Mining Platoon Patterns from Traffic Videos

Yijun Bei
Zhejiang University, China
beiyj@zju.edu.cn

Teng Ma
Zhejiang University, China
mt0228@zju.edu.cn

Dongxiang Zhang*
The State Key Laboratory of Blockchain and Data Security, Zhejiang University
zhangdongxiang@zju.edu.cn

Sai Wu
Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security
wusai@zju.edu.cn

Kian-Lee Tan
National University of Singapore
tankl@comp.nus.edu.sg

Gang Chen
Zhejiang University, China
cg@zju.edu.cn

ABSTRACT

Discovering co-movement patterns from urban-scale video data sources has emerged as an attractive topic. This task aims to identify groups of objects that travel together along a common route, which offers effective support for government agencies in enhancing smart city management. However, the previous work has made a strong assumption on the accuracy of recovered trajectories from videos and their co-movement pattern definition requires the group of objects to appear across consecutive cameras along the common route. In practice, this often leads to missing patterns if a vehicle is not correctly identified from a certain camera due to object occlusion or vehicle mis-matching.

To address this challenge, we propose a relaxed definition of co-movement patterns from video data, which removes the consecutiveness requirement in the common route and accommodates a certain number of missing captured cameras for objects within the group. Moreover, a novel enumeration framework called MaxGrowth is developed to efficiently retrieve the relaxed patterns. Unlike previous filter-and-refine frameworks comprising both candidate enumeration and subsequent candidate verification procedures, MaxGrowth incurs no verification cost for the candidate patterns. It treats the co-movement pattern as an equivalent sequence of clusters, enumerating candidates with increasing sequence length while avoiding the generation of any false positives. Additionally, we also propose two effective pruning rules to efficiently filter the non-maximal patterns. Extensive experiments are conducted to validate the efficiency of MaxGrowth and the quality of its generated co-movement patterns. Our MaxGrowth runs up to two orders of magnitude faster than the baseline algorithm. It also demonstrates high accuracy in real video dataset when the trajectory recovery algorithm is not perfect.

PVLDB Reference Format:
Yijun Bei, Teng Ma, Dongxiang Zhang, Sai Wu, Kian-Lee Tan, and Gang Chen. Mining Platoon Patterns from Traffic Videos. PVLDB, 18(6): 1839 - 1851, 2025.
doi:10.14778/3725688.3725710

*Corresponding author.
This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 18, No. 6 ISSN 2150-8097.
doi:10.14778/3725688.3725710

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Mateng0228/VPlatoon.

1 INTRODUCTION

Co-movement pattern mining from large-scale GPS trajectories has been extensively studied over the past two decades [2, 9, 13, 16]. Recently, this mining task was applied to video data to aid government agencies in smart city management, including detecting traffic congestion at various granularities [18] and monitoring suspicious groups for security [28]. The underlying motivation is that trajectory data are primarily collected and owned by commercial hail-riding or map-service companies, but not directly accessible to government agencies. In contrast, government-installed surveillance cameras have covered the entire urban area, prompting the re-examination of co-movement pattern mining from videos.

In the scenario of video mining, the trajectory of each individual object is extracted using cross-camera trajectory recovery algorithms [11, 22, 30] and represented as a sequence of camera IDs and time intervals. As an example, Figure 1 depicts the golden (ground-truth) trajectories and recovered trajectories of three objects o_1, o_2, and o_3. Since the GPS trajectory-based pattern definition is unsuitable for video data, the co-movement pattern is redefined as a group of objects traveling together along a common route in the road network [31]. Specifically, this definition involves three parameters: m, k, and ϵ. That is, each valid group must contain at least m objects; these objects must traverse at least k consecutive cameras which constitute the common route; and the objects must be captured by each common camera within a time interval of ϵ. We call this pattern VConvoy, as it can be regarded as a migration of the GPS trajectory-based pattern Convoy [12]. In Figure 1, three objects travel together across cameras B, C and D. If $m = 3$, $k = 2$ and $\epsilon = 4$, a valid VConvoy pattern $\langle\{o_1, o_2, o_3\}, B \rightarrow C \rightarrow D\rangle$ can be mined from the golden trajectories.

However, the trajectory recovery algorithms in practice are not perfect, and the available recovered trajectories often exhibit differences from the golden trajectories. On the other hand, VConvoy assumes no accuracy loss in the mining trajectories and requires

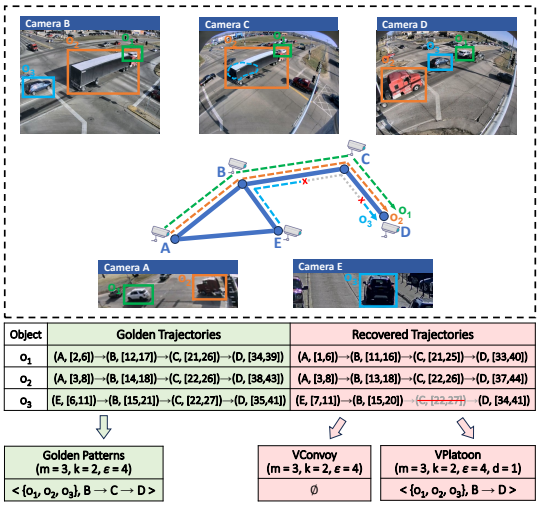


Figure 1: An illustrative example for relaxed co-movement pattern mining from real video data.

that the objects in a valid pattern appear across a consecutive sequence of cameras to constitute the common route. As a result, this rigid setting incurs the risk of pattern missing. For instance, in Figure 1, vehicle o_3 is occluded by truck o_2 in camera C, resulting in a recovered trajectory of $E \rightarrow B \rightarrow D$ that omits camera C. Hence, when applying VConvoy pattern mining to the recovered trajectories, the pattern $\langle\{o_1, o_2, o_3\}, B \rightarrow C \rightarrow D\rangle$, which is identifiable from the golden trajectories, will be absent. Additionally, object occlusion is not the sole issue; object ID switching poses a significant challenge for existing object tracking algorithms [6, 20, 24]. If moving objects are incorrectly identified and assigned wrong IDs, VConvoy mining will also fail under such circumstances.

To effectively support co-movement mining from imperfectly recovered trajectories, we propose a new definition of a relaxed pattern in this paper. Inspired by the platoon pattern [15], we remove the requirement for consecutive camera sequences and introduce a parameter d, which allows for a certain number of missing cameras in the common route of objects within the group. We call this new pattern VPlatoon. In other words, VPlatoon involves four parameters: m, k, ϵ, and d. It still requires a valid group of at least m objects to exhibit temporal proximity ϵ along a common route of length at least k, but the objects in the group now can briefly leave the common route with a maximum camera gap of d. For example, the common route among the recovered trajectories of $\{o_1, o_2, o_3\}$ in Figure 1 is $B \rightarrow D$, bypassing a missing camera C. With $d = 1$, the pattern $\langle\{o_1, o_2, o_3\}, B \rightarrow D\rangle$ becomes a valid VPlatoon according to our relaxed pattern definition.

After eliminating the constraint of a consecutive sequence of cameras, the search space is expanded and the problem becomes more complex than the original VConvoy mining problem, which has been proved to be NP-Hard in [31]. We present a baseline algorithm that extends the TCS-tree algorithm [31]. This baseline builds on the basic filter-and-refine framework of the TCS-tree with two key improvements. First, in the filter stage, we substitute the original frequent substring miner in TCS-tree with the sequential pattern miner [25, 29] and further improve filter effectiveness based on the idea of camera partitioning. Second, in the refinement stage, we incorporate multiple buffers to efficiently accommodate the gap tolerance parameter d.

According to our empirical analysis, the baseline suffers from high candidate verification cost because the candidate search space is huge and a valid candidate requires multiple constraints to be satisfied simultaneously. In this paper, we propose a novel candidate enumeration framework called MaxGrowth. It eliminates the necessity of the expensive verification procedure without generating any false positives. Thus, compared with the baseline, it incurs no verification cost and the total mining time can be remarkably reduced. The core idea is to treat a VPlatoon pattern as an equivalent sequence of clusters, where each cluster comprises a specific camera and a group of proximate objects. Patterns are then constructed by incrementally expanding the cluster sequences. In such a manner, the pattern mining task is decomposed into a series of feasible cluster selection problems, and MaxGrowth can effectively utilize pattern constraints during enumeration while avoiding the need to handle numerous intermediate false-positive candidates. Moreover, to efficiently remove valid but non-maximal candidate patterns, we introduce two pruning rules: the root pruning rule and the dependency pruning rule. The former prunes non-contributing search space at the root node of the search tree, while the latter allows for more refined pruning in subsequent search nodes during candidate enumeration.

To sum up, the key contributions of this paper are as follows.

- We present a new type of video-based co-movement pattern mining problem, whose goal is to be more tolerant with flawed video tracking and trajectory recovery algorithms.
- We present a novel candidate enumeration framework called MaxGrowth that eliminates the necessity of candidate validness verification cost.
- Two effective pruning rules are proposed to remove non-maximal candidate patterns, which can reduce the cost of dominance verification by over 90%.
- Extensive experiments are conducted on real-world datasets, validating the effectiveness of the proposed pattern while demonstrating the efficiency and scalability of our pattern mining techniques.

The rest of the paper is organized as follows. We review related literature in Section 2. The basic concepts and problem statement are presented in Section 3. Section 4 presents the baseline algorithm. we propose our enumeration framework MaxGrowth in Section 5 and the two pruning rules for maximal pattern mining are presented in Section 6. Experimental evaluation is conducted in Section 7. We conclude the paper in Section 8.

2 RELATED WORK

Co-movement pattern mining intends to find a group of objects moving within spatial proximity over a specified time interval. Previous studies can be divided into two categories according to data sources: GPS trajectory-based and video-based.

