DIP is formally defined as:

$$\text{DIP} = \frac{\max_i T_i}{\sum_i T_i} \tag{1}$$

Our goal is to maximize semantic completeness. When the semantic integrity is high, it implies that VSUs preserve and represent relatively complete video semantics.

5.2 Video Segmentation Algorithm

Initial Segmentation. As shown in Algorithm 2, the algorithm begins by identifying all objects present in the first frame of the video, denoted as $O_0 = \{o_1, o_2, \ldots, o_m\}$, and proceeds to track each object $o \in O_0$ until it disappears. It also tracks any new objects that may appear in subsequent frames (all tracked objects are denoted as O_{all}). Considering that some objects may persist across extended periods, the algorithm introduces an overlap threshold σ and a temporal threshold τ. When the proportion of frames in which a single object o appears, represented as $p(o) = \ell(o)/|V|$, exceeds σ, we calculate the ratio of the current inactive duration t_2 (frames elapsed since last seen) to $\ell(o)$ as $\tau' = t_2/\ell(o)$. If τ' exceeds τ, the algorithm retains $\ell(o)$ as the initial segment interval for that object; otherwise, t_2 is selected. To simplify the notation, the initial segment interval is uniformly denoted as t_{init}.

Algorithm 2 Video Segmentation Algorithm

Require: Video V, σ, τ, b'
Ensure: Video segments = $[(s_1, e_1), ..., (s_k, e_k)]$
$O_{\text{all}} \leftarrow$ Detect&Track(V), start $\leftarrow 0$, Segments $\leftarrow \varnothing$
$t_{\text{init}} \leftarrow \max\{\ell(o) \mid o \in O_{\text{all}}, p(o) = \ell(o)/|V| \le \sigma \text{ or } t_2/\ell(o) > \tau\}$
while start $< |V|$ **do**
  $O' \leftarrow$ ExtractObjects(O_{all}, start, start + t_{init})
  $T' \leftarrow \{t'(o) \mid o \in O'\}$, $H(T') \leftarrow$ HistogramStats(T')
  $b_{\max} \leftarrow \arg\max_{b_k} f_k$ from $H(T')$
  $t_{\text{new}} \leftarrow \max(b_{\max})$, $t_{\text{init}} \leftarrow t_{\text{new}} > b' ? t_{\text{init}} \times 2 : t_{\text{new}}$
  end $\leftarrow \min(\text{start} + t_{\text{init}}, |V|)$
  Segments.append((start, end)), start $\leftarrow$ end
end while
return Segments

Dynamic Refinement. The initial guidance interval t_{init} is used for subsequent segmentation. We identify all objects within this interval (denoted as $O' = \{o'_1, o'_2, \ldots, o'_n\}$) and their durations ($T' = \{t'_1, t'_2, \ldots, t'_n\}$), then perform histogram statistics [37] to obtain the duration distribution:

$$H(T') = \{(b_k, f_k)\}_{k=1}^{K} \tag{2}$$

where b_k denotes the k-th time interval, f_k denotes the frequency in that interval, and K is the total number of time intervals. We select the interval with maximum frequency as:

$$b_{\max} = \arg\max_{b_k} f_k \tag{3}$$

If $b_{\max}$ exceeds the designated maximum interval b', we double the guidance interval and repeat the process until a qualified $b_{\max}$ is obtained. The new guidance interval is set to the maximum value of $b_{\max}$:

$$t_{\text{new}} = \max(b_{\max}) \tag{4}$$

This iteration continues until the full video is segmented into several video segments. This algorithm maximizes semantic integrity and avoids the rigidity of fixed-duration segmentation, serving as the foundation to generate VSUs.

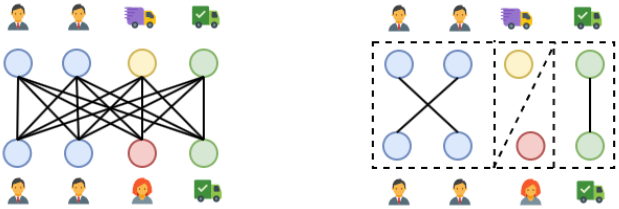
6 INDEX FRAMEWORK

This section introduces index framework by similarity metrics (Section 6.1), the construction of the hybrid index (Section 6.2), and the query execution (Section 6.3).

6.1 Similarity Metrics

The key to constructing the hybrid index is to cluster semantically similar VSUs to reduce the search space for queries. To achieve this objective, it is essential to accurately quantify the similarity among different VSUs. In our system, we establish distinct mechanisms for quantifying similarity on $\mathcal{F}$, $\mathcal{I}$, and $\mathcal{T}$, respectively: *Pairwise Matching-Trajectory Set Mover's Distance* **(PM-TSMD)**, *Interaction Distribution Distance* **(IDD)**, and *Coincidence Timing Mover's Distance* **(CTMD)**.

PM-TSMD. To quantify the similarity between two individual trajectories, we first propose the concept of *Trajectory Mover's Distance* (TMD) [8]. Suppose that two trajectories F_A and F_B are represented as feature vectors f_A and f_B respectively, and we employ cosine similarity [24] as their distance metric. The calculation of



(a) Original TSMD (b) Optimized PM-TSMD

Figure 2: Illustration of TSMD and PM-TSMD.

TMD is shown in the formula:

$$\mathrm{TMD}(F_A, F_B) = 1 - \frac{f_A \cdot f_B}{\|f_A\| \cdot \|f_B\|} \tag{5}$$

This leads to Trajectory Set Mover's Distance (TSMD), designed to quantify the overall similarity of trajectory distributions in two segments. TSMD first represents the trajectory set of each segment as a multidimensional distribution, then computes similarity via the average TMD value of all possible trajectory pairs. In contrast to TMD, TSMD captures both the local matching relationships among individual trajectories and the global distribution characteristics of the entire trajectory set, enabling a more effective measure of global similarities between trajectory distributions.

Let A and B represent two video segments, which contain m and n objects characterized by their trajectory feature vectors F_i^A and F_j^B, respectively. TSMD can be formalized as:

$$\mathrm{TSMD}(A, B) = \frac{1}{n \cdot m} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathrm{TMD}(F_i^A, F_j^B) \tag{6}$$

Assume each trajectory feature vector has dimensionality d. TSMD computes TMD for all $m \times n$ trajectory pairs, each with $O(d)$ complexity. Thus, its overall complexity is $O(m \cdot n \cdot d)$. Nonetheless, TSMD computation becomes a significant bottleneck in large-scale video datasets. When segments have numerous trajectories ($m \approx n \gg 1$), per-pair complexity is approximately $O(n^2 \cdot d)$. For datasets with t segments, the total complexity escalates to $O(t^2 \cdot n^2 \cdot d)$, which can reach $O(n^4 \cdot d)$ for a large t, making computation infeasible.

