State Summarization of Video Streams for Spatiotemporal Query Matching in Complex Event Processing

Piyush Yadav
Lero- Irish Software Research Centre
National University of Ireland, Galway
Galway, Ireland
piyush.yadav@lero.ie

Dibya Prakash Das
Indian Institute of Technology, Kharagpur
IIT Kharagpur
Kharagpur, India
dibyadas@iitkgp.ac.in

Edward Curry
Lero- Irish Software Research Centre
National University of Ireland, Galway
Galway, Ireland
edward.curry@lero.ie

***Abstract*—Modelling complex events in unstructured data like videos not only requires detecting objects but also the spatiotemporal relationships among objects. Complex Event Processing (CEP) systems discretize continuous streams into fixed batches using *windows* and apply operators over these batches to detect patterns in real-time. To this end, we apply CEP techniques over video streams to identify spatiotemporal patterns by capturing window *state*. This work introduces a novel problem where an input video stream is converted to a stream of graphs which are aggregated to a single graph over a given state. Incoming video frames are converted to a timestamped Video Event Knowledge Graph (VEKG) that maps objects to nodes and captures spatiotemporal relationships among object nodes. Objects coexist across multiple frames which leads to the creation of redundant nodes and edges at different time instances that results in high memory usage. There is a need for expressive and storage efficient graph model which can summarize graph streams in a single view. We propose an Event Aggregated Graph (EAG), a summarized graph representation of VEKG streams over a given state. EAG captures different spatiotemporal relationships among objects using an Event Adjacency Matrix without replicating the nodes and edges across time instances. These enable the CEP system to process multiple continuous queries and perform frequent spatiotemporal pattern matching computations over a single summarised graph. Initial experiments show EAG takes 68.35% and 28.9% less space compared to baseline and state of the art graph summarization method respectively. EAG takes 5X less search time to detect pattern as compare to VEKG stream.**

Keywords*—*Video Streams, Pattern Detection, Complex Event Processing, Spatiotemporal Operators, Graphs Summarization

I. Introduction

The world is now transitioning to an era of Internet of Multimedia Things (IoMT) [1], where visual sensors prevail everywhere. These media capturing sensors produce data in streaming fashion from different sources like smartphones, IoT devices, social media platforms and are generating an unprecedented volume of video data. For example, cities like London and New York have deployed thousands of CCTV cameras, streaming hours of videos daily [2]. Machine learning techniques are used on these video streams to detect events of interest in different applications like business intelligence, surveillance, and traffic monitoring [3].

During the last decade, Complex Event Processing (CEP) systems have been increasingly adopted in different domains like traffic and financial applications [4] to identify event patte-

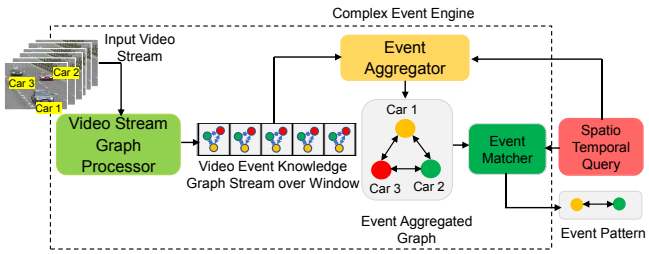

Fig. 1. System Overview: Video Stream Graph processor converts input video frames into graphs which are captured over a window of given time length. Event Aggregator fetches different spatiotemporal queries and summarizes the graph stream into an Event Aggregated Graph (EAG). The EAG captures all the required spatiotemporal relationships for the query which is later passed to Event Matcher for pattern detection.

-rns and send notifications in real-time. Both CEP and Data Stream Management System (DSMS) work on massive data streams. DSMS supports continuous transformation analytics over streams while the CEP system detects complex pattern using a combination of simple patterns over the streams [4]. In CEP, patterns are defined using the event rules which are encapsulated as an operator using SQL like declarative query languages. These CEP operators execute over streams to detect simpler atomic events and combine them to form more meaningful high-level semantics, i.e. complex events. Present CEP systems like Esper [5] and FlinkCEP [6] are focused on structured data processing, and less attention is paid to video event processing. As shown in Fig.1, we use CEP techniques for spatiotemporal video detection.

A. Motivating Scenario

Traffic surveillance is typically a manual effort where traffic personnel continuously monitors video stream feeds from different cameras installed in the city. This type of manual inspection is error-prone as well as challenging for humans to correlate multiple events at different time instances. As shown in Fig. 2, the traffic authority has declared busy routes in the city as 'no passing zones' during rush hour to prevent any traffic jam. They subscribe to a CEP engine for a '*Pass By*' pattern notification query (Q1) where a vehicle should not pass by another vehicle at a specific time and place. Fig. 2 shows a '*Pass By*' pattern [t=20-45], which is a complex event that constitutes atomic events like object(car) detection and its position with other objects at different time steps. Similarly, the authority may want to monitor high volume traffic flows (Q2) of a road at a given time of the day. Here *'high volume traffic'* event (*t=5*) is

978-1-7281-4550-1/19/$31.00 ©2019 IEEE
DOI 10.1109/ICMLA.2019.00022

composed of simple events like– a) Detection of 'Car' events and b) Counting the number of cars in each frame at different time instances.