In the realm of GPS trajectory-based co-movement patterns, the flock [2, 10] and convoy [12] patterns both require that candidate groups appear across consecutive timestamps. The primary

distinction between them lies in their definition of spatial proximity. The flock pattern requires objects within the same cluster to be within a disk with a diameter smaller than a specified parameter, whereas the convoy pattern employs density-based spatial clustering [8]. The performance bottleneck in mining these two types of co-movement patterns is the clustering overhead applied at each timestamp. Besides general acceleration methods in spatiotemporal management [5, 7, 26], several specialized techniques for co-movement mining have been developed to mitigate this issue, such as trajectory simplification [13], spatial partitioning [4, 17], and the divide-and-conquer scheme [18].

In the definitions of group [27], swarm [16], and platoon [15] patterns, the temporal duration constraint is relaxed. Their mining algorithms follow similar solving schemes, where the main idea is to grow an object set from an empty set. Meanwhile, various pruning techniques were designed to ensure mining efficiency. The group pattern mining primarily utilizes its proposed VG-graph structure. While the platoon pattern miner applies multiple fine-grained pruning rules based on the prefix table. On the other hand, the swarm pattern mining algorithm mainly introduces two pruning rules called backward and forward pruning to trim the search space for mining closed swarm patterns. These optimization techniques are based on the analysis of time dimension, but in our scenario, the time information is restricted to independent time intervals, which hinders their adaptation to our problem.

Video-based co-movement pattern mining was first examined by Zhang et al. [31]. The problem was proven to be NP-hard. To solve it efficiently, an index called temporal-cluster suffix tree (TCS-tree) and a sequence-ahead pruning framework based on TCS-tree were proposed. To reduce verification cost on the candidate paths, the authors introduced a sliding-window based co-movement pattern enumeration strategy and a hashing-based dominance eliminator. Our work extends the definition to address the issue of missing patterns incurred by object occlusion or vehicle mis-matching.

3 PROBLEM DEFINITION

Given a corpus of input video data captured by surveillance cameras in scenes such as urban areas or highways. We denote all objects appearing in the input video as $\mathbb{O} = \{o_1, o_2, o_3 \dots\}$. Following previous definition of co-movement pattern mining [31], we assume that the travel paths of these objects can be extracted via trajectory recovery algorithms [11, 22] as a pre-processing step.

Definition 1. Travel Path
The travel path P_i of an object $o_i \in \mathbb{O}$ is defined as a sequence of surveillance cameras with associated time intervals:
$P_i = (c_1, [s_1, e_1]) \rightarrow (c_2, [s_2, e_2]) \rightarrow \dots \rightarrow (c_n, [s_n, e_n])$
where $(c_j, [s_j, e_j])$ indicates that o_i is captured by camera c_j during time interval $[s_j, e_j]$ with $s_j < e_j$ and $s_j < s_{j+1}$.

We call s_j and e_j the entrance and exit timestamp of o_i at camera c_j, respectively, and the time interval of P_i is $[s_i, e_n]$. Moreover, when there is no need to mentioned the temporal dimension, we abbreviate P_i as $c_1 \rightarrow c_2 \rightarrow \dots \rightarrow c_n$ and let c_j^i denote the j-th camera (c_j) that object o_i passes through.

Example 1. *Figure 2 illustrates the travel paths of four example objects, where the road network is represented by thick solid lines and the extracted travel paths are represented by dashed lines. The travel path P_1 of object o_1 is $(A, [1, 6]) \rightarrow (B, [15, 25]) \rightarrow (D, [35, 39])$, and the time interval of P_1 is $[1, 39]$.*

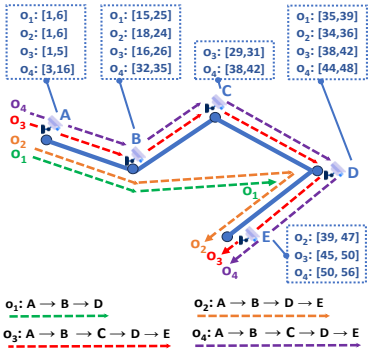


Figure 2: Travel paths for four objects $\{o_1, o_2, o_3, o_4\}$.

To represent the subpath relationships between imperfectly recovered trajectories and suppoert our relaxed definition of co-movement pattern, we define the notion of d-subpath.

Definition 2. d-subpath
The travel path P_i is a d-subpath of P_j, if the time interval of P_i is contained in P_j and there exists an injection function $f : \{1, 2, \dots, |P_i|\} \rightarrow \{1, 2, \dots, |P_j|\}$ satisfying:

- $\forall 1 \leq t < |P_i|, f(t+1) - f(t) \leq d + 1$.
- $\forall 1 \leq t \leq |P_i|, c_t^i = c_{f(t)}^j$.

The gap tolerance parameter d is used to controls the number of missing cameras between adjacent cameras when mapping P_i to P_j. Specifically, given that the time interval of P_i is contained within P_j, P_i exactly matches a consecutive substring of P_j when d = 0. In contrast, P_i can be any subsequence of P_j when d = ∞.

Example 2. *In Figure 2, when $d = 1$, P_1 is a d-subpath of P_3. First, the time interval of P_1 ($[1, 39]$) is contained in the time interval of P_3 ($[1, 50]$). Additionally, given the mapping $f : \{1, 2, 3\} \rightarrow \{1, 2, 4\}$, we find the corresponding consistent cameras in P_3 for each camera in P_1 according to f: $c_1^1 = c_1^3$, $c_2^1 = c_2^3$, and $c_3^1 = c_4^3$. Furthermore, for the mapped cameras in P_3, the distances between adjacent cameras are no more than 2, i.e., $c_4^3 - c_2^3 \leq 2$ and $c_2^3 - c_1^3 \leq 2$.*

We continue to define the notion of ϵ-close to indicate the proximity between objects. Here, ϵ-close implies the temporal closeness of objects passing through the same camera.

Definition 3. ϵ-close at camera c
A group of objects $O \subseteq \mathbb{O}$ is ϵ-close at camera c, if $\forall$ objects $o_i, o_j \in O$, the gap between their entrance timestamps at c (i.e., $|s_i - s_j|$) does not exceed ϵ.

If an object o belongs to a group of ϵ-close objects O, we also call that o is ϵ-close to any other objects in O.

Example 3. *As shown in Figure 2, When $\epsilon = 6$, we can say that the object set $\{o_1, o_2, o_3\}$ is ϵ-close at camera B, as the time gap between the entrance timestamps of any object pair is less than ϵ. Specifically,*

$|s_1 - s_2| = 3 \leq 6$, $|s_1 - s_3| = 1 \leq 6$, *and* $|s_2 - s_3| = 2 \leq 6$. *In contrast, the object set* $\{o_1, o_2, o_3, o_4\}$ *is not* ϵ*-close at camera B because* $|s_1 - s_4| = 17 > 6$.

Now, we formalize the definition of the relaxed co-movement pattern from videos (VPlatoon), which represents a group of objects traveling together satisfying ϵ-close along a common d-subpath.

Definition 4. The Relaxed Co-movement Pattern
For parameters m, k, d and ϵ, the relaxed co-movement pattern is defined as $R = \langle O, P \rangle$, where O and P represent a set of objects and a common path respectively. Meanwhile, it must satisfy the following conditions:

(1) O is ϵ-close at each camera of P.
(2) For each object $o_i \in O$, P is a d-subpath of P_i.
(3) $|O| \geq m$.
(4) $|P| \geq k$.

Example 4. *In Figure 2, given* $m = 2$, $k = 3$, $d = 1$, *and* $\epsilon = 6$, $\langle \{o_1, o_2, o_3\}, A \rightarrow B \rightarrow D \rangle$ *is a relaxed co-movement pattern. The object set* $\{o_1, o_2, o_3\}$ *contains more than 2 objects and is* ϵ*-close at cameras A, B and D. Meanwhile, the common route* $A \rightarrow B \rightarrow D$ *has a length of at least 3 and is a d-subpath of* P_1, P_2, *and* P_3.

Those relaxed co-movement patterns with a large number of objects or a long common route could possess a combinatorial number of redundant "sub-patterns" that do not convey more information than their parent pattern. For the sake of conciseness, we define the concept of maximal pattern.

Definition 5. Maximal Relaxed Co-movement Pattern
A relaxed co-movement pattern $R_i = \langle O_i, P_i \rangle$ is maximal if there exists no other pattern $R_j = \langle O_j, P_j \rangle$ such that $O_i \subseteq O_j$ and P_i is a d-subpath of P_j.

Our goal is to discover all the maximal relaxed co-movement patterns under parameters m, k, d and ϵ.

Example 5. *In Figure 2, for* $m = 2$, $k = 3$, $d = 1$, *and* $\epsilon = 6$, $R = \langle \{o_1, o_2, o_3\}, A \rightarrow B \rightarrow D \rangle$ *is a maximal pattern, whereas* $R_i = \langle \{o_2, o_3\}, A \rightarrow D \rangle$ *is not a maximal pattern. This is because the object set* $\{o_2, o_3\}$ *in* R_i *is a subset of* $\{o_1, o_2, o_3\}$, *and the common path* $A \rightarrow D$ *in* R_i *is a d-subpath of* $A \rightarrow B \rightarrow D$. *In contrast, there is no other pattern for R that satisfies above similar conditions.*

Table 1 provides a summary of the frequently used notations.

Table 1: Notation Table.