To address this issue, we introduce an optimized metric called PM-TSMD. Its core design reduces complexity by leveraging object class and spatial position information, eliminating unnecessary computations. PM-TSMD first groups trajectories in $\mathcal{F}$ by object class, splitting them into subsets and removing invalid cross-class computations. Within each subset, spatial position information is incorporated by pairing trajectories by their start and end positions. When defining matching priority, we assign equal weights ($w_1 = w_2 = 0.5$) to the start position distance d_{start}, and end position distance d_{end}. Each trajectory is matched at most once, and unmatched trajectories are penalized with a constant C. The detailed calculation formula for PM-TSMD is as follows:

$$\text{PM-TSMD}(A, B) = \frac{1}{\sum_{c=1}^{k} (|A_c| + |B_c|)} \left(\sum_{c=1}^{k} \mathrm{TSMD}(A_c, B_c) + \sum_{T \in \mathrm{UM}} C \right) \tag{7}$$

where k denotes the number of class subsets, A_c and B_c are the trajectory subsets of class c from video segments A and B respectively, and UM stands for the set of unmatched trajectories.

Figure 2 illustrates the computation process of TSMD and PM-TSMD. We now analyze the time complexity of PM-TSMD. In the ideal case (balanced trajectory distribution across k class), each class contains approximately $\frac{m}{k}$ and $\frac{n}{k}$ trajectories from segments A and B, reducing intra-category trajectory pairs to $\sum_{c=1}^{k} \left(\frac{m}{k} \cdot \frac{n}{k} \right) = \frac{m \cdot n}{k}$. Combined with the $O(\min(m, n) \cdot d)$ cost of location pairing, the total complexity is $O\left(\frac{m \cdot n}{k} + \min(m, n) \cdot d \right)$. Even in the worst case (i.e., $k = 1$, all objects in one class), the complexity simplifies to $O(m \cdot n + \min(m, n) \cdot d)$. For large-scale datasets, PM-TSMD's ideal complexity becomes $O\left(\frac{n^2}{k} + n \cdot d \right)$ per segment pair, with the total complexity across t segments reaching $O\left(t^2 \cdot \frac{n^2}{k} + t^2 \cdot n \cdot d \right)$; its worst-case complexity is $O\left(n^2 + n \cdot d \right)$ per segment pair, with total complexity across t segments at $O\left(t^2 \cdot n^2 + t^2 \cdot n \cdot d \right)$. In both scenarios, PM-TSMD incurs markedly lower computational cost than TSMD, substantially accelerating trajectory-set similarity computation and ensuring scalability for large-scale video datasets.

IDD. To effectively measure the similarity of high-level semantics between two VSUs, we propose the IDD, which captures the resemblance between object interaction patterns in each VSU. The respective $\mathcal{I}$ are converted into probability distribution vectors, $V_1 = [p_1, p_2, \ldots, p_n]$ and $V_2 = [q_1, q_2, \ldots, q_n]$, and their difference is evaluated using the Manhattan distance [30] as follows:

$$\mathrm{IDD}(V_1, V_2) = \|V_1 - V_2\|_1 \tag{8}$$

The total time complexity of IDD is $O(n)$, and its linear growth allows efficient and reliable evaluation of behavioral-pattern similarity among video segments, making it particularly suitable for large-scale applications.

CTMD. To accurately distinguish different events (Section 5.1), we introduce CTMD inspired by the Earth Mover's Distance (EMD) [16]. EMD is calculated by treating trajectories as distributions and calculating the minimum cost required to "carry" one trajectory to another. Let $P = \{p_1, p_2, \ldots, p_m\}$ and $Q = \{q_1, q_2, \ldots, q_n\}$ be two distributions where m and n denote the number of elements in distributions P and Q respectively. The EMD is calculated as:

$$\mathrm{EMD}(P, Q) = \min_{F} \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} \cdot d(p_i, q_j) \tag{9}$$

where F denotes the optimal handling and allocation plan and $d(p_i, q_j)$ denotes the distance metric.

Suppose that $\mathcal{T}_A$ and $\mathcal{T}_B$ are two temporal matrices. If their rows are unequal, we first pad the one with fewer rows to maintain consistency of rows. Let the maximum number of rows be n, which is the row count of the larger temporal matrix. We pad 0 for the first column and 1 for the second in the new rows as penalty values. The calculation of CTMD is shown as follows:

$$\mathrm{CTMD}(\mathcal{T}_A, \mathcal{T}_B) = L \cdot \frac{\mathrm{EMD}(D_{u1}, D_{u2}) + \mathrm{EMD}(D_{v1}, D_{v2})}{2} \tag{10}$$

where D_{u1} and D_{v1} denote the first and second columns of $\mathcal{T}_A$ and D_{u2} and D_{v2} denote the first and second columns of $\mathcal{T}_B$ respectively. L is a coefficient calculated as the ratio of coincident index mappings (from two EMD computations) to row count n.

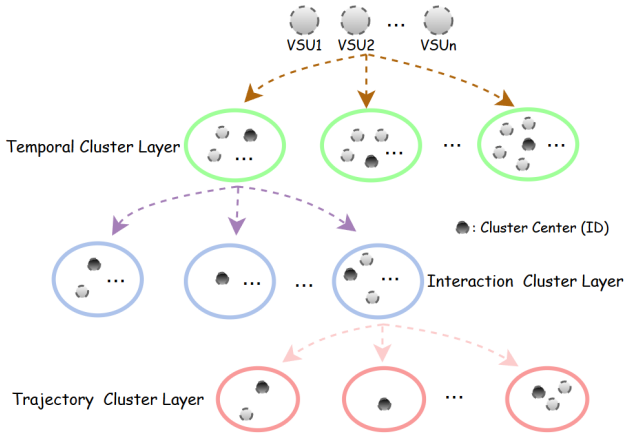


Figure 3: Cluster Index Layer Architecture.

The total time complexity of CTMD is dominated by two EMD computations. Instead of solving exact EMD with $O(n^3)$ complexity, we use entropic-regularized OT and compute EMD approximately with Sinkhorn iterations [12, 27], which costs $O(n^2)$ per iteration for dense cost matrices. Thus, the practical complexity of CTMD is $\widetilde{O}(n^2)$. The overhead is substantially reduced while preserving event distinction and matching performance.

6.2 Hybrid Index Framework

For rapid query execution, we develop a hybrid index framework, achieving efficient candidate filtering [36] for multi-level quires. This framework contains two layers: the inverted index layer is mainly used for existence and predefined event-based dynamic queries, while the cluster index layer [40] for similarity queries.

Inverted Index Layer. In order to quickly locate video segments containing specific information, we use the inverted index as a preliminary filtering tool, narrowing down the VSU candidate set [2]. The inverted index comprises two primary components: a keyword dictionary and an inverted list. The keyword dictionary serves as a repository for all object classes and interaction types, each associated with an inverted list that enumerates the VSU segments where the keyword appears.

Cluster Index Layer. To accelerate similarity queries, we establish a cluster index layer to refine the organization of VSUs [52]. Although we propose three distinct similarity metrics (Section 6.1), if directly clustering all VSUs, especially on large-scale datasets, there will still be significant computational overhead in clustering and low query efficiency. Therefore, we implement a tiered cluster index layer architecture, segmented into three distinct layers and illustrated in Figure 3.