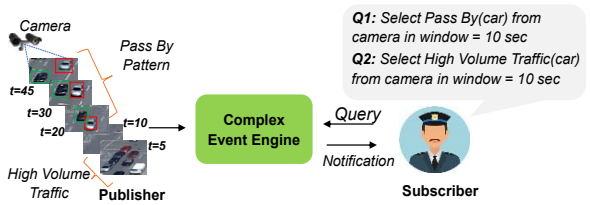

Fig. 2. Motivational scenario

In CEP systems there can be different continuous user queries (like 'High Volume Traffic' and 'Pass By') at different time instances. Within computer vision, Deep Neural Network (DNN) models need to be trained for pattern detection. It is challenging to train every pattern where requirement changes due to subscriber's query dynamicity. Training each pattern is costly in terms of resources and computation and is not feasible in the highly dynamic videos where objects are in motion and generate varying nature of events. We follow a hybrid approach where inductive reasoning methods like DNN models are used for simple event detection (like object detection). Later a deductive reasoning approach is used to craft complex event rule patterns.

B. Problem Statement and Contribution

The objects in the video relate to the other objects across space and time and generate a complex network. In this work, we represent input video streams as a time-stamped graph termed as Video Event Knowledge Graph (VEKG) using DNN models. The same object can exist in multiple frames leading to the creation of thousands of nodes in just a few minutes of video. This creates storage and a computational bottleneck for executing queries. The contribution of work are as follows:

- We propose an Event Adjacency Matrix (EAM) to capture spatiotemporal relationships between objects.
- We propose Event Aggregated Graph (EAG), a summarized representation of VEKG streams which explicitly captures the spatiotemporal changes of the objects over the CEP window without any loss of information using 68.35% less storage space. EAG handles multiple queries relationship in a single summarized graph.
- We demonstrate different spatiotemporal query operators related to the traffic management domain to show the efficacy of EAG with 82.7% faster query execution.

The rest of the paper is organized as follows: Section II presents the basic concepts. Section III discusses relationships, event extraction and representation for video streams. Section IV presents traffic management related operators while Section V focusses on Event Aggregated Graph (EAG). Section VI shows the experimental evaluation, while Section VII discusses related work. Section VIII concludes the paper.

II. Preliminaries

A. Windows: A stateful operator in CEP

Streams are an unbounded sequence of data items that are continuously evolving. CEP systems work over the concept of *state*, which is the snapshot of the stream. CEP and other DSMS systems use *windows* operator to capture the state of the stream. Windows are stateful operators which discretize the continuous stream into fixed batches and apply computations over these sequence of input data [7]. Fig. 3 shows an example of a time window of 5 seconds which performs an aggregation query (average price >9) over a stock stream of company 'X'. Window (W) between time (T) 5-10 satisfies the query as the average price is greater than 9 (*Avg.(W)* =10). In this work aggregation over windows are performed over video graph streams to detect complex spatiotemporal patterns. From here, we will be using the terms aggregation and summarization interchangeably.

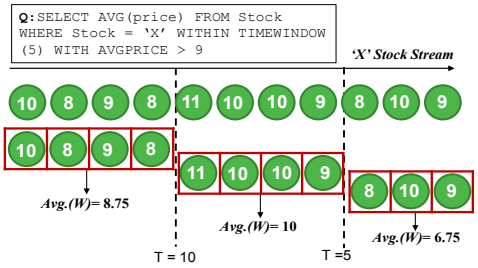

Fig. 3. Aggregation (Avg.) over a time window

B. Object Detection and Image Representation

Videos are timestamped continuous sequence of image frames, which consists of objects. Vision algorithms like SIFT [8] (Fig.4 (a)) have been proposed to detect objects by processing low-level feature of images. Recently, Deep Neural Networks (DNN) based methods have evolved to analyze images and videos with good accuracy and performance. DNN based object detection models like YOLO [9] (Fig. 4 (b)), and M-RCNN [10] gives bounding boxes across the objects in the images which are highly accurate. Similarly, relation detection models like Scene Graphs [11] describe the relationship between the objects like *'girl holding racket'* where '*girl'* and '*racket'* are objects while 'holding' defines a relationship between these two objects. (Fig. 4 (c)).

Fig. 4. Object detection technique- (a) SIFT (b) YOLO and Image representation technique- (c) Scene Graphs [11]

III. Formal Concepts for Video Representation

In this section, video representation and formal spatial constructs are defined which are used to create complex spatial relationships among video objects across time.

A. Video Stream Extraction and Representation

Video streams have no fixed data model, and it is difficult to detect patterns without any structured representation. There is a need to extract video content and convert it to a representation which can handle a complex relationship at different granularity ranging from low-level feature-based properties to high-level semantics. From a representation perspective, we have divided video features into two aspects:

- *Objects:* Objects are the basic building block of multimedia (video, images) data which are a collection of low-level features and have been given a high-level semantic label (like Car, Person). These objects can have multiple characteristics and properties which are represented as its attributes (e.g. color, type). We assume that an event occurs in the video if an object exists.
- *Relationship between objects:* The objects in video frames interact with other objects within the frame (intraframe) and across frames (interframe). In intraframe relationships, objects interact with other objects with a spatial relation as they occupy a specific position within the frame while in interframe, the objects interact temporally.