$\mathbb{O}$	All moving objects contained in $\mathbb{V}$
P_i	The travel path of an object $o_i \in \mathbb{O}$
R	The relaxed co-movement pattern from video data
m	The minimum number of objects in R
k	The minimum length of the common route in R
d	The gap tolerance in R
ϵ	The threshold of closeness in R
$R.O/P$	The object set / common route of R
$CL_i = (O_i, c_i)$	A certain cluster with object set O derived from camera c
p_j^i	The position of camera $CL_i.c$ in object o_j's travel path P_j
S	A cluster sequence
$\lambda(S)$	The core objects of S
$R(S)$	The candidate co-movement pattern corresponding to S

4 BASELINE ALGORITHM

In this section, we propose a baseline algorithm that extends TCS-tree [31] for VPlatoon pattern mining. It adopts the filter-and-refine framework similar to TCS-tree and we call this baseline FRB algorithm. We begin with a brief introduction to TCS-tree, then provide the detailed presentation of the FRB.

The TCS-tree algorithm is a state-of-the-art VConvoy mining algorithm that follows the filter-and-refine framework. Its filter stage relies on an efficient index, the temporal-cluster suffix tree, which performs two-level temporal clustering within each camera and constructs a suffix tree from the resulting clusters. By adopting frequent substring mining based on this index, TCS-tree identifies common subsequences as candidates through low granularity filtering. Each subsequence is a consecutive camera sequence with a length of at least k and support from at least m proximate objects. In the subsequent refinement stage, TCS-tree uses a modified CMC algorithm [13, 14] to precisely verify the temporal proximity (parameter ϵ) of objects along the common subsequence for each candidate and obtain the final results.

In order to extend TCS-tree as a tailored algorithm for VPlatoon pattern mining, we introduce two key improvements. First, in the filter stage, we replace the original frequent substring miner with the sequential pattern miner [25, 29] for candidate enumeration and partition each camera based on temporal proximity to improve enumeration effectiveness. Second, in the refinement stage, we incorporate multiple buffers into the original candidate verification algorithm to accommodate the gap tolerance parameter d.

We now present the filter stage in detail. In this stage, FRB focuses on the route information within the input travel paths and identifies frequent camera subsequences that represent common routes as candidates. Specifically, each candidate subsequence must contain at least k cameras and appear in the travel paths of at least m objects. Moreover, since the definition of VPlatoon removes the consecutive constraint on the common route, candidate subsequences can be traversed both consecutively and non-consecutively. Therefore, the basic scheme of FRB in the filter stage is to first remove the time intervals from each moving object's travel path and represent the path as a sequence of cameras in ascending order of their entrance times. Then, the sequential pattern mining algorithm [25, 29] with the minimum length of k and support of m is applied to these camera sequences for candidate enumeration.

Furthermore, since the sequential pattern mining algorithm cannot be accelerated by the TCS-tree index, we adapt the core idea of this index to partition the cameras and generate a more nuanced sequence representation. Performing sequential pattern mining on these nuanced sequences instead of the camera sequences, FRB could filter out more false positives in the filter stage.

Specifically, we partition each camera based on the temporal proximity of the objects passing through it. The partitions of a camera c is defined as $\mathbb{T}_c = \{T_c^1, T_c^2, ..., T_c^n\}$, where:

- For $1 \leq i \leq n$, T_c^i is a set of objects passing through c.
- $\bigcup_{i=1}^{n} T_c^i$ constitutes all the objects passing through c.
- $\forall o_u \in T_c^i$, if there exists o_v that o_u is ϵ-close to o_v, then o_v is also in T_c^i; otherwise, o_u is the only element in T_c^i.

It's not difficult to see that an object belongs to exactly one partition when passing through a given camera. We can transform each

object's camera sequence into a partition sequence corresponding to the cameras it passes through. Two objects might have a common camera subsequence, but their partition sequences can be completely different if they are not ϵ-close. Therefore, this fine-grained representation offers stronger candidate filtering capabilities.

Example 6. *In Figure 2, let $\epsilon = 6$. Since all objects at camera A are ϵ-close, $\mathbb{T}_A = \{\{o_1, o_2, o_3, o_4\}\}$. For camera B, we have o_1, o_2, o_3 are ϵ-close but o_4 is not ϵ-close to them, so $\mathbb{T}_B = \{\{o_1, o_2, o_3\}, \{o_4\}\}$. Similarly, $\mathbb{T}_C = \{\{o_3\}, \{o_4\}\}$. If we only consider cameras A, B and C, objects o_3 and o_4 share the same sequence of camera $A \rightarrow B \rightarrow C$. However, their partition sequences are $T_A^1 \rightarrow T_B^1 \rightarrow T_C^1$ and $T_A^1 \rightarrow T_B^2 \rightarrow T_C^2$, respectively. Mining on these two partitions prevents the generation of false positive $(A \rightarrow B \rightarrow C, \{o_3, o_4\})$.*

The following lemma is presented to demonstrate the completeness of FRB in the filter stage.

Lemma 1. *Each valid VPlatoon pattern is contained in at least one candidate from the filter stage.*

Proof. See our extended technical report [1]. □

We proceed to describe the refinement stage, where FRB refines each candidate (Seq, O) to ensure the temporal proximity (ϵ) and gap tolerance (d) of objects in O as they travel along the common camera subsequence Seq. The approach is similar to the candidate verification algorithm in TCS-tree, with additional buffer structures introduced to improve efficiency in adapting to the VPlatoon definition. Specifically, the algorithm first computes all groups of objects in O that are ϵ-close for each camera of Seq. It then searches for valid patterns along Seq while maintaining a global buffer at each camera to store partial patterns that successfully reach it. During the search at each camera, the groups of ϵ-close objects are intersected with the partial patterns stored in the buffers of the previous $d + 1$ cameras in Seq, thereby generating new VPlatoon patterns.

Algorithm 1 demonstrates the FRB algorithm. During the filter stage (lines 1-4), we first retrieve the camera sequences of all objects by removing the time intervals from their travel paths (line 1). Then, we compute all the partition sequences based on our aforementioned description of partitions (lines 2-3). In the final step of the filter stage, sequential pattern mining is performed on $\mathbb{S}_T$ for candidate enumeration (line 4). During the subsequent refinement stage (lines 5-19), we first initialize $\mathbb{R}$ as the final result set and Buf as the global buffer set (line 5). Then, the algorithm iterates over the cameras in Seq of each candidate (lines 6-7). At each processed camera c_i, all groups of ϵ-close objects in O are computed according to Definition 3 and intersected with previous partial patterns to generate new partial patterns R_{new} (lines 8-11). At this point, it's important to note that we still need to further verify whether each object in $R_{new}.O$ has traversed more than d cameras that are not in Seq between the current camera c_i and the last reached camera c_j. This is done using the distance function $distance(o, c_i, c_j)$ (lines 12-13). Afterward, we can then update Buf and $\mathbb{R}$ (lines 14-17). Finally, the algorithm derives the final results by removing all non-maximal patterns from $\mathbb{R}$. This step is similar to the one used in TCS-tree.

Algorithm 1: The FRB Algorithm

```
1  S_c ← retrieve the camera sequences of all objects;
2  T ← compute partitions of each camera;
3  S_T ← convert S_c into partition sequences based on T;
4  R_c ← sequential pattern mining on S_T with m and k;
5  R ← ∅; Buf ← ∅;
6  foreach candidate (Seq, O) ∈ R_c do
7      foreach camera c_i in Seq do
8          O_ε ← compute groups of ε-close objects in O at c_i;
9          for camera c_j ∈ {c_{i-1-d}, ..., c_{i-1}} do
10             for (O_ε, R_b) ∈ O_ε × Buf_{c_j} do
11                 R_new = ⟨R_b.O ∩ O_ε, R_b.P ∪ c_i⟩;
12                 for each object o ∈ R_new.O do
13                     if distance(o, c_i, c_j) > d then discard o;
14                 if |R_new.O| < m then continue;
15                 Buf_{c_i} ← Buf_{c_i} ∪ {R_new};
16                 if |R_new.P| ≥ k then R ← R ∪ {R_new};
17         foreach O_ε ∈ O_ε do Buf_{c_i} ← Buf_{c_i} ∪ {⟨O_ε, [c_i]⟩};
18 R ← remove non-maximal patterns from R;
19 return R;
```

5 CANDIDATE ENUMERATION FRAMEWORK

The drawback of FRB algorithm is that its filter-and-refine framework incurs substantial computation overhead for candidate verification. As will be empirically examined in Section 7, the candidate verification becomes the performance bottleneck in FRB algorithm. To address the challenge, we present MaxGrowth algorithm, which eliminates the requirement for candidate verification. In this section, we first present its candidate enumeration scheme to find valid candidate patterns, and in the next section, we present the pruning rules to efficiently eliminate patterns that are non-maximal.

The core idea of MaxGrowth is to treat the co-movement pattern as a sequence of fundamental units (clusters) and iteratively grow such a sequence for pattern generation. This scheme contrasts with the traditional filter-and-refinement framework, which treats the co-movement pattern as an overall combination of objects with their common route and performs refinement for pattern generation. In other words, MaxGrowth decomposes the co-movement pattern mining problem into a series of feasible cluster selection problems during candidate enumeration. The expensive verification procedure is avoided without generating false positives. We next introduce the cluster representation of the co-movement pattern in Subsection 5.1 and then elaborate on the candidate enumeration scheme of MaxGrowth in Subsection 5.2.

5.1 Cluster Representation

As the fundamental unit in MaxGrowth, a cluster is essentially an aggregation of a set of objects and a traversed camera.

Definition 6. Cluster
A cluster is defined as $CL = (O, c)$, if $|O| \geq m$ and O is ϵ-close at c.

We use $CL.O$ and $CL.c$ to denote the object set and the traversed camera of a given cluster CL, respectively. Besides, it is sometimes

necessary to determine the specific position of an object as it traverses the camera of a given cluster in order to analyze the positional relationships between clusters. Therefore, we use the notation $\boldsymbol{p_j^i}$ to represent the position of object o_j in its travel path P_j when passing through the camera in cluster CL_i.