- Temporal ($\mathcal{T}$) cluster layer: Given the temporal distribution from $\mathcal{T}$ matrix, all VSUs are classified via their temporal distribution to distinguish different events. This generates several temporal clusters. Each cluster has a representative temporal distribution center, reflecting the typical temporal pattern of all VSUs within the cluster. The center is selected by identifying a specific VSU in the cluster. This VSU has the minimum average CTMD to all other VSUs in the same cluster. Its corresponding $\mathcal{T}$ matrix serves as the representative center.
- Interaction ($\mathcal{I}$) cluster layer: Further clustering is performed based on the interaction relationships within each temporal cluster, gaining various interaction clusters. Each cluster is characterized by a representative interaction distribution center that represents the high-level semantics.
- Trajectory ($\mathcal{F}$) cluster layer: The tertiary cluster layer refines the grouping based on trajectory features, following the construction procedures outlined above. This layer is utilized to identify and match specific trajectory patterns.

The construction of each cluster layer follows a systematic approach and is constructed once without re-clustering. We employ the hierarchical clustering algorithm [15] to organize data hierarchically. To clarify the internal data structure, we use the interaction cluster layer as an example. Its structure consists of the following components: VSU_{list}, a list containing all VSUs within the cluster; $Cluster_{\text{id}}$, a unique identifier assigned to the cluster; $Center_{\text{interaction}}$, denoting the central point of interactions within the cluster; and $Trajectory_{\text{cluster}_{\text{id}}}$, a list containing the identifiers of internal trajectory clusters.

We further discuss how to select the number of clusters at each level. The choice is primarily determined by object density. For low-density scenarios (e.g., sparse highway traffic), fewer clusters suffice for the $\mathcal{T}$ layer, as temporal dynamics are simple, lacking diverse occurrence patterns. The $\mathcal{I}$ layer also requires fewer clusters, since limited object pairs lack diverse interaction patterns. In contrast, the $\mathcal{F}$ layer needs a moderately higher number. Although most trajectories are regular, they still exhibit variations, such as straight lines and turns.

6.3 Query Execution

We implement query execution on a dual-database architecture. Specifically, a relational database stores video segment metadata, while a vector database preserves VSU vectors—both linked via unique identifiers. This design leverages separate strengths: the relational database's proficiency in high-capacity metadata management, and the vector database's efficiency in similarity computation. Under this architecture, the inverted index supports existence and predefined event-based dynamic queries (*Query B2*) and also serves as the first-stage filter for trajectory-based dynamic queries (*Query A1*), after which candidate VSUs are verified by trajectory-level matching on $\mathcal{F}$. Similarity queries (*Query C3*) are executed on the cluster index layer, where each layer ranks candidates by metrics [14] and retains only Top-k results until the final stage. This staged pruning reduces pairwise comparisons and keeps query latency low while preserving accuracy.

7 EVALUATION

We evaluate the accuracy, latency, and scalability of Craw through the following research questions (RQs):

- **RQ1** - How does Craw perform compared with existing systems for query execution?
- **RQ2** - How does the video segmentation algorithm proposed by Craw perform optimally?

- **RQ3** - How much does PM-TSMD improve computational efficiency?
- **RQ4** - To what extent does Craw enhance similarity query accuracy and efficiency relative to Video-zilla?
- **RQ5** - How do the VLMs perform for multi-level queries?

7.1 Experimental Setup

All experiments are conducted on a computing server equipped with a single NVIDIA GeForce RTX 4090, paired with an AMD EPYC 7773X processor comprising 128 cores and 512 GB of system memory. The operating system is Ubuntu 22.04 with kernel version 6.5.0-18-generic. We use the GPU to pretrain the YOLO, Recurrent Neural Network (RNN) and Graph Neural Network (GNN), and then perform YOLO with BOT-SORT for multi-object tracking on each dataset. To ensure tracking accuracy, we do not perform frame skipping operations. We only use the similarity module of Video-zilla rather than the whole system. The vector database is implemented using Milvus [41]. We operate the VLM via its official API.

Datasets. We evaluate Craw using the following datasets:

- **Highway [33]:** A dataset captured on a highway by a fixed surveillance camera over 147 seconds, featuring numerous fast-moving, mostly linear trajectories of vehicles.
- **Resort [6]:** A dataset containing 60 hours of video from a fixed camera monitoring a pedestrian walkway outside a resort hotel. We selected six subsets of 51 minutes each. The scene contains relatively few, slow, and occasionally stationary objects with long trajectories.
- **Bellevue [11]:** A dataset recorded from a crossroad camera in Bellevue, using a 52-minute video segment. The scene presents a moderate number of objects with medium speeds and balanced scene dynamics.
- **Warsaw [6]:** A dataset obtained from a traffic junction, from which we selected a 51-minute video segment. The scene features many diverse objects moving at moderate speeds along complex, frequently turning trajectories.

Baselines. To measure the performance of multi-level queries, we compare Craw against baselines that cover the main design choices relevant to Craw in Section 2. For existence and dynamic queries, we choose FIGO [9], MIRIS [6] and EQUI-VOCAL [49], which represent fine-grained, trajectory-based and predefined event-based queries respectively. For similarity queries, we choose Video-zilla [18], as Video-zilla is also via semantic units and uses the cluster index. For multi-level queries, we use Qwen3-VL [3] that is theoretically capable but incurs high overhead. For video segmentation, we select four conventional baselines widely used in prior work: fixed interval [49], temporal difference [18], background subtraction [5], and optical flow [46]. The fixed interval is set to the average object occurrence time, while temporal difference, background subtraction and optical flow all take the average pixel difference between the top 20 percent of video frames as their segmentation threshold. Finally, we further evaluate the efficiency improvement of PM-TSMD against the original TSMD.

Experiment Query. To reflect practical application requirements, we select example existence and dynamic queries based on the following principles. First, existence queries cover both basic existence (e.g., vans and cars exist) and object counting (e.g., more than four cars exist). Second, dynamic queries include both predefined events (e.g., people side by side) and spatio-temporal dynamics (e.g., vehicles braking). These two categories aim to validate the effectiveness and efficiency of the inverted index and VSU. The specific query examples are shown in Table 2 and 3. For similarity queries, we take integral events as units (i.e., complete VSUs) and randomly sample partial segments from VSUs as query inputs to evaluate the efficiency and robustness of the cluster index layer.

Table 2: Example Existence Queries for Different Datasets

Dataset	Query ID	Query Description
Highway	EQ1.1	More than five cars exist.
	EQ1.2	Vans and cars exist.
Resort	EQ2.1	More than three people exist.
	EQ2.2	People side by side and trucks exist.
Bellevue	EQ3.1	Buses and cars exist.
	EQ3.2	Four cars and people exist.
Warsaw	EQ4.1	Two trams and one bus exist.
	EQ4.2	More than four cars exist.

Table 3: Example Dynamic Queries for Different Datasets

Dataset	Query ID	Query Description
Highway	DQ1.1	Cars move more than 400 pixels in 2 seconds.
	DQ1.2	Vans or cars move away from the screen.
Resort	DQ2.1	People walk more than 100 pixels in 1 second.
	DQ2.2	People stop more than 5 seconds.
Bellevue	DQ3.1	Cars go straight at the traffic light.
	DQ3.2	People cross the road.
Warsaw	DQ4.1	Trams or cars brake.
	DQ4.2	Cars turn a corner.

Metrics. We employ the F1 score and query execution time as metrics to evaluate the performance of existence and dynamic queries. For the similarity query, we compare accuracy, query execution time, and preprocessing time. We also leverage Accuracy for multi-level queries (VLMs). All queries are manually checked within each dataset. To evaluate the performance of video segmentation, we adopt DIP (Section 5.1): the segmentation is considered to satisfy the completeness condition if and only if DIP $\geq x\%$, where $x\%$ is the preset threshold for semantic integrity.