Objects and relationships are mapped to an event-centric representation using entity-centric knowledge graph. The input video frames are converted to a Video Event Knowledge Graph (VEKG) representation, where nodes correspond to objects and edges represent spatial and temporal relationships among objects [12], [13]. Thus, VEKG can be defined as:

Definition1 (VEKG Graph): For any image frame, the resulting Video Event Knowledge Graph is a labelled directed graph with six tuples represented as VEKG = $\{\mathbf{V}, \mathbf{E}, \mathbf{Av}, \mathbf{R_E}, \boldsymbol{\lambda_v}, \boldsymbol{\lambda_E}\}$ where

$\mathbf{V}$ = set of object nodes $\boldsymbol{O_i}$

$\mathbf{E}$ = set of edges such $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$

$\mathbf{Av}$= set of properties mapped to each object nodes such that

$\boldsymbol{O_i}$= (id,attributes, label, confidence,features)

$\mathbf{R_E}$= set of spatiotemporal relations classes

$\boldsymbol{\lambda_v}, \boldsymbol{\lambda_E}$ are class labelling functions -$\boldsymbol{\lambda_v}: \mathbf{V} \rightarrow \boldsymbol{O}$ and $\boldsymbol{\lambda_E}: \mathbf{E} \rightarrow \mathbf{R_E}$

Definition2 (VEKG Stream): A Video Event Knowledge Graph stream is a sequence ordered representation of VEKG such that $\boldsymbol{VEKG}\ (\boldsymbol{S}) = \{(\boldsymbol{VEKG_1}, \boldsymbol{t_1}), (\boldsymbol{VEKG_2}, \boldsymbol{t_2}) \ldots\ldots (\boldsymbol{VEKG_n}, \boldsymbol{t_n})\}$ where $\boldsymbol{t\epsilon\ timestamp}$ such that $\boldsymbol{t_i} < \boldsymbol{t_{i+1}}$.

Fig. 5 shows the architecture for the construction of VEKG stream from input video frames. The frames are passed to a video frame decoder to convert it into a feature matrix which is later passed to DNN cascade for object and attribute classification. In the DNN cascade, the YOLO [9] object detector model fetches an input feature matrix from the video frame decoder and passes the detected object's region of interest (ROI) features to an attribute classifier. For example, in Fig. 5 the Frame (T1) has an object 'Car1' which has a color attribute 'Black'. The Deep SORT [14] object tracking algorithm is used to keep track of the same objects across multiple frames. The detected objects and its attributes are then passed to a graph constructor to create a VEKG graph for each frame, which is then captured over CEP windows. Fig. 5 shows VEKG graphs in a CEP window for three video frames at different time instances. The object nodes (label-Car1, Car2, Car3) in VEKG graphs are connected using spatial edges. VEKG is a complete digraph, which means that each object is spatially related to another object which is present in the image frame. The temporal relation edge between object nodes is created by identifying the same object nodes in different frames using object tracking. The spatial relationship weights across VEKG edges are stored in an Event Adjacency Matrix which is discussed in detail in Section III-D.

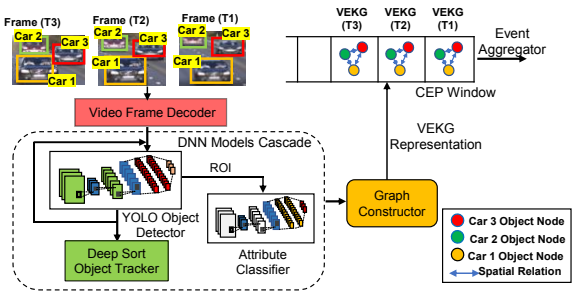

Fig. 5. Video Event Knowledge Graph (VEKG) construction architecture

B. Spatial Relation

The interaction among objects in videos can happen in a spatial dimension in which relationships can be modelled using spatial calculus. There are different spatial calculus like topological, directional, and orientation, to capture these interactions [15]. We have categorized spatial relations into three main classes:

1) Geometric Relation for Spatial Object ($\boldsymbol{O}$): A spatial entity can be represented using geometry-based features like point, line and polygon. Current DNN models detect objects from images by either creating bounding box [9] or by creating a segmented region across these objects' boundaries [10] (Fig. 6). In this work, bounding box based polygons are used to represent the objects in video frames.

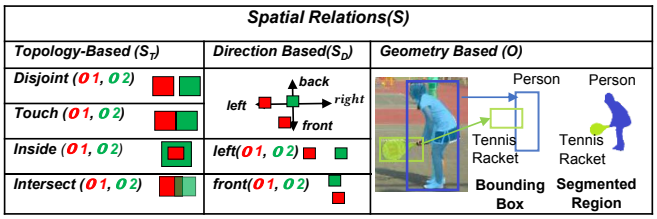

Fig. 6. Spatial relations between objects

2) Topology-Based Spatial Relation ($\boldsymbol{S_T}$): We have used the Dimensionally Extended Nine-Intersection Model (DE-9im) a 2-dimensional topological model which describes pairwise relationships between spatial geometries (O). This mathematical model is based on Clementini Matrix [16], which describes relationships between geometries based on their interior(I), boundary(B), and exterior(E) features. The nine relationships it captures are- *{Disjoint, Touch, Contains, Intersect, Within, Covered by, Crosses, Overlap, Inside}*. The four topological relations are shown in Fig. 6 using a bounding box *(O)*.

3) Direction Based Spatial Relation ($\boldsymbol{S_D}$): Direction captures the projection and orientation of an object in space. We use the Fixed Orientation Reference System (FORS) [17] which divides the space into eight regions: {*front, back, left, right, left-front, right-front, left-back, right-back*}. As shown in Fig. 6, we took a simplified version of FORS with only four major direction relations, i.e. *{left, right, front, back}*.