Example 7. *In the example of Figure 2, if we set $m = 2$ and $\epsilon = 6$, then the object set $\{o_1, o_2, o_3\}$ contains more than 2 objects and is ϵ-close at camera B. Thus, $(\{o_1, o_2, o_3\}, B)$ can be a cluster, which is shown as CL_2 in Figure 3. Meanwhile, since o_1's travel path is $P_1 = (A, [1, 6]) \rightarrow (B, [15, 25]) \rightarrow (D, [35, 39])$, and o_1 belongs to $CL_2.O$ at the second position of P_1, we have $p_1^2 = 2$.*

Intuitively, a cluster describes the co-movement scene under a single camera. To further characterize the co-movement pattern along the common route, we need to link the clusters in a reasonable manner. To this end, we now present several essential definitions related to the cluster sequence.

Definition 7. Core Object
We denote object o_c as a core object of the cluster sequence S if:

- $o_c \in \bigcap_{i=1}^{|S|} CL_i.O$.
- For $1 < i \leq |S|$, $0 < p_c^i - p_c^{i-1} \leq d + 1$.

Here, $|S|$ represents the number of clusters (length) in the cluster sequence $S = [CL_1, \ldots, CL_{|S|}]$. We further use the notation $\boldsymbol{\lambda(S)}$ to denote all the core objects in S. Actually, $\lambda(S)$ indicates the group of objects that is ϵ-close along a common d-subpath constructed from the cameras in S.

Example 8. *Given parameters $m = 2$ and $d = 1$ in Figure 3, we take the cluster sequence $S = [CL_1, CL_2, CL_3]$ as an example. We have $CL_1.O \cap CL_2.O \cap CL_3.O = \{o_1, o_2, o_3\}$. Meanwhile, the conditions $0 < p_1^3 - p_1^2 < 2$ and $0 < p_1^2 - p_1^1 < 2$ hold. Therefore, we can conclude that o_1 is a core object of S. Similarly, $\lambda(S) = \{o_1, o_2, o_3\}$.*

Definition 8. Feasible Cluster
We call cluster CL_i a feasible cluster for the cluster sequence $S = [CL_1, \ldots, CL_n]$ if there exist at least m objects $o_c \in CL_i.O$ such that $o_c \in \lambda(S)$ and $0 < p_c^i - p_c^n \leq d + 1$.

Besides, we also define the cluster sequence $S = [CL_1, ..., CL_n]$ as a **feasible sequence** if the last cluster CL_n in S is a feasible cluster for the prefix subsequence $[CL_1, ..., CL_{n-1}]$.

Example 9. *We use Figure 3 for illustration with parameters $m = 2$ and $d = 1$. The cluster CL_3 is a feasible cluster for $[CL_1, CL_2]$. This is because $CL_3.O \cap \lambda(S) = \{o_1, o_2, o_3\}$. Furthermore, we have $p_1^3 - p_1^2 = 1$, $p_2^3 - p_2^2 = 1$, and $p_3^3 - p_3^2 = 2$. Thus, there exist three objects (greater than m) that satisfy the conditions specified in Definition 8. As a result, we conclude that $[CL_1, CL_2, CL_3]$ is a feasible sequence. In contrast, CL_4 is not a feasible cluster for $[CL_1, CL_2]$ because $CL_1.O \cap CL_2.O \cap CL_4.O$ are $\{o3\}$ which consists of only one object.*

The following lemma demonstrates the equivalence between the feasible sequence and the candidate co-movement pattern.

Lemma 2. *For a feasible sequence $S = [CL_1, \ldots, CL_n]$, the corresponding $\langle \lambda(S), CL_1.c \rightarrow \ldots \rightarrow CL_n.c \rangle$ must be a candidate co-movement pattern of length n that satisfies m, ϵ, and d.*

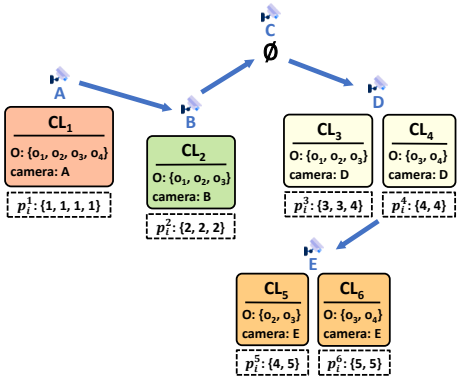


Figure 3: Example clusters with parameters $m = 2$ and $\epsilon = 6$.

Proof. We first consider the parameter m. Since S is a feasible sequence, CL_n must be a feasible cluster for $S_{n-1} = [CL_1, \ldots, CL_{n-1}]$. In other words, there are at least m objects o_c in $CL_n \cap \lambda(S_{n-1})$ such that $0 < p_c^n - p_c^{n-1} \leq d + 1$. As $\lambda(S_{n-1})$ contains all the core objects of S_{n-1}, we further have $CL_n \cap \lambda(S_{n-1}) = \bigcap_{i=1}^{n} CL_i.O$ and $0 < p_c^i - p_c^{i-1} \leq d+1 (1 < i \leq n)$. This implies that all the aforementioned at least m objects are core objects of S. As for the parameter ϵ, the core objects of $\lambda(S)$ remain within the same cluster, which ensures that they are ϵ-close at each camera. Finally, we analyze the parameter d. For each core object $o_c \in \lambda(S)$, we can define an injection function $f : \{1, 2, \ldots, n\} \rightarrow \{p_c^1, p_c^2, \ldots, p_c^n\}$ from the common route $CL_1.c \rightarrow \ldots \rightarrow CL_n.c$ to o_c's travel path P_c, such that the common route is a d-subpath of P_c. The proof is complete. □

We will henceforth use the notation $\boldsymbol{R(S)}$ to denote the corresponding candidate co-movement pattern $\langle \lambda(S), CL_1.c \rightarrow \ldots \rightarrow CL_n.c \rangle$ for a feasible sequence $S = [CL_1, \ldots, CL_n]$. For example, in Figure 3 with parameters $m = 2$ and $d = 1$, For the feasible sequence $S = [CL_1, CL_2, CL_3]$, $R(S)$ is $\langle \{o_1, o_2, o_3\}, A \rightarrow B \rightarrow D \rangle$.

5.2 The Candidate Enumeration Scheme

Based on the aforementioned cluster representation, the candidate enumeration scheme of MaxGrowth enumerates feasible sequences in a depth-first search manner for candidate generation. Specifically, it starts with an empty sequence and selects feasible clusters for the current cluster sequence to grow a longer feasible sequence at each search step. Meanwhile, candidate co-movement patterns are retrieved from the enumerated feasible sequences.

Algorithm 2 presents the complete pseudocode of MaxGrowth. We first initialize $\mathbb{R}$ as the final result set (line 1). Next, the candidate enumeration scheme of MaxGrowth is executed. Specifically, all the clusters are computed from the input travel paths according to Definition 6 for enumeration preparation (line 2). Each cluster is then used as the first element of the current cluster sequence S to initiate the Growth routine for subsequent steps in candidate enumeration (lines 3-5). Finally, we obtain the final result set $\mathbb{R}$ after removing non-maximal patterns (lines 6-7). This step uses a technique similar to that in [31]. We now continue to provide a detailed description of the Growth routine (lines 8-23), which is the core component of MaxGrowth's candidate enumeration scheme.

Algorithm 2: MaxGrowth

```
1  ℝ ← ∅;
2  ℂℒ ← compute all clusters using parameters m and ϵ;
3  foreach cluster CL ∈ ℂℒ do
4      S ← [CL];
5      Growth(S, ℝ);
6  ℝ ← remove non-maximal patterns in ℝ;
7  return ℝ;
8  Routine Growth(S = [CL_1, ..., CL_n], ℝ):
9      𝔽ℂℒ ← ∅; DT ← empty dictionary;
10     foreach core object o_c ∈ λ(S) do
11         ℂℒ_c ← {CL_i | o_c ∈ CL_i.O ∧ 0 < p_c^i − p_c^n ≤ d + 1};
12         foreach candidate cluster CL_i ∈ ℂℒ_c do
13             if CL_i ∉ DT then DT[CL_i] ← ∅;
14             DT[CL_i] ← DT[CL_i] ∪ {o_c};
15     foreach candidate cluster CL_i ∈ DT do
16         if |DT[CL_i]| ≥ m then 𝔽ℂℒ ← 𝔽ℂℒ ∪ {CL_i};
17     foreach feasible cluster FCL ∈ 𝔽ℂℒ do
18         Append FCL to S;
19         Growth(S, ℝ);
20         Discard FCL from S;
21     if |S| ≥ k then ℝ ← ℝ ∪ R(S) ;
```

In this routine, the feasible clusters for the current cluster sequence S are first computed in lines 9-16. Specifically, we iterate over each core object o_c of S, identifying candidate clusters that contain o_c and satisfy the gap tolerance condition $0 < p_c^i - p_c^n \leq d + 1$ (lines 10-11). o_c must be one of the objects that satisfy the conditions specified in Definition 8 within these candidate clusters. Therefore, each pair of candidate cluster and o_c is recorded in the auxiliary dictionary DT (lines 12-14). If a candidate cluster contains at least m core objects recorded in DT, it is considered a feasible cluster and added to $\mathbb{FCL}$ (lines 15-16). After obtaining all feasible clusters, each of them is selected and appended to S to continue the sequence growth routine (lines 17-20). As the final step of the Growth routine, we check if the length of the current cluster sequence is at least k. If so, the corresponding candidate co-movement pattern $R(S)$ is directly added to the result set $\mathbb{R}$ as a valid pattern.

From the above description, we can see that MaxGrowth derives valid co-movement patterns directly from the execution of the candidate enumeration scheme, thus eliminating the need for the subsequent expensive verification procedure. Moreover, during candidate enumeration, MaxGrowth searches only within the feasible clusters relevant to the current cluster sequence. This well-restricted search space typically shrinks as the sequence grows, further ensuring the efficiency of pattern growth.