7.2 Existence and Dynamic Query Performance

Accuracy Analysis. We evaluate Craw against baselines on different queries shown in Tables 2 and 3. As illustrated in Figure 4, Craw achieves the highest F1 score on the vast majority of query tasks. Its relatively high accuracy stems from using video segment units rather than individual frames, thereby reducing errors caused by tracker inaccuracies. However, Craw performs poorly on some

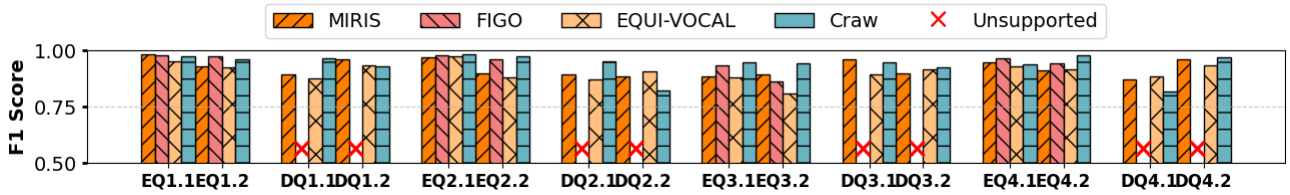


Figure 4: F1 Scores of Craw against baselines for existence and dynamic queries (EQ1.1–DQ4.2)

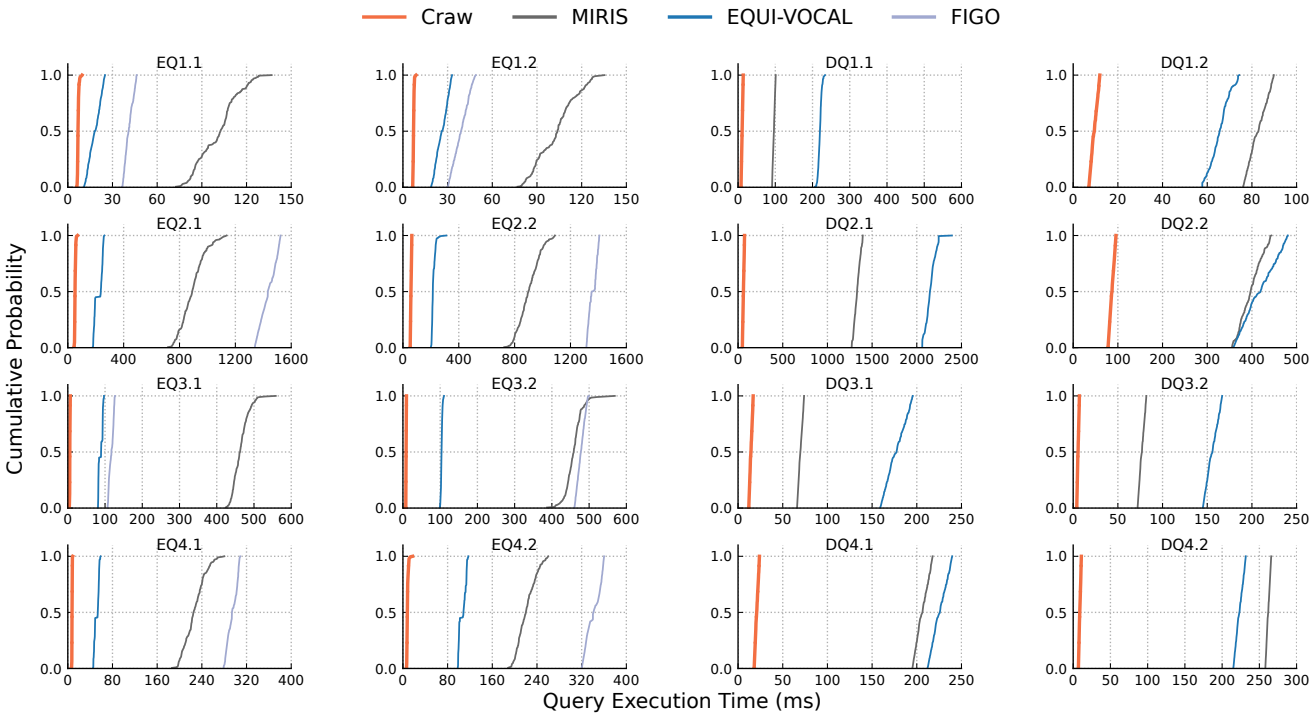


Figure 5: Cumulative distributions of query execution times for eight existence queries (EQ1.1–EQ4.2) and eight dynamic queries (DQ1.1-DQ4.2), comparing Craw with MIRIS, EQUI-VOCAL, and FIGO. The x-axis scales differ between subplots to accommodate the varying ranges of query execution times.

dynamic queries. This is because we simplified the trajectories during processing. Here, we can only roughly judge correctness by setting a threshold, which results in a relatively low F1 score in these cases. MIRIS performs well on dynamic queries, but its performance on existence queries is less outstanding compared to other methods. This leads to certain accuracy loss, as EQUI-VOCAL and FIGO rely on object detection while MIRIS utilizes object tracking. EQUI-VOCAL is optimized for compositional or synthetic queries but demonstrates instability on instance-level or motion-sensitive predicates. FIGO is tailored for existence queries and tends to yield higher accuracy than MIRIS and EQUI-VOCAL in static contexts. However, it lacks support for dynamic queries.

Efficiency Analysis. The cumulative distribution functions (CDFs) in Figure 5 demonstrate that Craw's curve consistently lies furthest left and remains steepest across all subplots and datasets, indicating strictly shorter execution times, lower median and tail latencies, and reduced variance compared to baselines. This reflects more predictable latency and faster candidate retrieval enabled by its inverted index with verification. Results highlight two core design choices that collectively compress query response time by one to two orders of magnitude. First, the inverted index accelerates existence and predefined event-based queries via direct lookups: simple predicates such as "cars exist" are resolved by probing in-memory postings lists in microseconds, and even multi-term predicates like "cars and vans exist" reduce to fast list intersections. This lookup-driven execution avoids the costly data traversal and full scans that dominate the runtime of baselines. Second, the staged query execution pipeline localizes the dominant cost in a small candidate set. Craw first segments each video into VSUs, replacing the tens of thousands of individual frames that baselines must scan and shrinking the search space by two to three orders of magnitude. Within this reduced space, query execution decomposes into two

Table 4: Top-k Similarity Retrieval Results for Craw and Video-zilla. "k-Avg" denotes Top-k Average. Metrics: (i) Hit(%) – fraction of queries retrieving at least one relevant segment; (ii) k-Avg – mean rank of the correct hit; (iii) Time (ms) – average query latency (excluding I/O). Note: "10–100" represents the duration proportion of the query video segment relative to the original video segment (e.g., "10" indicates randomly selecting 10 consecutive seconds as the query segment from a 100-second video segment).