C. Spatial Functions

Spatial functions quantify spatial relations among objects by calculating numerical values. These numerical values establish

different association among objects. We have devised two types of spatial functions:

1) Boolean Spatial Function (bsf): The function calculates boolean value between spatial relation.

$$\boldsymbol{bsf \rightarrow s_{T/D}(O_1, O_2)} = \begin{cases} 0 \ \ if\ spatial\ relation\ is\ false \\ 1\ if\ spatial\ relation\ is\ true \end{cases}$$

For example, for a topological relation (S_T) '*Overlap*' the boolean spatial function for two objects O_1, O_2 will be *bsf* (*Overlap* (O_1, O_2)) = 0 or 1.

2) Metric Spatial Function (msf): *msf* gives the metric value between the objects:

$$\boldsymbol{msf \rightarrow \mathcal{M}(O_1, O_2)\ |C|\ r}$$

where $C \in comparison\ operator$ ($<, >, \leq, \geq, =$) and $r \in \mathbb{R}$ (real number) and $\mathcal{M} \in metric\ function$ like *distance* and *count*. For example, *msf* (*distance*(O_1, O_2)) = 5. The spatial calculation is performed from the bounding boxes of the object, taking the image frame of the object as the spatial reference.

D. Event Adjacency Matrix

The VEKG graph stores spatial relationships between object nodes in an Event Adjacency Matrix (EAM). It is a NXN matrix where N represents object nodes such that matrix element a_{ij} represents the relation between $object\ i\ and\ j$. EAM is a square matrix with all its diagonal element having values zero. This is because we are not considering any relation of an object with itself. As defined in Section III-C, spatial function (*bsf* and *msf*) is applied over object nodes to calculate the relationships among them. There are two types of EAM:

1) Intraframe Event Adjacency Matrix: Intra-EAM captures the spatial relationships of object nodes within the image frame. Fig. 7 shows four Intra-EAM (E_{t1},E_{t2}, E_{t3} and E_{t4}) for the frame at time t1, t2, t3 and t4. The matrix shows *left* boolean spatial function (*bsf*) applied over objects O_1, O_2 and O_3. The function compares *left* direction relation like- '*Is the object O_1 is left with respect to the object O_2*' and give a boolean answer in 0 and 1.

2) Interframe Event Adjacency Matrix: The Inter-EAM captures the spatial relationship among objects between two-time instances. The Inter-EAM is formed by applying the XNOR over Intra-EAM, which is motivated from the work of target adjacency matrices [18]. The XNOR function returns 1 if there is no change in the relative spatial position of objects else it returns 0 if there is a relative change in spatial position among objects at different time instances. Fig. 7 shows an Inter-EAM($E_{t3\text{-}t4}$), which is an XNOR result of E_{t3} and E_{t4}. In a row-wise analysis, it can be seen that O_1 has changed its relative spatial position (left) with respect to object O_2 as evident in the frames. In the video, the objects may appear and disappear after a certain time. To handle such situations where two Intra-EAM matrices have some identical and different objects, a don't care (X) condition is applied among different objects. In Fig.7, frame(t1) does not have an object O_2 as in frame(t2). Thus, a don't care (X) condition is present in (E_{t1}) since the object O_2 is having no relation in frame(t1). The different use cases of Inter-EAM is discussed in the next section.

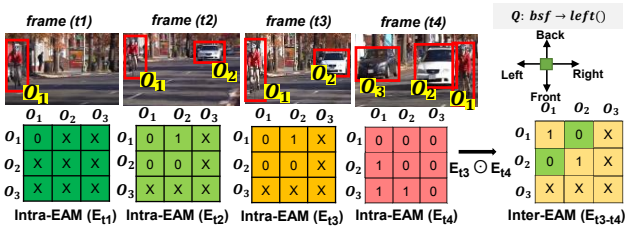

Fig. 7. Event Adjacency Matrix

IV. SPATIOTEMPORAL QUERY OPERATORS IN TRAFFIC MANAGEMENT

Videos stream analysis can be used to identify events of interest in traffic management like object identification (car type, car color, license plate recognition) and other spatiotemporal event patterns. In this section, spatiotemporal operators related to traffic management are discussed, which later act as queries for detecting event patterns.

A. Pass By

The query operator '*Pass By'* is defined as a '*change in the relative position of the object (back-front) in the same direction of motion'*. In Fig. 8, two frames of a video are shown at time t_i and t_{i+j} such that $t_i < t_{i+j}$. We see that the relative position of the object o_1 was '*back'* of o_2 at t_i which becomes '*front*' at t_{i+j}. This signifies that object o_1 crosses the o_2 in i+jth time instance. Thus, as per eq.1, we can write the '*Pass By*' operator as:

$$[\boldsymbol{back}((o_1, o_2)^{t_i}) \rightarrow \boldsymbol{front}((o_1, o_2)^{t_{i+j}})]^{\boxplus[t_m, t_n]} \quad (1)$$