We now justify the correctness and completeness of MaxGrowth in detail. First, the following lemma demonstrates its correctness.

Lemma 3. *MaxGrowth always generates valid co-movement patterns satisfying parameters m, k, d and ϵ from candidate enumeration.*

Proof. Since MaxGrowth only uses feasible clusters to grow the cluster sequence, each enumerated sequence S must be a feasible sequence. According to Lemma 2, S corresponds to a candidate co-movement pattern $R(S)$ that satisfies the parameters m, d, and ϵ. Meanwhile, as the length of the common route in $R(S)$ is the same as that of S, and MaxGrowth generates candidate patterns only for corresponding sequences with a length of at least k, $R(S)$ must also satisfy the parameter k. □

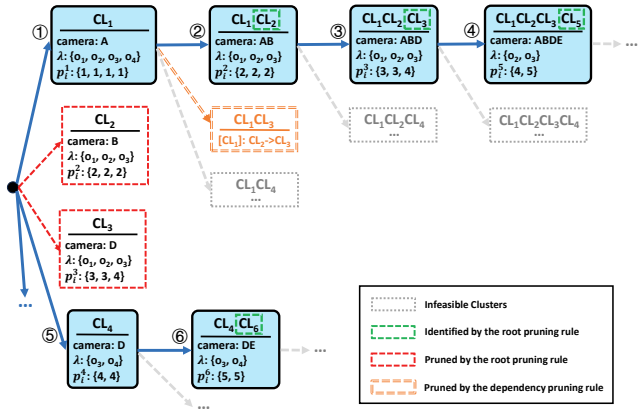


Figure 4: The search tree of MaxGrowth with parameters $m = 2$, $k = 2$, $d = 1$, and $\epsilon = 6$.

The following lemma implies the completeness of MaxGrowth.

Lemma 4. *A valid relaxed co-movement pattern corresponds to at least one cluster sequence enumerated by MaxGrowth.*

Proof. Let $R = \langle O, P \rangle$ be a relaxed co-movement pattern. Since O satisfies ϵ-close at each camera of P, there exists at least one cluster sequence $S = [CL_1, CL_2, \ldots, CL_n]$ such that $n = |P|$ and for $1 \leq i \leq n$, $O \subseteq CL_i.O$ and $CL_i.c$ is the i-th camera of P. Since MaxGrowth will enumerate all feasible sequences, we now only need to prove that S is a feasible sequence. First, we know $O \subseteq \bigcap_{i=1}^{n} CL_i.O$ contains at least m objects. Meanwhile, since P is the d-subpath of the travel path for each object in O, then $\forall o_j \in O$, $0 < p_j^i - p_j^{i-1} \leq d + 1$ holds for $1 < i \leq n$. This further implies that CL_n is a feasible cluster for the prefix subsequence $[CL_1, \ldots CL_{n-1}]$ of S. This completes the proof. □

We can now derive the following theorem which guarantees the correctness and completeness of MaxGrowth.

Theorem 1. *The MaxGrowth framework can generate both valid and complete relaxed co-movement patterns.*

Proof. The proof follows directly from Lemmas 3 and 4. □

6 MAXIMAL PATTERN MINING

Building upon the candidate enumeration scheme of MaxGrowth, we introduce two efficient pruning rules that eliminate the generation of most maximal patterns during candidate enumeration.

6.1 The Root Pruning Rule

The candidate enumeration scheme of MaxGrowth can be represented as a search tree, where each node corresponds to a selected feasible cluster and each branch represents a different direction of

sequence growth. Typically, there are a large number of clusters, resulting in an overwhelming number of subtrees at the root of the whole search tree. However, a significant number of these root subtrees cannot produce any maximal pattern. As shown in Figure 4, only the search subtrees starting from clusters CL_1 and CL_4 can generate the maximal patterns $\langle\{o_2, o_3\} : A \rightarrow B \rightarrow D \rightarrow E\rangle$ and $\langle\{o_3, o_4\} : D \rightarrow E\rangle$, respectively.

To effectively prune those non-contributing subtrees for maximal pattern mining at the root of the search tree, we present the root pruning rule. The lemma below illustrates the core idea of it.

Lemma 5. *For a cluster CL_i, if there is a feasible sequence S that CL_i is a feasible cluster for S and $CL_i.O \subseteq \lambda(S)$, then cluster sequences starting with CL_i cannot correspond to maximal patterns.*

Proof. Given a feasible sequence $S_n = [CL_1, \ldots, CL_n]$ with $CL_1 = CL_i$, it is straightforward to prove by showing that $R(S_n)$ is not a maximal pattern. See [1] for details. □

In Figure 4, since the cluster CL_2 is a feasible cluster for $S = [CL_1]$ and $CL_2.O \subseteq (CL_1.O = \lambda(S))$, then any feasible sequence starting with CL_2 cannot correspond to a maximal pattern. As an example, the candidate pattern $\langle\{o_1, o_2, o_3\}, B \rightarrow D\rangle$ of the sequence $[CL_2, CL_3]$ is not maximal because there exists another maximal pattern $\langle\{o_1, o_2, o_3\}, A \rightarrow B \rightarrow D\rangle$.

Based on Lemma 5, we state the following root pruning rule.

Rule 1. (The Root Pruning Rule) During the growth of a feasible sequence S, if a feasible cluster CL_i satisfies $CL_i.O \subseteq \lambda(S)$, then the search subtree at the root starting with CL_i can be pruned.

It's worth noting that the effectiveness of the root pruning rule is influenced by the search order of clusters in MaxGrowth. For example, in Figure 4, if we search for cluster CL_2 before CL_1, the substree starting with CL_2 cannot be pruned by Rule 1. Because this subtree can only be identified as prunable by Rule 1 during the growth of cluster sequences starting with CL_1. However, it has already been searched by that time. To mitigate this issue, we define the binary relation "precedence" on all the clusters and utilize it to specify the search order of clusters in MaxGrowth.

Definition 9. Precedence Relation
The precedence relation between two clusters CL_u and CL_v holds, denoted as $CL_u \leq_p CL_v$, if there is a series of clusters $CL_1, \ldots, CL_n$ such that:

- $CL_1 = CL_u$ and $CL_n = CL_v$.
- For $1 \leq i < n$, CL_i and CL_{i+1} share at least one common object o_j with $p_j^i \leq p_j^{i+1}$.

Example 10. *In the clusters shown in Figure 3, we have $CL_1 \leq_p CL_3$. This is because there exists a cluster series CL_1, CL_2, CL_3. Meanwhile, for CL_1 and CL_2, we can find a common object o_1 with $o_1 \in CL_1.O \wedge o_1 \in CL_2.O \wedge p_1^1 < p_1^2$. For CL_2 and CL_3, we can also a common object o_1 with $o_1 \in CL_2.O \wedge o_1 \in CL_3.O \wedge p_1^2 < p_1^3$.*

The precedence relation indicates the non-strict order in which objects move between clusters. In Definition 9, two adjacent clusters CL_i and CL_{i+1} share a common object o_j with $p_j^i \leq p_j^{i+1}$ ensuring that at least one object will pass through camera $CL_i.c$ before camera $CL_{i+1}.c$. And the existence of cluster series from CL_u to CL_v allows this movement order to be transitive from CL_u to CL_v.

Based on this relation, we determine the search order of clusters in MaxGrowth as follows: For clusters CL_u and CL_v, if $CL_u \leq_p CL_v$ and $CL_v \not\leq_p CL_u$, then MaxGrowth will search CL_u before CL_v.

Note that for two clusters CL_u and CL_v where both $CL_u \leq_p CL_v$ and $CL_v \leq_p CL_u$ are satisfied, we do not explicitly specify the search order between them. In practice, although symmetric pairs of precedence relations exist, their number is very small with the proportion less than ten percent. This is due to the inherent characteristics of clusters: Since objects within the same cluster exhibit temporal proximity, satisfying both $CL_u \leq_p CL_v$ and $CL_v \leq_p CL_u$ indicates that these two clusters simultaneously contain two sets of objects moving in opposite directions, which is rare and does not consistently occur across multiple clusters.

For clusters in Figure 3, since $CL_1 \leq_p CL_2$ and $CL_1 \not\leq_p CL_2$, MaxGrowth will search CL_1 before CL_2. In contrast, MaxGrowth will search CL_3 and CL_4 in arbitrary order. Thus, the overall search order is $CL_1 \rightarrow CL_2 \rightarrow \{CL_3, CL_4\} \rightarrow \{CL_5, CL_6\}$. With this order, Rule 1 successfully identifies all target clusters including CL_2, CL_3, CL_5, and CL_6, and prunes their search subtrees at the root node.

Since symmetric pairs are rare in precedence relation, this allows us to determine the search order for most of clusters. The lemma below theoretically guarantees the effectiveness of this search order.

Lemma 6. *If the precedence relation can specify the search order for all clusters, then the root pruning rule can prune the maximum number of search subtrees.*

Proof. See our extended technical report [1]. □

6.2 The Dependency Pruning Rule

Although the root pruning rule can prune subtrees at the root of the search tree, it cannot prune subtrees at higher levels during candidate enumeration. In fact, most maximal patterns are generated from the search subtrees of non-root nodes. Specifically, in the basic candidate scheme of MaxGrowth, we need to append each feasible cluster to the end of the current cluster sequence for further growth. However, searching through all feasible clusters for a cluster sequence as described above is unnecessary and time-consuming, as many subsequent search subtrees of feasible clusters will not produce any maximal patterns.

To facilitate more refined optimization of the search space for non-maximal patterns, we introduce the dependency pruning rule. The main idea of this rule is to determine non-essential search feasible clusters by analyzing whether a specific relationship between feasible clusters is satisfied. We call this specific relationship as **the dependency relationship** and describe it in detail below.