Highway

Method	Metric	10	20	30	40	50	60	70	80	90	100
Craw	Hit (%)	4.8	7.1	23.8	38.1	57.1	**71.4**	78.6	88.1	95.2	100
	k-Avg	3.00	2.67	2.20	2.00	2.21	1.87	1.73	1.41	1.15	1.00
	Time (ms)	11	14	21	28	33	**38**	52	61	65	**68**
Video-zilla	Hit (%)	50.0	57.1	54.8	66.7	61.9	**71.4**	78.6	73.8	78.6	92.9
	k-Avg	2.19	2.54	2.04	2.18	2.15	1.90	1.79	2.13	1.58	1.00
	Time (ms)	766	890	913	923	929	**877**	935	938	963	**1002**

Resort

Method	Metric	10	20	30	40	50	60	70	80	90	100
Craw	Hit (%)	9.4	12.3	26.4	38.2	57.6	**76.2**	85.2	92.8	98.7	100
	k-Avg	3.22	2.87	2.42	2.03	1.88	1.72	1.38	1.21	1.05	1.00
	Time (ms)	38	42	46	67	83	**103**	126	144	163	**182**
Video-zilla	Hit (%)	39.3	59.3	66.7	72.6	72.7	73.9	82.9	88.6	97.1	100
	k-Avg	1.18	1.66	1.67	1.24	1.25	1.16	1.24	1.27	1.18	1.00
	Time (ms)	2394	2659	2741	2759	2778	**2809**	2854	2892	2911	**2930**

Bellevue

Method	Metric	10	20	30	40	50	60	70	80	90	100
Craw	Hit (%)	6.3	16.5	21.5	35.8	56.3	74.2	83.7	93.8	98.2	100
	k-Avg	2.92	2.73	2.48	1.94	1.75	1.48	1.34	1.16	1.03	1.00
	Time (ms)	27	33	41	53	75	86	92	106	118	**123**
Video-zilla	Hit (%)	35.9	44.9	59.4	76.4	76.6	86.9	90.7	97.2	99.1	100
	k-Avg	1.32	1.90	1.67	1.52	1.29	1.20	1.10	1.03	1.02	1.00
	Time (ms)	2710	2875	3210	3181	3218	3237	3188	3258	3210	**4174**

Warsaw

Method	Metric	10	20	30	40	50	60	70	80	90	100
Craw	Hit (%)	14.6	25.3	32.8	47.7	60.5	73.4	81.5	92.4	97.2	100
	k-Avg	2.58	2.36	2.14	1.86	1.78	1.56	1.34	1.12	1.06	1.00
	Time (ms)	23	29	38	49	57	63	72	78	86	**91**
Video-zilla	Hit (%)	48.3	59.6	60.0	63.1	75.6	77.7	88.3	90.4	95.8	100
	k-Avg	1.10	1.11	1.31	1.21	1.35	1.27	1.22	1.02	1.09	1.00
	Time (ms)	944	1390	1596	1860	1904	1898	1861	1977	1977	**2005**

clearly separated stages: an indexed probe returns candidate VSUs in microseconds, and the heavier verification is invoked only on the resulting candidates in milliseconds. This reduces the search space by two to three orders of magnitude, directly shortening the time needed to locate relevant data. In contrast, baselines operate on tens of thousands of individual frames and pay the full I/O and parsing overhead for every query.

7.3 Similarity Query Performance

Accuracy. As shown in Table 4, we compare the performance of Craw and Video-zilla across different k values and observe that Craw achieves monotonic gains in Hit with k, reaching 100% by k=100 on all datasets. While Video-zilla performs better for small k, Craw quickly converges to comparable accuracy for larger k. Its k-Avg approaches 1.00 at large k, indicating effective ranking once relevant segments are retrieved. At small k, Craw exhibits low Hit, since the query segment only covers a small portion of the original VSU, which may miss part of an event (Section 5.1) and makes semantic integrity less precise. This incompleteness first impacts $\mathcal{F}$: some objects' trajectories are truncated, shifting their start/end points and potentially missing critical trajectory information (e.g., turns). This truncated $\mathcal{F}$ further propagates to $\mathcal{T}$ and $\mathcal{I}$: $\mathcal{T}$ becomes shifted after normalization, thereby changing the duration, while $\mathcal{I}$ may change because the co-occurrence duration is shortened or even absent, altering the interaction patterns. For example, consider a 50 seconds segment where two people intersect at 20 seconds and disappear at 30 seconds. If a short query segment (e.g., k = 10) misses the intersection moment, $\mathcal{F}$, $\mathcal{I}$ and $\mathcal{T}$ can deviate significantly from the complete VSU, leading to a sharp drop in hit rate. In contrast, Video-zilla captures the object distribution characteristics, while our video segmentation algorithm exceedingly ensures the integrity and stability of the object distribution. Thus, although in the case of small k, Video-zilla reflects a much higher Hit than Craw.

Efficiency. We observe that Craw consistently achieves one to two orders of magnitude faster query execution than Video-zilla. At k = 100, Craw exhibits latencies of 68 ms for Highway, 182 ms for Resort, 123 ms for Bellevue, and 91 ms for Warsaw, whereas Video-zilla requires 1002 ms, 2930 ms, 4174 ms, and 2005 ms, respectively. This corresponds to a 14x–34x speedup. This significant acceleration primarily stems from two key factors: feature dimensionality and clustering architecture. For Craw, trajectory similarity takes up the highest computational calculation. However, we substantially reduce the feature dimension from hundreds or thousands of dimensions down to only tens of bits, achieving a reduction of one order of magnitude. In contrast, Video-zilla is constrained by high-dimensional features. Even after dimensionality reduction to 256 dimensions while preserving accuracy, Video-zilla still maintains a feature dimension gap of at least 4x compared with Craw. Furthermore, Craw's three-layer clustering structure reduces the

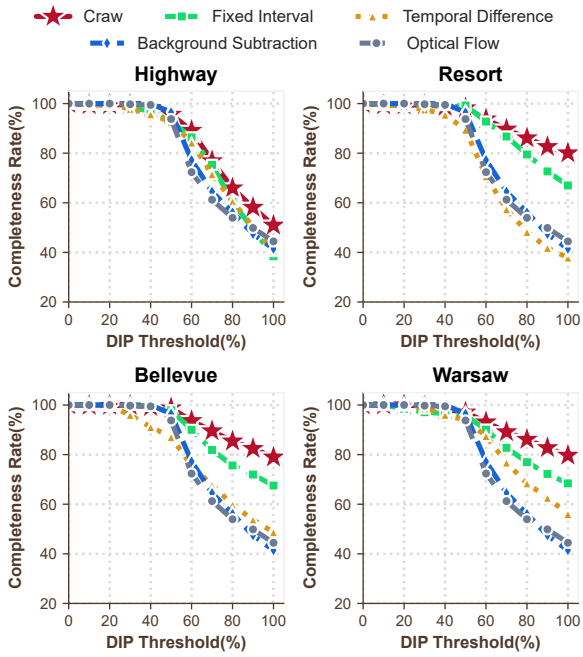


Figure 6: Proportion of objects satisfying DIP $\geq x$ under varying DIP thresholds x on four datasets. Higher curves correspond to better segmentation completeness.