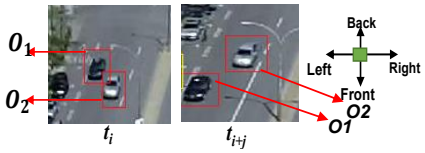

Fig. 8. Pass By scenario

Now, let us deduce the detailed version of equation 1, which will be applied over video stream to detect the '*Pass By*' pattern:

$$\exists(\boldsymbol{t_i}) \in \boldsymbol{T\ if}\ [\boldsymbol{bsf}(\boldsymbol{S_D}(o_1, o_2)\boldsymbol{x}^{t_i}) \odot \boldsymbol{bsf}(\boldsymbol{S_D}(o_1, o_2)\boldsymbol{x}^{t_{i+j}})]^{\boxplus[t_m, t_n]}$$

$$= \begin{cases} \boldsymbol{0\ if\ Pass\ By} \\ \boldsymbol{1\ if\ no\ Pass\ By} \end{cases} \quad (2)$$

$$\boldsymbol{where\ x \in back - front\ direction\ and\ t_m \leq t_i and\ t_{i+j} \leq t_n}$$

In equation 2, $bsf(S_D(o_1, o_2)x^{t_i})$ means the boolean spatial function over spatial direction (S_D) on the object (o_1, o_2) at time t_i where direction we are looking is in *back -front*. This will evaluate as '*Is o_1 back of o_2 in back front direction*', which is true, so it will return 1. Similarly, $bsf(S_D(o_1, o_2)x^{t_{i+j}})$ is the calculation for the next frame at time t_{i+j}. In this case the relative position of the object o_1 become front of the object o_2, so 'o_1 back of o_2' become false and returns 0. If we do an XNOR ($\odot$) of these two values, i.e. 1 at t_i and 0 at t_{i+j}, then we get 0 which as per equation (2) means '*Pass By'*. If there was no change in the relative position of objects in the given direction, then XNORing will return 1, which means no passing between the two objects. The evaluation of each frame was done in a time window $\boxplus[t_m, t_n]$. So, for any time instance in this time range if we get 0 between consecutive frames of objects, then we say there is a pass by between these objects.

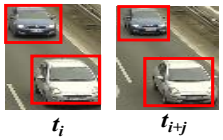

Fig. 9 . Follows scenario

Fig. 10. High Volume Traffic scenario

B. Follows

'*Follows'* is defined as '*no change in the relative position(front-back) of an object in the same direction of motion'*. This can be defined as:

$$[\boldsymbol{back}((o_1, o_2)^{t_i}) \rightarrow \boldsymbol{back}((o_1, o_2)^{t_{i+j}})]^{\boxplus[t_m, t_n]} \quad (3)$$

'*Follows'* is like '*Pass By'* with a difference that for every time instance in a given window, there is no change in the relative position of objects in consecutive frames. Thus, if eq. 2 results 1 for two different time instances then it means o_1 monotonically *follows* o_2 (see Fig. 9).

C. Lane Change

'Lane Change' is defined as a '*change in relative position (left-right) of an object in the same direction of motion'*. This is defined:

$$[\boldsymbol{left}((o_1, o_2)^{t_i}) \rightarrow \boldsymbol{right}((o_1, o_2)^{t_{i+j}})]^{\boxplus[t_m, t_n]} \quad (4)$$

As shown in Fig7., o_1 was *left* of o_2 at time t2 which later become *right* changing its lane at time t4.

D. High Volume Traffic

'*High Volume Traffic'* query operator is defined as '*the average count of objects at a given space is greater than a certain threshold for a specific time range*'. For example, if there are more than eight cars at a specific location of the road for more than 5 minutes, then we termed it as high-volume traffic for that location (Fig. 10). *High Volume Traffic* is defined as

$$\exists \eta \in G \text{ and } \forall ti \in T \; if$$

$$\boldsymbol{msf}\left(\mathcal{M}(\mathbf{O})_{\eta}^{\boxplus[t_1,t_2]}\right) = \begin{cases} > r \;\; \boldsymbol{traffic} \\ < r \;\; \boldsymbol{not\; traffic} \end{cases} \quad (5)$$

$$\boldsymbol{where\; G\; is\; a\; space\; and\; T\; is\; time\; such\; that\; t1 \leq ti \leq t2\;\; and\;\; \mathcal{M} = Avg.COUNT, O = object\;\; and\; r \in \mathbb{Z}}$$

In eq. 5, a metric spatial function (*msf*) $\mathcal{M}$ is applied which counts the average number of objects in every frame for a time window of $\boxplus$ [t1, t2] for a specific location (η).

In the next section, we discuss graph aggregation, which caters to multiple query operators.

V. Query Operator Based State Summarization

CEP queries are continuous, which means once the query is registered, it will continuously monitor the incoming streams over a given window state and detect pattern as per the operator rule. Multiple queries with different operators (like 'Pass By' and 'High Volume Traffic') can be registered to the system at any instance of time. As discussed in Section IV, multiple queries have different event patterns and matching all queries over a VEKG stream is time-consuming. Following the concept of state aggregation over windows (Section II-A), we summarize VEKG stream over windows into an Event Aggregated Graph (EAG) to handle different spatiotemporal queries.