For a cluster sequence S and its feasible cluster CL_i, let the sequence $S_i = S \mid\mid [CL_i]$ represents the concatenation of two sequences S and $[CL_i]$. Then, the dependency relationship between two feasible clusters is defined as follow:

A feasible cluster CL_i depends on CL_j for a same feasible sequence S, if object $o_c \in \lambda(S_i) \rightarrow (o_c \in \lambda(S_j) \wedge p_c^j < p_c^i)$.

Intuitively, if a feasible cluster CL_i depends on another feasible cluster CL_j, it actually means that any object in $\lambda(S_i)$ must pass through camera $CL_j.c$ before reaching camera $CL_i.c$. In other words, S_i not only lacks information about the common route of objects

passing through camera $CL_j.c$, but its core objects can also be obtained through the further growth of S_j.

Example 11. *Given $m = 2$, $d = 1$, and $\epsilon = 6$ in Figure 3, there are two feasible clusters CL_2 and CL_3 for the feasible sequence $S = [CL_1]$. For these two clusters, we can also derive that $S_2 = [CL_1, CL_2]$, $S_3 = [CL_1, CL_3]$ and $\lambda(S_2) = \lambda(S_3) = \{o_1, o_2\}$. Since object o_1 satisfies $o_1 \in \lambda(S_2)$ as well as $o_1 \in \lambda(S_2) \wedge p_1^2 < p_1^3$; and object o_2 satisfies $o2 \in \lambda(S_3)$ as well as $o_2 \in \lambda(S_3) \wedge p_2^2 < p_2^3$, we can conclude that CL_3 depends on CL_2.*

The following lemma demonstrates how to prune non-essential subsequent search subtrees for feasible clusters based on the aforementioned dependency relationship.

Lemma 7. *If cluster CL_i depends on CL_j for a feasible sequence S, then the subsequent sequence growth after appending CL_i to S will not produce any maximal patterns.*

Proof. Assuming $S = [CL_1, CL_2, ..., CL_n]$, let S_i denote the cluster sequence $S \,||\, [CL_i]$ and S_{ji} denote $S \,||\, [CL_j, CL_i]$, where $||$ still represents the concatenation of two cluster sequences. We prove that the search tree rooted at S_i and S_{ji} are identical. Since the last cluster of S_i and S_{ji} are both CL_i, this is equivalent to proving their core objects are the same. Based on the dependency relationship, each $o_c \in \lambda(S_i)$ satisfies the following conditions: (1) $p_c^i - p_c^n \leq d + 1$, (2) $p_c^i - p_c^j > 0$, (3) $p_c^j - p_c^n > 0$. From (3) and (1) - (2), we can obtain $0 < p_c^j - p_c^n < d + 1$. Thus, $\lambda(S_i) \subseteq \lambda(S_j)$ and $\lambda(S_i) \subseteq (CL_i.O \cap \lambda(S_j))$. Meanwhile, from (2) and (1) - (3), we have $0 < p_c^i - p_c^j < d + 1$. Therefore, S_{ji} is a feasible sequence with core objects that contain at least $\lambda(S_i)$. As a result, S_i and S_{ji} have the same core objects. □

Based on Lemma 7, we state the dependency pruning rule.

Rule 2. (The Dependency Pruning Rule) During the growth of a feasible sequence $S = [CL_1, CL_2, ..., CL_n]$, if a feasible cluster CL_i depends on at least one other feasible cluster, then the search substree for CL_i after S can be pruned.

As shown in Figure 4, we know from the previous example that for candidate sequence $S = [CL_1]$, the feasible cluster CL_3 depends on CL_2. Therefore, we do not need to append CL_3 to S for further sequence growth.

The dependency pruning rule prunes on non-root search nodes during sequence growth (candidate enumeration) by detecting dependency relationships between feasible clusters. Specifically, if a feasible cluster CL_i can be pruned, it means that searching CL_i directly will lead to the same subsequent search subtree as first searching the clusters that CL_i depends on and then searching CL_i. Therefore, there is no need to continue searching CL_i after S.

7 EXPERIMENTAL EVALUATION

In this section, we evaluate the effectiveness of the relaxed co-movement pattern and the scalability of MaxGrowth algorithm.

7.1 Experimental Setup

Datasets. Following previous work [31], we use real GPS trajectories and road network, including DIDI Chengdu [23] and Singapore Taxi [9], to generate approximate trajectories recovered from videos. The idea is to deploy a specified number of cameras on the road network. Then, the entrance time and exit time from a camera according to the travel speed estimation and the attributes of the camera can be roughly estimated from the GPS trajectories.

- **Singapore**: This is a large-scale cross-camera trajectory dataset which contains travel routes of 2,756 taxis in the road network of Singapore over one month.
- **Chengdu**: This dataset contains cross-camera trajectories of 12,000 ride-hailing vehicles in Chengdu, China.

Besides these two semi-synthetic datasets from GPS trajectories, we also construct a real dataset from video data:

- **CityFlow** [21]: This video dataset contains 215 minutes of videos collected from 46 cameras spanning 16 intersections in a mid-sized U.S. city. We adopt the multi-camera multi-target tracking algorithm [30] to recover travel paths of objects in the CityFlow videos. The recovered trajectory dataset includes cross-camera trajectories of 337 unique objects over a duration of approximately 5 minutes.

Since CityFlow is a small-scale dataset, we employ Singapore and Chengdu for scalability analysis. CityFlow serves to measure the effectiveness of relaxed co-movement pattern mining.

Table 2: Statistics of datasets.

Attributes	Singapore	Chengdu	CityFlow
# objects	2,756	12,000	337
# cameras	37,370	9,476	19
# data points	2,458,742	1,841,071	2,563
Avg. trajectory length (# cameras per vehicle)	892	153	8
Time span per vehicle	6,246s	6,801s	59s

Comparison Setup. In the scalability analysis, we compare MaxGrowth with the baseline algorithm FRB, which is an extension of previous video-based co-movement pattern miner and presented in Section 4. For the effectiveness analysis, we compare our proposed relaxed co-movement pattern (referred to as VPlatoon) with previous co-movement pattern from [31] (referred to as VConvoy) in terms of their effectiveness for pattern discovery.

Parameter Setup. We examine the scalability performance w.r.t. the parameters in Table 3, including four query-related parameters (m, k, d and ϵ) and two database-related parameters object number and path length. The default parameters are highlighted in bold.

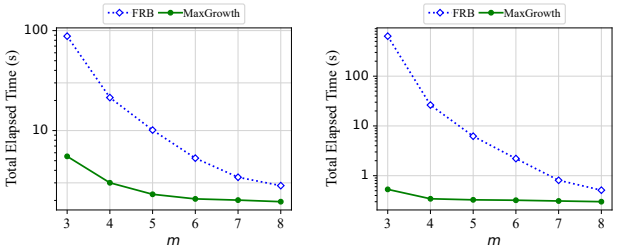
All the experiments are conducted on a server with 3.20GHz i9-12900K CPU, 256GB of memory and 2TB hard drive.

7.2 Scalability Analysis

Varying m. We first analyze the performance of the relaxed co-movement pattern mining algorithms w.r.t. group size m. As shown in Figure 5, when m is small, MaxGrowth demonstrates remarkably faster performance than the baseline algorithm FRB. Specifically, the runtime of MaxGrowth is reduced by two orders of magnitude when $m = 3$ in Chengdu dataset. The high efficiency of MaxGrowth stems from its ability to avoid the time-consuming verification of false positives and effectively discard non-maximal candidate patterns through pruning rules. As m increases, the performance gap narrows, with FRB showing similar efficiency to MaxGrowth.

Table 3: Evaluation parameter settings.

Group size m	Singapore	**3**, 4, 5, 6, 7, 8
	Chengdu	
Common cameras k	Singapore	2, **3**, 4, 5, 6, 7, 8
	Chengdu	
Gap tolerance d	Singapore	0, 1, 2, **3**, 4
	Chengdu	
Proximity ϵ	Singapore	40, 50, **60**, 70, 80, 90, 100
	Chengdu	
Object number	Singapore	1.5k, 1.7k, 1.9k, 2.1k, 2.3k, 2.5k, **2.7k**
	Chengdu	0.5k, **2.5k**, 4.5k, 6.5k, 8.5k, 10.5k, 12k
Average path length	Singapore	200, 300, 400, 500, 600, 700, **800**
	Chengdu	90, 100, 110, 120, 130, 140, **150**

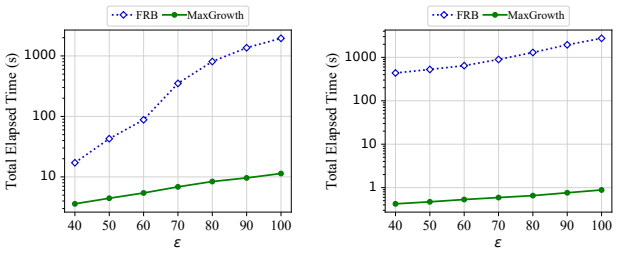


(a) Singapore (b) Chengdu

Figure 5: Varying m.

This is mainly due to the significant reduction in the number of total valid candidate patterns.

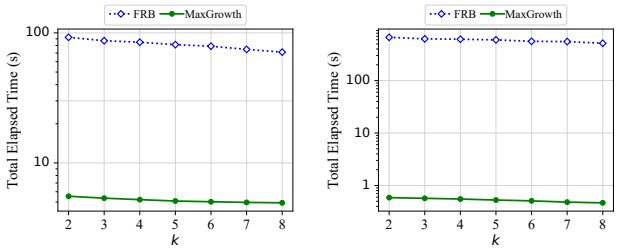
Varying ϵ. We examine the performance with increasing ϵ in Figure 6. MaxGrowth consistently outperforms FRB mainly because FRB requires heavy computation overhead to verify the constraint of temporal proximity. In contrast, the candidate enumeration scheme in MaxGrowth eliminates the verification cost. When ϵ is large, the search space grows dramatically. The widened performance gap between MaxGrowth and its competitor again validates the effectiveness of its enumeration and pruning schemes.