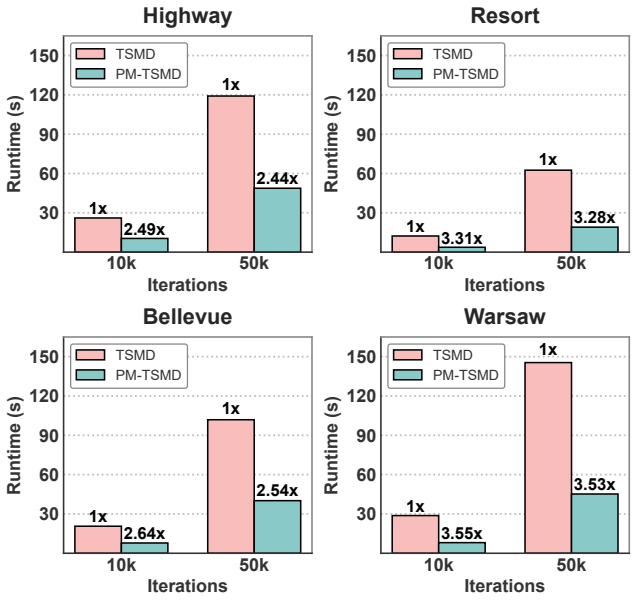


Figure 7: TSMD vs. PM-TSMD runtime and speedup.

number of intra-layer traversals relative to Video-zilla's single-layer clustering, further boosting query efficiency.

7.4 Segmentation Optimality

Result Analysis. As illustrated in Figure 6, Craw demonstrates superior performance across all datasets and DIP thresholds compared to baselines. As x increases, Craw exhibits a clear performance edge: the performance difference is negligible when $x \leq 40\%$, while other methods degrade rapidly beyond $x \geq 70\%$, and Craw maintains high performance. Since DIP measures how a segmentation preserves continuous object durations to maintain semantic integrity, this indicates that Craw's boundaries align better with event transitions, thereby reducing trajectory fragmentation. For $x \geq 70\%$, Temporal Difference, Background Subtraction, and Optical Flow drop sharply across all datasets, due to their sensitivity to subtle background changes and viewpoint-dependent motion magnitudes, which makes boundary detection unstable. Furthermore, the distance between the target object and the camera influences pixel level difference magnitudes, resulting in unstable segmentation outcomes. Fixed interval exhibits markedly better performance than the other three baselines across three datasets, where objects move at a steady pace. This is due to our utilization of the average duration for which an object appears within a scene as an interval rather than employing arbitrary fixed values such as 5 seconds or 10 seconds that may not adapt well to different datasets. Notably, on Highway where objects move rapidly, we observe that as x increases, there is a rapid decrease in performance, as fast-moving objects are easily split by a fixed interval.

7.5 PM-TSMD Efficiency

Speedup Analysis. To analyze the speedup ratio between TSMD and PM-TSMD, we randomly selected VSUs for calculation while ensuring consistent selection methods. Results are presented in Figure 7. Across all datasets, PM-TSMD achieves a speedup of 2.4x to 3.6x over TSMD, with acceleration mainly influenced by object count, category number, and trajectory point count. The third exerts the most significant impact, as more objects, richer categories, or longer trajectories can all enhance acceleration by reducing redundant pairwise computations through category-wise pruning and geometric pairing. Specifically, Highway contains many objects but has a single category and linear motion trajectories, which limits pruning opportunities and results in the lowest speedup of approximately 2.4x. In contrast, Warsaw is the most complex scene, with dense multiple categories and dynamic content, allowing it to fully leverage optimization effects and achieve the highest speedup of approximately 3.55x. Although Resort is smaller in scale than Highway, it delivers a speedup of approximately 3.3x. This may be because its longer trajectories or stationary trajectory states enhance the efficiency of geometric pairing, a feature lacking in Highway's linear motion. Bellevue lies between these two extremes, with balanced category diversity and moderate trajectory complexity, leading to an intermediate speedup of approximately 2.6x.

Scalability Analysis. We further evaluated the scalability of PM-TSMD through experiments involving 10k and 50k random trajectory computations, covering medium to large data volumes. The results demonstrate that the speedup remains nearly constant as the scale of computations increases, confirming PM-TSMD's strong scalability. This characteristic enables PM-TSMD to sustain consistent acceleration performance across large-scale datasets. Notably, this scalability is achieved without compromising accuracy.

Table 5: Module Time Distribution Comparison for Craw and Video-zilla

	Phase	Highway	Resort	Warsaw	Bellevue
Video-zilla	Processing (s)	273.53	4522.16	882.35	2884.99
	Indexing (s)	2.68	52.96	16.88	14.36
	Total (s)	276.52	5270.04	899.23	2899.35
Craw	Processing (s)	221.65	5652.19	881.37	3370.02
	Indexing (s)	1.26	10.57	2.75	5.13
	Total (s)	222.90	5662.26	884.09	3375.12

7.6 Preprocessing Cost

Module Time Distribution. We analyze the module time distribution of Craw and Video-zilla across two phases: video processing and index construction. As shown in Table 5, indexing is short relative to processing on both systems. For example, on Highway, Craw spends only 1.26s on indexing out of 222.90s total, and Video-zilla spends 2.68s out of 276.52s. For Craw, object tracking accounts for the majority; for Video-zilla, the cost splits between object detection and feature extraction, which scales with the number of segments. Although object tracking is 15-20% more expensive than detection per frame, Craw avoids feature extraction, which explains why Craw is faster on Highway/Warsaw but slower on Resort/Bellevue.

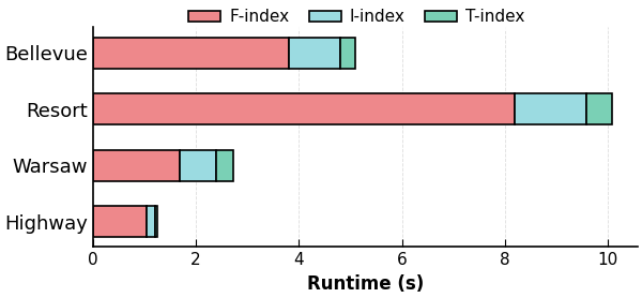


Figure 8: Build Time for Each Cluster Index Layer

Layer-wise Build Time. To further demonstrate the contribution of each layer, Figure 8 shows the build time of each layer. Results show that the temporal and interaction layers require significantly less time compared to the trajectory layer, accounting for only approximately one-sixth to one-fifth of the total build time. Notably, despite their low computational cost, prior experimental results (Section 7.3) clearly demonstrate the critical role of these two layers—filtering substantial dissimilar candidate segments, indicating that our system achieves substantial contributions without incurring significant time overhead.

7.7 VLM Discussion for Multi-level Queries

We further explore the application of Qwen3-VL (VLM) in multi-level queries. Due to the token limit of VLMs and the model's inherent constraints, we only conduct small-scale experiments on the Highway dataset. Instead of extracting all frames, we sample the first frame per second as key frames for evaluation. The results are summarized as follows.

Table 6: Existence Query Performance for Qwen3-VL

Model	Dataset	Accuracy	Runtime (s)
Qwen3-VL	Highway	0.80	25.4

Existence Query. We chose EQ1.1, a more challenging query case, as the example. As shown in Table 6, Qwen3-VL achieves a moderate accuracy of 0.80, but its query latency reaches 25.4 seconds, far exceeding the real-time query requirements. This reflects the high computational overhead of VLMs in frame-level semantic parsing, a problem that structured systems like Craw avoid via lightweight semantic representation.