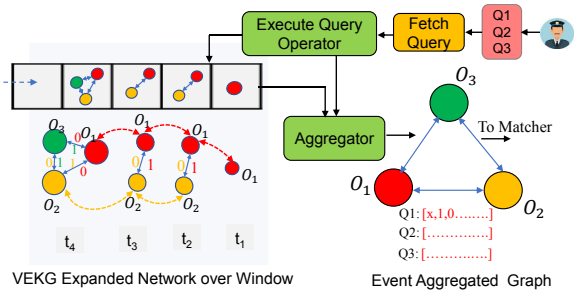

Fig. 11. Event Aggregated Graph construction

A. Event Aggregated Graph

In videos, object stay in different frames for some time, creating multiple redundant instances across time. This not only leads to redundancy and storage inefficiency but also increases the matching time as a query needs to traverse the same node across different time instances. For example, Fig. 11 shows an expanded network of VEKG graph for four frames (Fig.7) over a window. The expanded graph edges have binary weights represented from Intra-EAM shown in Fig.7. A single object node (like red) is repeated in all-time instances because of its presence in all four frames leading to redundant nodes and edges. A video streaming at a rate of 30 fps with two objects in every frame will lead to the creation of thousands of nodes in minutes, increasing the network size. Such large networks lead to computational bottlenecks for different query operators. Thus, storing and matching over VEKG graphs is expensive and time-consuming as it models every incoming video frame.

We present Event Aggregated Graphs (EAG), a summarized representation of VEKG streams over a window. We extend the Time Aggregated graph (TAG) model [19] for EAG by adding multiple queries over the edge for a given window length. EAG can handle multiple continuous queries using a single representation where objects span across time and space. EAG not only gives an aggregated view of a state but also preserves all required relationships which need to be captured by different query operators. It does not replicate redundant nodes and edges of VEKG across time and captures the spatiotemporal properties of VEKG graphs using Intra and Inter-EAM matrices. EAG reduces redundancy and is a storage efficient representation to model dynamic networks like video streams where objects are mostly in motion changing their relationship with each other with time. EAG keep tracks of spatiotemporal changes as per query operators.

Definition(EAG Graph): For a given window length *T*, having *n* video frames represented as VEKG graph, the Event Aggregated Graph is a labelled directed graph with 8 tuples such that $EAG = \{\boldsymbol{V_{Node}}, \boldsymbol{E_{Edge}}, \boldsymbol{W_{Time}}, \boldsymbol{fq}, \boldsymbol{t}, \boldsymbol{w}, \boldsymbol{\lambda_v}, \boldsymbol{\lambda_E}\}$ where:

- $\boldsymbol{V_{Node}}$ = set of unique object nodes $\boldsymbol{O_i}$ from VEKG stream over a window
- $\boldsymbol{E_{Edge}}$ = set of edges such $\boldsymbol{E_{Edge}} \subseteq \boldsymbol{V_{Node}} \times \boldsymbol{V_{Node}}$
- $\boldsymbol{W_{Time}}$ = window time length
- $\boldsymbol{fq}$ = different query mappings over $\boldsymbol{E_{Edge}}$
- $\boldsymbol{t}, \boldsymbol{w}$ = time and spatiotemporal weight mappings over $\boldsymbol{fq}$
- $\boldsymbol{\lambda_v}, \boldsymbol{\lambda_E}$ are class labelling functions -$\boldsymbol{\lambda_v}$: $\boldsymbol{V_{Node}} \rightarrow \boldsymbol{O}$ and $\boldsymbol{\lambda_E}$: $\boldsymbol{E_{Edge}} \rightarrow \boldsymbol{fq}$

In Fig. 11, the Execute Query operator fetches all queries registered in the system. It updates the VEKG graph edges over a window with spatiotemporal weights by executing operator logic using an Event Adjacency Matrix. Later, the Aggregator performs an aggregation operation over the window to update the VEKG stream to create an EAG. Fig. 11 shows an EAG (right-side) of an expanded network. It consists of three unique object nodes (o_1, o_2, o_3) which were present in VEKG expanded network. EAG is a complete digraph which captures the relationship among all object nodes. The edge captures the spatiotemporal weights of different queries at different time instances. For example, the edge between object o_1 and o_2 captures relation for three queries Q1, Q2, and Q3. Q1 is a left query discussed in Fig. 7 and it captures the Inter-EAM values by XNORing Intra-EAM matrices, i.e. E(t1) ⊙ E(t2), E(t2) ⊙ E(t3), and E(t3) ⊙ E(t4) resulting in {X,1,0,…..} time-series vector representation. The length of this value vector is equivalent to the number of frames in window time T. The 'X' is a don't care condition meaning there is no spatial relation between o_1 and o_2 at time t1-t2. Later '1' represents that o_1 is *left of* o_2 at t2-t3 and becomes 0 at t3-t4 representing that o_1 is no longer in the *left* of o_3 representing a lane change. Similarly, other queries like Q2 and Q3 are given spatiotemporal relation weights at different time instances. The spatiotemporal weights can be a real number if a metric spatial function (*msf*) is applied or expressed using other formats depending on operator rule.

Algorithm 1: Event Aggregated Graph

Input:
Window (Win) = $\{VEKG_1, VEKG_2, VEKG_3 \dots\dots\dots, VEKG_n\}$
Query (Q) = $\{Q_1, Q_2, Q_3 \dots\dots\dots, Q_n\}$
Output:
Event Aggregated Graph (EAG) over *Win*
Procedure:
$EAG \leftarrow intializeEAG(V, E, fq, t, w)$
for *each* $VEKG_i$ in *Win* **do**
  $V \leftarrow V \cup \{getnodes(VE_{KG_i})\}$
  for *each* Q_i *in Query* **do**
    $w_i \leftarrow SpatiotemporalOperator(Q_i(VEKG_i))$
    $fq_i \leftarrow t_i(w_i)$
    E$\leftarrow fq_i$
    $EAG \leftarrow UpdateEAG(V, E)$
  end
end
SendtoMatcher (EAG)

The EAG construction process is shown in Algorithm 1. The EAG is then passed to an *Event Matcher* for pattern detection. The matcher fetches the edge query vector of spatiotemporal weights between objects to detect the pattern. For example, Q1 = *left* (o_1, o_2) will fetch [x, 1, 0, …] and send a notification that at time t3-t4 as o_1 changes its position with o_2. This can be verified by looking at the frames in Fig. 7. The complex matching can be performed among multiple objects by traversing the path among the object nodes. The adjacency matrix-based graph representation requires $O(n^2)$ memory where n= $|\boldsymbol{V_{Node}}|$, i.e. number of nodes. EAG has two extra dimensions, i.e. time and query. If $\boldsymbol{W_{Time}}$ be **T,** and the number of queries is **Q** then EAG would require $O(n^2QT)$ memory to represent the video stream over a window.