(a) Singapore (b) Chengdu

Figure 6: Varying ϵ.

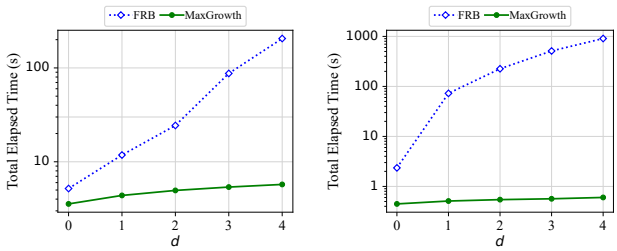
Varying k. As depicted in Figure 7, the MaxGrowth algorithm is not sensitive to the variation of k. This is because its candidate patterns are progressively enumerated with increasing route length. In other words, regardless of k, all candidate patterns that meet the group size and temporal proximity constraints are enumerated before undergoing the maximal pattern mining. The slight decline observed in the MaxGrowth algorithm is attributed to the reduction in the number of valid candidate patterns as k increases, leading to a lower computational cost for identifying maximal patterns. In contrast, FRB employs sequence mining to filter candidates with route lengths shorter than k, allowing more candidates to be excluded as k grows, thus reducing the overall mining overhead.



(a) Singapore (b) Chengdu

Figure 7: Varying k.

Varying d. Recall that we introduce an additional parameter d in this work to allow missing cameras for relaxed co-movement pattern mining. When $d = 0$, the problem is reduced to traditional co-movement pattern mining. FRB demonstrates comparable performance with MaxGrowth because FRB is customized for the setting with consecutive camera sequences. When the condition is relaxed, its running time increases remarkably, paying large amount of computation overhead for candidate verification. MaxGrowth achieves the best performance and its running time remains stable with increasing d. When $d = 4$, the time cost of MaxGrowth is two orders of magnitude lower than FRB.

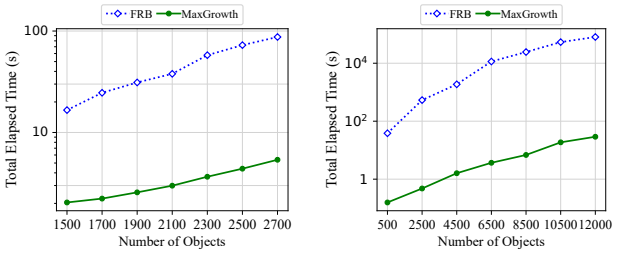


(a) Singapore (b) Chengdu

Figure 8: Varying d.

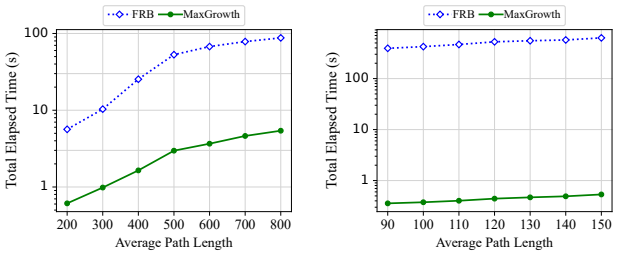
Varying Object Number. We continue to analyze the performance with respect to database-related parameters. The results for increasing dataset cardinality are presented in Figure 9. The outcomes share similar patterns with previous experiments — the running time increases with higher number of objects and the performance gap is widened with larger-scale dataset. In Chengdu, the running time of FRB is two orders of magnitude higher than MaxGrowth.

Varying Path Length. In Figure 10, we examine the performance with increasing travel path length for the moving vehicles. We construct subsets of Singapore and Chengdu with increasing path length. MaxGrowth is also scalable to this parameter and outperforms FRB with a large margin.



(a) Singapore (b) Chengdu

Figure 9: Varying object number.



(a) Singapore (b) Chengdu

Figure 10: Varying path length.

7.3 In-Depth Analysis of MaxGrowth

Running Time Breakdown Analysis. To further validate the effectiveness of the proposed techniques in MaxGrowth, we first perform a breakdown analysis on the running time of each component, including candidate generation, validness verification and dominance verification. Table 4 reports the time cost of each component in FRB and MaxGrowth. The cost of validness verification is the main performance bottleneck for FRB in both Singapore and Chengdu datasets. MaxGrowth can eliminate the requirement for candidate verification and thus save considerable amount of verification cost. Meanwhile, the candidate generation and dominance verification costs of FRB are also higher than those of MaxGrowth. This is because FRB cannot effectively avoid the search and generation of dominated non-maximal candidate patterns, leading to a larger search space during candidate generation and increased cost for dominance verification. In contrast, MaxGrowth significantly reduces these overheads through two pruning rules for maximal pattern mining.

Table 4: Running time breakdown analysis.

Datasets	Methods	Runtime of the breakdown stage (s)		
		Candidate Generation	Validness Verification	Dominance Verification
Singapore	FRB	20.2	43.8	12.1
	MaxGrowth	8.5	-	3.0
Chengdu	FRB	92.3	444.5	96.6
	MaxGrowth	5.3	-	2.3

Ablation Study. We conduct ablation experiments to further investigate the impact of pruning rules on MaxGrowth. We design three additional variants of MaxGrowth. Among these variants, MG-Root employs only the root pruning rule, MG-Dep uses only the dependency pruning rule, and MG-NoPrune does not utilize either pruning rule. As shown in Table 5, both the root pruning rule and the dependency pruning rule effectively prune redundant search subtrees, resulting in the clear performance improvement. Moreover, because the two rules prune at different levels of the search tree, MaxGrowth can further enhance efficiency. In the Chengdu dataset that contains numerous redundant patterns, while the root pruning rule achieves a 50% performance improvement through greatly pruning the search tree at the root node, many non-maximal patterns still emerge in subsequent search subtrees, which limits overall performance. In this scenario, the dependency pruning rule provides a decisive enhancement by enabling fine-grained pruning during pattern growth.

Table 5: Effect of candidate pruning.

Datasets	The number of non-maximal patterns			
	MG-NoPrune	MG-Root	MG-Dep	MaxGrowth
Singapore	2,097,851	1,402,269	59,329	31,067
Chengdu	15,961,467	8,568,710	5,424	2,075

7.4 Effectiveness Analysis

In this part, we perform quality analysis on the relaxed co-movement patterns discovered according to our problem definition. The experiment is conducted on the real video dataset CityFlow, which was designed for the task of multi-camera multi-object tracking. We use its annotated trajectories as groundtruth and derive golden co-movement patterns. F_1-score is used as the performance metric. A discovered co-movement pattern is considered a positive match with some golden pattern by default if the intersection over union (IoU) of their objects and time span are both greater than 80%.

Table 6: Comparison of pattern discovery effectiveness.

Pattern Type	F_1-score	Precision	Recall
VConvoy	0.719	0.861	0.617
VConvoy-TMerge	0.742	0.874	0.644
VPlatoon	0.840	0.826	0.855

To evaluate the effectiveness of VPlatoon and VConvoy, we perform these two algorithms on the trajectories recovered from CityFlow. Additionally, we construct a competitive baseline that applies the idea of TMerge [3] as a post-processing step to improve the accuracy of the recovered trajectories. Certain trajectory segments split by the missing observations can be merged according to spatial and temporal clues. The new baseline performs VConvoy against the refined dataset and we call it VConvoy-TMerge. As to the pattern parameters, we set $m = 2$, $k = 3$, $\epsilon = 60$, and $d = 3$. Since the input dataset is small-scale, it takes 0.03 seconds to run the TCS-tree algorithm for VConvoy pattern mining and 0.05 seconds to run MaxGrowth for VPlatoon pattern mining. However, VConvoy-TMerge incurs a considerable cost of 5.4 hours to obtain the refined dataset with parameters $\tau = 800{,}000$ and $K = 0.001$.

The F_1-score of discovered patterns are reported in Table 6. VPlatoon achieves significantly higher F_1-score, because VConvoy has missed many valid patterns and its recall is remarkably lower than

VPlatoon. In terms of precision, though the pattern is relaxed, the precision of VPlatoon is still comparable to VConvoy, implying that the number of false positive patterns introduced by pattern relaxation is limited. Moreover, there is a slight performance improvement for VConvoy-TMerge, yet its F_1-score still lags behind VPlatoon by nearly 10%. This implies that current trajectory accuracy improvement techniques alone may be insufficient for VConvoy to discover most of the missing patterns. In contrast, our proposed VPlatoon demonstrates greater effectiveness at a lower cost.

To better understand the relationship between our introduced parameter d and the patterns discovered by VPlatoon, we present in Figure 11 the comparison of VPlatoon-mined patterns with the ground truth across different values of d. As d increases, the precision gradually decreases while the recall increases, with the optimal F_1-score achieved at $d = 3$. This is because parameter d controls the degree of relaxation by adjusting the distance between cameras in the common route. Thus, a larger d enables the discovery of more patterns but also introduces potential irrelevant results which may reduce the precision.

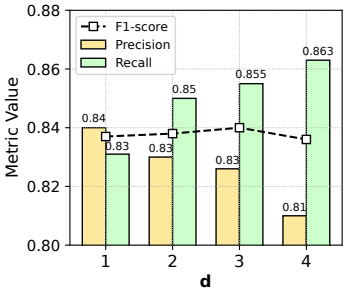


Figure 11: Pattern quality with varying d.