Dynamic Query. We select both DQ1.1 and DQ1.2 for dynamic queries. Experimental results show that Qwen3-VL fails to locate specific cars or vans, only returning candidate counts. Notably, the count is high initially but drops sharply later, likely due to token limit-induced performance decay. The final returned count is a small proportion of the ground truth, demonstrating that existing VLMs struggle to handle fine-grained dynamic queries.

Table 7: Similarity Query Performance for Qwen3-VL

Model	Dataset	Hit (%)	Ave_Runtime (s)
Qwen3-VL	Highway	90.5	272.3

Similarity Query. We employ complete VSUs (corresponding to $k = 100$ in Table 4) for similarity queries, which can be regarded as video segment re-identification. As presented in Table 7, Qwen3-VL achieves a relatively high Hit of 90.5%. However, its average runtime of 272.3 seconds is impractical for large-scale deployment.

8 CONCLUSION

In this paper, we present Craw, a novel framework enabling efficient multi-level queries while ensuring semantic integrity and low overhead for large-scale video datasets. Specifically, we design the VSU for fine-grained representation, a semantic-preserving segmentation algorithm for VSU quality, and a hybrid index framework for efficiency. With trajectory simplification and optimized similarity metrics, Craw overcomes traditional limitations, supporting multi-level queries with low latency, high accuracy, and scalability. Experimental results show Craw achieves up to two orders of magnitude lower latency than SOTA. In the future, we plan to expand the query scope by multi-modal (e.g., kiss), as Craw only supports spatio-temporal relationships.

ACKNOWLEDGMENTS

We sincerely thank the shepherd and the reviewers for their insightful comments. This research is sponsored by the National Natural Science Foundation of China (No.62572222). We are also grateful to Dr. Jinhui Lai from Nanchang University for his patient guidance during the revision process.

REFERENCES

[1] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. 2022. BoT-SORT: Robust Associations Multi-Pedestrian Tracking. arXiv:2206.14651 [cs.CV] https://arxiv.org/abs/2206.14651

[2] Artem Babenko and Victor Lempitsky. 2015. The Inverted Multi-Index. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37, 6 (2015), 1247–1260. https://doi.org/10.1109/TPAMI.2014.2361319

[3] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. 2025. Qwen3-VL Technical Report. *arXiv preprint arXiv:2511.21631* (2025).

[4] 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 (May 2023), 2289–2301. https://doi.org/10.14778/3598581.3598599

[5] Olivier Barnich and Marc Van Droogenbroeck. 2009. ViBE: A powerful random technique to estimate the background in video sequences. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. 945–948. https://doi.org/10.1109/ICASSP.2009.4959741

[6] Favyen Bastani, Songtao He, Arjun Balasingam, Karthik Gopalakrishnan, Mohammad Alizadeh, Hari Balakrishnan, Michael Cafarella, Tim Kraska, and Sam Madden. 2020. MIRIS: Fast Object Track Queries in Video. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 1907–1921. https://doi.org/10.1145/3318464.3389692

[7] Favyen Bastani and Samuel Madden. 2022. OTIF: Efficient Tracker Pre-processing over Large Video Datasets. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 2091–2104. https://doi.org/10.1145/3514221.3517835

[8] F. Boem, F.A. Pellegrino, G. Fenu, and T. Parisini. 2011. Trajectory clustering by means of Earth Mover's Distance. *IFAC Proceedings Volumes* 44, 1 (2011), 4741–4746. https://doi.org/10.3182/20110828-6-IT-1002.01230 18th IFAC World Congress.

[9] Jiashen Cao, Karan Sarkar, Ramyad Hadidi, Joy Arulraj, and Hyesoon Kim. 2022. FiGO: Fine-Grained Query Optimization in Video Analytics. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 559–572. https://doi.org/10.1145/3514221.3517857

[10] Tat-Seng Chua and Li-Qun Ruan. 1995. A video retrieval and sequencing system. *ACM Trans. Inf. Syst.* 13, 4 (Oct. 1995), 373–407. https://doi.org/10.1145/211430.211431

[11] City of Bellevue. 2017. Traffic Video Dataset. https://github.com/City-of-Bellevue/TrafficVideoDataset [Online; accessed: August 14, 2025].

[12] Marco Cuturi. 2013. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *Advances in Neural Information Processing Systems*, C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger (Eds.), Vol. 26. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2013/file/af21d0c97db2e27e13572cbf59eb343d-Paper.pdf

[13] Deep Lake. 2025. https://docs.deeplake.ai/4.0/ [Online; accessed: January 17, 2026].

[14] Yanbo Fan, Siwei Lyu, Yiming Ying, and Bao-Gang Hu. 2017. Learning with Average Top-k Loss. arXiv:1705.08826 [stat.ML] https://arxiv.org/abs/1705.08826

[15] Chen Feng, Yuichi Taguchi, and Vineet R. Kamat. 2014. Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 6218–6225. https://doi.org/10.1109/ICRA.2014.6907776

[16] Manal Al Ghamdi and Yoshihiko Gotoh. 2014. Video Clip Retrieval by Graph Matching. In *Advances in Information Retrieval*, Maarten de Rijke, Tom Kenter, Arjen P. de Vries, ChengXiang Zhai, Franciska de Jong, Kira Radinsky, and Katja Hofmann (Eds.). Springer International Publishing, Cham, 412–417.

[17] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Paramvir Bahl, Matthai Philipose, Phillip B. Gibbons, and Onur Mutlu. 2018. Focus: Querying Large Video Datasets with Low Latency and Low Cost. arXiv:1801.03493 [cs.DB] https://arxiv.org/abs/1801.03493

[18] B. Hu, P. Guo, and W. Hu. 2022. Video-zilla: An indexing layer for large-scale video analytics. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD)*. Association for Computing Machinery, New York, NY, USA, 1905–1919. https://doi.org/10.1145/3514221.3517840

[19] Weiming Hu, Nianhua Xie, Li Li, Xianglin Zeng, and Stephen Maybank. 2011. A Survey on Visual Content-Based Video Indexing and Retrieval. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 41, 6 (2011), 797–819. https://doi.org/10.1109/TSMCC.2011.2109710

[20] Jing Huang, S.R. Kumar, M. Mitra, Wei-Jing Zhu, and R. Zabih. 1997. Image indexing using color correlograms. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. San Juan, Puerto Rico, 762–768. https://doi.org/10.1109/CVPR.1997.609412

[21] Jinwoo Hwang, Minsu Kim, Daeun Kim, Seungho Nam, Yoonsung Kim, Dohee Kim, Hardik Sharma, and Jongse Park. 2022. CoVA: Exploiting Compressed-Domain Analysis to Accelerate Video Analytics. arXiv:2207.00588 [cs.CV] https://arxiv.org/abs/2207.00588

[22] D. Kang, P. Bailis, and M. Zaharia. 2019. Blazeit: Optimizing declarative aggregation and limit queries for neural network-based video analytics. *Proceedings of the VLDB Endowment* 13, 4 (2019), 533–546.