VI. Experimental Results

A. Implementation and Datasets

The above prototype is implemented in Python 3. All the experiments were performed on a 16-core AMD Ryzen 7 1700 Linux machine with a 16 GB of RAM running on a 3.1 GHz processor and a Nvidia Titan Xp GPU. OpenCV is used for initial video frame decoding. For object detection and tracking pre-trained YOLOv3 [9] model with Deep SORT [14] algorithm is used. For attribute extraction, the features based on bounding box coordinates were fetched from the YOLO model layer and passed to the attribute classifier, which is an OpenCV based color filter. NetworkX [20], a python library for graphs was used for VEKG and EAG graph construction.

A small video dataset was created by cropping video clips of event patterns based on operators identified in Section IV. The videos were extracted from websites YouTube, Newsflare, and the DETRAC [21] dataset. Most of the videos are streamed at an average rate of 25 frames per second. The ground truth data for events was created manually by identifying different event patterns which act as a baseline for the comparison.

B. Evaluation Results

1) Reduction in Nodes, Edges and Storage: The summarization effectiveness can be known by calculating the reduction in nodes and edges without any loss of information. Reduction in Nodes (RIN) is defined as the ratio of the difference in the number of nodes between original ($|v_{VEKG}|$) and summarized graph($|v_{EAG}|$) with the number of original graph nodes (eq. 6). Similarly, Reduction in Edges (RIE), can be defined by replacing the number of nodes with edges (eq. 7).

$$RIN = \frac{|v_{VEKG}| - |v_{EAG}|}{|v_{VEKG}|} \quad (6) \qquad RIE = \frac{|E_{VEKG}| - |E_{EAG}|}{|E_{VEKG}|} \quad (7)$$

Fig. 12 shows RIN of EAG with its variant Time Aggregated Graph (TAG) [19]. The comparison is made with VEKG graphs over a window of 10 sec and for four queries over 12 video streams. The average EAG RIN score (red line) across videos is above 0.975, which means greater than 97.5% of VEKG nodes have been reduced to create an EAG summary. The average TAG RIN score (green line) lies between 0.91 to 0.95 across videos. Thus, TAG only reduces 90%- 95% of VEKG nodes. The reason is TAG creates a different summary graph for each of the queries (4 here) while EAG handles all the queries in a single summarized graph. The spike in video V4 means that the number of objects is high, leading to more object nodes and edge creation. Similarly, Fig. 13 shows the RIE score where EAG reduces greater than 96% of VEKG edges, while TAG reduces between 82.2% to 95.8% of edges. Thus, EAG reduces 5.1% nodes and 8.1% of edges as compared to TAG and will perform better with an increase in the number of queries.

In CEP, window states are stored in a state backend for historical analysis. Summarization of states will not only lead to storage efficiency but also faster retrieval and search. Fig. 14 compares the storage cost (KB) for all three representation. For the given experiments, i.e. 12 videos and four queries, TAG takes 55.4% while EAG takes 68.35% less storage space as compared to VEKG graphs. Similarly, EAG requires 28.9% less storage space as compare to TAG as it maintains only a single representation for multiple queries.

2) Graph Construction and Search Time: Graph construction is the time to create the VEKG graphs over a window. This includes graph initialisation with nodes and pop-

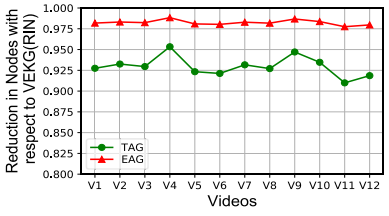

Fig. 12. Reduction in nodes (RIN) for different video streams

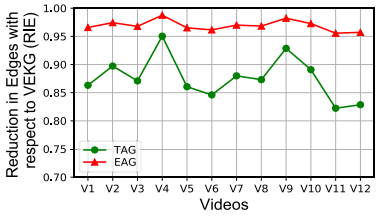

Fig. 13. Reduction in edges (RIE) for different video streams

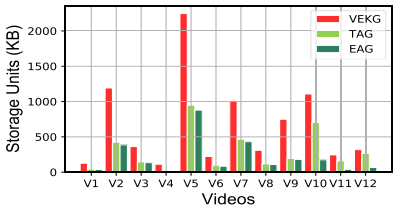

Fig. 14. Storage comparison for summarized EAG over different video streams

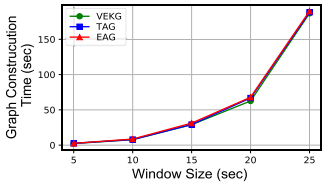

Fig. 15. Graph construction time with the change in window size

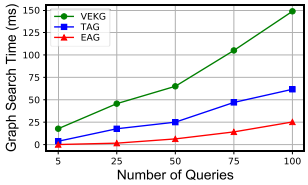

Fig. 16. Graph search time over multiple queries

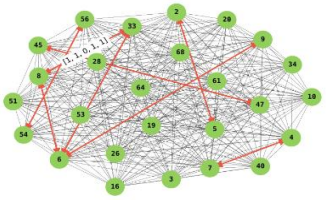

Fig. 17. Visualization of an EAG summarised graph

Table 1. Accuracy of different spatiotemporal operators

Query	Precision	Recall	F-Score
Pass By	0.733333	0.833333	0.777778
Follows	0.758772	0.776786	0.751082
Lane Change	0.663636	0.761905	0.700767
High Traffic	0.916667	0.816667	0.861111