In the next examination, we examine the sensitivity to the accuracy of offline trajectory recovery algorithms. Specically, we inject noise into the recovered trajectories to control the degree of IDF1 [19], which is a popular metric for the accuracy of multi-object tracking. In terms of noise injection operators, we randomly perform trajectory shift, node deletion and ID switches to construct degraded trajectories. Specifically, the trajectory shift operator offsets time frame and bounding box values in a segment of the tracked trajectory, the node deletion operator removes a specific tracked segment, and the ID switch operator replaces the moving object of a segment of the tracked trajectory with another random object. Figure 12 presents the comparison of the VPlatoon and VConvoy-mined patterns with the ground truth at different accuracies of input trajectories. We observe that VPlatoon achieves a clear advantage in F_1-score for both low and high accuracy trajectories. Our relaxed definition of co-movement pattern can identify more high-quality patterns while introducing negligible irrelevant results.

7.5 Case Study

Finally, we present a case study in Figure 13 to further demonstrate the effectiveness of VPlatoon. It visualizes representative co-movement patterns from the CityFlow dataset with parameters $m = 2$, $k = 3$, $\epsilon = 12$s, and $d = 2$. More examples are also available in

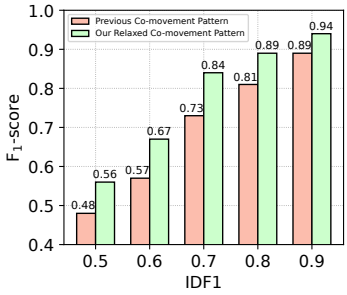


Figure 12: Pattern quality with varying trajectory accuracies.

our technical report [1]. In Figure 13, o_{398} experiences ID switches at both cameras c_{19} and c_{21} due to the differing viewpoints, causing VConvoy to detect two separate patterns. In contrast, VPlatoon correctly identifies that these two vehicles maintain close motion along the entire route between cameras c_{16} and c_{25}, confirming its discovery effectiveness under flawed video tracking and trajectory recovery algorithms.

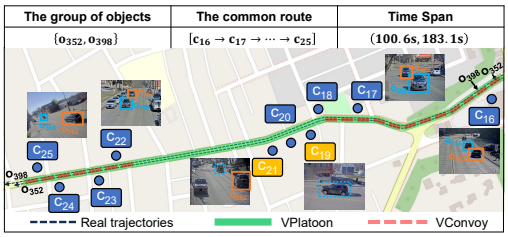


Figure 13: The visualization of a VPlatoon pattern from videos with vehicle mis-matching.

8 CONCLUSION

In this paper, we propose a relaxed definition of co-movement pattern from surveillance videos, which uncovers more comprehensive interesting patterns in inaccurately recovered trajectories. We devise a baseline based on previous video-based co-movement pattern mining. A novel enumeration framework called MaxGrowth is also proposed. It efficiently identifies all maximal relaxed co-movement patterns through eliminating the requirement for candidate verification and utilizing the further developed two pruning rules. Extensive experiments confirm the efficiency of MaxGrowth and the effectiveness of our proposed pattern definition.

In the future, it would be interesting to leverage probabilistic models to develop more general and robust definitions of co-movement patterns. In addition, distributed optimization for co-movement pattern mining from video data could also be a promising research direction.

ACKNOWLEDGMENTS

The work is supported by the National Key Research and Development Project of China (2022YFF0902000), the Fundamental Research Funds for the Central Universities (226-2024-00145, 226-2024-00216), and the Key Research Program of Zhejiang Province (2023C01037).

REFERENCES

[1] Yijun Bei, Teng Ma, Dongxiang Zhang, Sai Wu, Kian-Lee Tan, and Gang Chen. 2024. Mining Platoon Patterns from Traffic Videos. arXiv:2412.20177 [cs.CV] https://arxiv.org/abs/2412.20177

[2] Marc Benkert, Joachim Gudmundsson, Florian Hübner, and Thomas Wolle. 2008. Reporting flock patterns. *Computational Geometry* 41, 3 (2008), 111–125. https://doi.org/10.1016/j.comgeo.2007.10.003

[3] Daren Chao, Yueting Chen, Nick Koudas, and Xiaohui Yu. 2023. Track merging for effective video query processing. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 164–176.

[4] Lu Chen, Yunjun Gao, Ziquan Fang, Xiaoye Miao, Christian S Jensen, and Chenjuan Guo. 2019. Real-time distributed co-movement pattern detection on streaming trajectories. *Proceedings of the VLDB Endowment* 12, 10 (2019), 1208–1220.

[5] Lu Chen, Yunjun Gao, Xinhan Li, Christian S Jensen, and Gang Chen. 2017. Efficient Metric Indexing for Similarity Search and Similarity Joins. *IEEE Transactions on Knowledge and Data Engineering* 29, 3 (2017), 556–571.

[6] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. 2020. Mot20: A benchmark for multi object tracking in crowded scenes. *arXiv preprint arXiv:2003.09003* (2020).

[7] Xin Ding, Lu Chen, Yunjun Gao, Christian S Jensen, and Hujun Bao. 2018. UlTraMan: A unified platform for big trajectory data management and analytics. *Proceedings of the VLDB Endowment* 11, 7 (2018), 787–799.

[8] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, Vol. 96. 226–231.

[9] Qi Fan, Dongxiang Zhang, Huayu Wu, and Kian-Lee Tan. 2016. A general and parallel platform for mining co-movement patterns over large-scale trajectories. *Proceedings of the VLDB Endowment* 10, 4 (2016), 313–324.

[10] Joachim Gudmundsson and Marc Van Kreveld. 2006. Computing longest duration flocks in trajectory data. In *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*. 35–42.

[11] Yuhang He, Jie Han, Wentao Yu, Xiaopeng Hong, Xing Wei, and Yihong Gong. 2020. City-scale multi-camera vehicle tracking by semantic attribute parsing and cross-camera tracklet matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 576–577.

[12] Hoyoung Jeung, Heng Tao Shen, and Xiaofang Zhou. 2008. Convoy queries in spatio-temporal databases. In *2008 IEEE 24th International Conference on Data Engineering*. IEEE, 1457–1459.

[13] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S. Jensen, and Heng Tao Shen. 2008. Discovery of convoys in trajectory databases. *Proc. VLDB Endow.* 1, 1 (aug 2008), 1068–1080. https://doi.org/10.14778/1453856.1453971

[14] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. 2005. On discovering moving clusters in spatio-temporal data. In *Advances in Spatial and Temporal Databases: 9th International Symposium, SSTD 2005, Angra dos Reis, Brazil, August 22-24, 2005. Proceedings 9*. Springer, 364–381.

[15] Yuxuan Li, James Bailey, and Lars Kulik. 2015. Efficient mining of platoon patterns in trajectory databases. *Data & Knowledge Engineering* 100 (2015), 167–187.

[16] Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. 2010. Swarm: Mining relaxed temporal moving object clusters. *Proceedings of the VLDB Endowment* 3, 1-2 (2010), 723–734.

[17] Yiyang Liu, Hua Dai, Bohan Li, Jiawei Li, Geng Yang, and Jun Wang. 2021. ECMA: an efficient convoy mining algorithm for moving objects. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1089–1098.

[18] Faisal Moeen Orakzai, Toon Calders, and Torben Bach Pedersen. 2019. k/2-hop: fast mining of convoy patterns with effective pruning. *Proceedings of the VLDB Endowment* 12, 9 (2019), 948–960.

[19] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. 2016. Performance measures and a data set for multi-target, multi-camera tracking. In *European conference on computer vision*. Springer, 17–35.

[20] Peize Sun, Jinkun Cao, Yi Jiang, Zehuan Yuan, Song Bai, Kris Kitani, and Ping Luo. 2022. Dancetrack: Multi-object tracking in uniform appearance and diverse motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20993–21002.

[21] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David Anastasiu, and Jenq-Neng Hwang. 2019. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8797–8806.

[22] Panrong Tong, Mingqian Li, Mo Li, Jianqiang Huang, and Xiansheng Hua. 2021. Large-scale vehicle trajectory reconstruction with camera sensing network. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 188–200.

[23] Yongxin Tong, Yuxiang Zeng, Zimu Zhou, Lei Chen, Jieping Ye, and Ke Xu. 2018. A unified approach to route planning for shared mobility. *Proceedings of the VLDB Endowment* 11, 11 (2018), 1633.

[24] Gaoang Wang, Yizhou Wang, Renshu Gu, Weijie Hu, and Jenq-Neng Hwang. 2022. Split and connect: A universal tracklet booster for multi-object tracking. *IEEE Transactions on Multimedia* 25 (2022), 1256–1268.

[25] Jianyong Wang and Jiawei Han. 2004. BIDE: Efficient mining of frequent closed sequences. In *Proceedings. 20th international conference on data engineering*. IEEE, 79–90.

[26] Sheng Wang, Zhifeng Bao, J Shane Culpepper, Timos Sellis, and Xiaolin Qin. 2019. Fast large-scale trajectory clustering. *Proceedings of the VLDB Endowment* 13, 1 (2019), 29–42.

[27] Yida Wang, Ee-Peng Lim, and San-Yih Hwang. 2006. Efficient mining of group patterns from user movement data. *Data & Knowledge Engineering* 57, 3 (2006), 240–282.

[28] Munkh-Erdene Yadamjav, Zhifeng Bao, Baihua Zheng, Farhana M Choudhury, and Hanan Samet. 2020. Querying recurrent convoys over trajectory data. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, 5 (2020), 1–24.

[29] Xifeng Yan, Jiawei Han, and Ramin Afshar. 2003. Clospan: Mining: Closed sequential patterns in large datasets. In *Proceedings of the 2003 SIAM international conference on data mining*. SIAM, 166–177.

[30] Xipeng Yang, Jin Ye, Jincheng Lu, Chenting Gong, Minyue Jiang, Xiangru Lin, Wei Zhang, Xiao Tan, Yingying Li, Xiaoqing Ye, et al. 2022. Box-grained reranking matching for multi-camera multi-target tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3096–3106.

[31] 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 (nov 2023), 604–616. https://doi.org/10.14778/3632093.3632119