[23] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: optimizing neural network queries over video at scale. *Proc. VLDB Endow.* 10, 11 (Aug. 2017), 1586–1597. https://doi.org/10.14778/3137628.3137664

[24] Sukhdeep Kaur and Deepak Aggarwal. 2013. Image Content Based Retrieval System using Cosine Similarity for Skin Disease Images. *Advances in Computer Science : an International Journal* 2 (2013), 89–95. https://api.semanticscholar.org/CorpusID:15526374

[25] LanceDB. 2025. https://docs.lancedb.com/indexing [Online; accessed: January 17, 2026].

[26] Rainer Lienhart, Silvia Pfeiffer, and Wolfgang Effelsberg. 1997. Video abstracting. *Commun. ACM* 40, 12 (Dec. 1997), 54–62. https://doi.org/10.1145/265563.265572

[27] Tianyi Lin, Nhat Ho, and Michael I. Jordan. 2022. On the Efficiency of Entropic Regularized Algorithms for Optimal Transport. *Journal of Machine Learning Research* 23, 137 (2022), 1–42. http://jmlr.org/papers/v23/20-277.html

[28] Y. Liu, Y. Peng, H. Zhou, H. Liu, X. Lu, J. Lou, C. Wu, W. Zhao, and J. Li. 2025. LOVO: Efficient Complex Object Query in Large-Scale Video Datasets. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. 1938–1951. https://doi.org/10.1109/ICDE65448.2025.00148

[29] Cheng Long, Raymond Chi-Wing Wong, and H. V. Jagadish. 2014. Trajectory simplification: on minimizing the direction-based error. *Proc. VLDB Endow.* 8, 1 (Sept. 2014), 49–60. https://doi.org/10.14778/2735461.2735466

[30] M. D. Malkauthekar. 2013. Analysis of euclidean distance and Manhattan Distance measure in face recognition. In *Third International Conference on Computational Intelligence and Information Technology (CIIT 2013)*. 503–507. https://doi.org/10.1049/cp.2013.2636

[31] S. G. Mallat. 1989. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11, 7 (July 1989), 674–693. https://doi.org/10.1109/34.192463

[32] MarketsandMarkets. 2025. Video Surveillance Market Size, Share & Analysis, 2031. Report Code SE 2873. https://www.marketsandmarkets.com/Market-Reports/video-surveillance-market-645.html [Online; accessed: July 21, 2026].

[33] Nicholas Abraham-Raegan Martinez. 2018. 4K Video of Highway Traffic! YouTube video. https://www.youtube.com/watch?v=KBsqQez-O4w [Online; accessed: August 26, 2025.

[34] Carlos Plou, Cesar Borja, Ruben Martinez-Cantin, and Ana C. Murillo. 2025. FALCONEye: Finding Answers and Localizing Content in ONE-hour-long videos with multi-modal LLMs. arXiv:2503.19850 [cs.CV] https://arxiv.org/abs/2503.19850

[35] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, Faster, Stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 6517–6525. https://doi.org/10.1109/CVPR.2017.690

[36] Babak Rezaee. 2010. A cluster validity index for fuzzy clustering. *Fuzzy Sets and Systems* 161, 23 (2010), 3014–3025. https://doi.org/10.1016/j.fss.2010.07.005 Theme: Information processing.

[37] DAVID W. SCOTT. 1979. On optimal and data-based histograms. *Biometrika* 66, 3 (12 1979), 605–610. https://doi.org/10.1093/biomet/66.3.605 arXiv:https://academic.oup.com/biomet/article-pdf/66/3/605/632347/66-3-605.pdf

[38] Guolei Sun, Yun Liu, Henghui Ding, Thomas Probst, and Luc Van Gool. 2022. Coarse-to-Fine Feature Mining for Video Semantic Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. New Orleans, LA, USA, 3116–3127. https://doi.org/10.1109/CVPR52688.2022.00313

[39] Ba Tu Truong and Svetha Venkatesh. 2007. Video abstraction: A systematic review and classification. *ACM Trans. Multimedia Comput. Commun. Appl.* 3, 1 (Feb. 2007), 3–es. https://doi.org/10.1145/1198302.1198305

[40] Georgios Vardakas and Aristidis Likas. 2023. Global k-means++: an effective relaxation of the global k-means clustering algorithm. arXiv:2211.12271 [cs.LG] https://arxiv.org/abs/2211.12271

[41] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. 2021. Milvus: A

Purpose-Built Vector Data Management System. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA, 2614–2627. https://doi.org/10.1145/3448016.3457550

[42] Lan Wang, Yujia Chen, Du Tran, Vishnu Naresh Boddeti, and Wen-Sheng Chu. 2025. SEAL: Semantic Attention Learning for Long Video Representation. arXiv:2412.01798 [cs.CV] https://arxiv.org/abs/2412.01798

[43] Liang Wang, Xiaoyang Qu, Jianzong Wang, Guokuan Li, Jiguang Wan, Nan Zhang, Song Guo, and Jing Xiao. 2024. Gecko: Resource-Efficient and Accurate Queries in Real-Time Video Streams at the Edge. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*. 481–490. https://doi.org/10.1109/INFOCOM52122.2024.10621399

[44] Li Wang and Dennis Sng. 2015. Deep Learning Algorithms with Applications to Video Analytics for A Smart City: A Survey. arXiv:1512.03131 [cs.CV] https://arxiv.org/abs/1512.03131

[45] Y. Weng, M. Han, H. He, X. Chang, and B. Zhuang. 2024. LongVLM: Efficient Long Video Understanding via Large Language Models. arXiv preprint. https://doi.org/10.48550/arXiv.2404.03384 arXiv:2404.03384 [cs.CV]

[46] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. 2022. GMFlow: Learning Optical Flow via Global Matching. arXiv:2111.13680 [cs.CV] https://arxiv.org/abs/2111.13680

[47] Li Xu, He Huang, and Jun Liu. 2021. SUTD-TrafficQA: A Question Answering Benchmark and an Efficient Network for Video Reasoning over Traffic Events. arXiv:2103.15538 [cs.CV] https://arxiv.org/abs/2103.15538

[48] Z. Xu, G. T. Kakkar, J. Arulraj, and U. Ramachandran. 2022. Eva: A symbolic approach to accelerating exploratory video analytics with materialized views. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 602–616.

[49] E. Zhang, M. Daum, D. He, et al. 2023. EQUI-VOCAL: Synthesizing Queries for Compositional Video Events from Limited User Interactions. *Proceedings of the VLDB Endowment* 16, 11 (2023), 2714–2727. https://doi.org/10.14778/3611479.3611482

[50] Miao Zhang, Fangxin Wang, Yifei Zhu, Jiangchuan Liu, and Bo Li. 2021. Serverless Empowered Video Analytics for Ubiquitous Networked Cameras. *IEEE Network* 35, 6 (2021), 186–193. https://doi.org/10.1109/MNET.101.2000668

[51] W. Zhang, Q. M. Wu, and H. B. Yin. 2010. Moving vehicles detection based on adaptive motion histogram. *Digital Signal Processing* 20, 3 (2010), 793–805.

[52] Sheng Zhou, Hongjia Xu, Zhuonan Zheng, Jiawei Chen, Zhao li, Jiajun Bu, Jia Wu, Xin Wang, Wenwu Zhu, and Martin Ester. 2022. A Comprehensive Survey on Deep Clustering: Taxonomy, Challenges, and Future Directions. arXiv:2206.07579 [cs.LG] https://arxiv.org/abs/2206.07579