Fig. 18. Follows pattern(V8)

Fig. 19. Pass By pattern(V4)

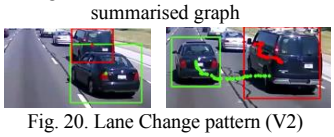

Fig. 20. Lane Change pattern (V2)

Fig. 21. High &Low Traffic Flow (V11)

-ulating edges with relation labels defined as per query logic. The process will be repeated 'n' times if 'n' number of queries are registered with the system. EAG summarization is achieved over these VEKG graphs for given queries. In Fig 15, the summarization overhead for EAG construction is in sub-seconds as compared to VEKG because all the operations have been already performed over VEKG and only the time to initialise the summarized graph is required. In Fig. 15, on increasing the window size the graph construction time increases because the number of nodes and edges will increase with time, but the summarization process for EAG and TAG is nearly same with respect to VEKG graphs. Fig 16. shows the search time of EAG, TAG and VEKG over a different number of registered queries. The EAG search shows the advantage of summarisation as it takes less time with respect to TAG and VEKG graphs. For five queries, the search time of EAG and TAG is nearly the same, as only 5 TAG graphs were created, but its search time will increase with more queries. For 100 queries the EAG search requires only 25.6 ms as compared to TAG and VEKG which have search times of 61.7 ms and 148.6 ms respectively.

3) Query Accuracy: The query accuracy examines how many relevant event patterns were detected for each query as compared to the ground truth. Query accuracy is evaluated using F-score (eq. 8), which is the harmonic mean of precision and recall.

$$F-score = \frac{2 * Precison * Recall}{Precision + Recall} \quad \textbf{(8)}$$

Table 1 shows the mean precision, recall and F-score for different queries, which is averaged across a window of 10 seconds. The first three queries, i.e. Pass By, Follows and Lane Change are based on the spatial directions of objects and have an F-score of 0.77, 0.75 and 0.70 respectively. Lane Change has a lower F-score (red color) because of occlusion and changes in the aspect ratio of objects, leading to many false positives. High Volume Traffic is a count-based query, and thus counting the number of objects depends on DNN models performance. YOLO has state of the art object detection performance. Thus, the proposed system can detect objects with good accuracy leading to high F-score of 0.86 for high volume traffic query.

Fig. 17 shows the visualization of an EAG for a 'Pass By' query. The green nodes are different unique objects present in the video in a window. The black edges are the spatial relationships existing between different object nodes while the red edges show the relations satisfied for a given query. For example, the sample red edge label [1,1,0,1,1] between object nodes 8 and 33 shows that there is 'Pass By' at t3-t4 as the value is 0. The visualization of 'Follows' is shown in Fig. 18, where color dots shows the tracking point of objects at different time instances. No tracking points of objects have crossed another object, while in Fig. 19 green dots crossed over pink dots leading to a 'Pass By' pattern. Fig. 20 shows 'Lane Change' where green dots move to left with respect of red dots. Similarly, Fig. 21 shows the high and low traffic flow.

C. Limitation

Our work has some limitations which are as follows- 1) DNN models are a basic building blocks in our CEP system, and any prediction failure in them will decrease our CEP engine's performance, 2) The spatial calculation was performed in the 2-dimensional plane while in the real world the relations are complex and spread in 3-dimensions, leading to many patterns miss or false event detection, 3) System works well over fixed camera with correct orientation.

VII. Related Work

A. Graph Summarisation

George et al. proposed TAG [19], an aggregated data model for spatiotemporal networks. EAG is an extension to the above work where we added multiple query relation to the edges of the summarized graph over a window for detecting complex video

event patterns. Lee et al. [22] focused on spatiotemporal indexing of video streams instead of summarization. Kwon et al. [23] detect rare events in videos using a graph editing framework and minimize it using the predefined energy model. They decompose video into a graph where a node represents a spatiotemporal event and have connected edges to its neighbors. In contrast, VEKG captures each frame as a graph of objects with spatial information which is then summarized to EAG over temporal dimension for multiple queries. Adhikari et al. proposed NETCONDENSE [24], which merges adjacent node-pair and time-pair for time-varying graphs. The time-pair merge loses initial edge information which is preserved in EAG.

B. Video Representation and Event Detection

Techniques like visual relation detection [25] work on static data like images where relationships are annotated among objects manually, and then the model is trained to detect a pattern among objects. In contrast to this, EAG focuses on dynamic video data and detect spatiotemporal patterns among objects using event rules. In [11], the author used object tracklets over a neural network to capture object relationships in video, where relationships were encoded in the video frames manually. Lee et al. proposed Region Adjacency Graphs (RAG) [22] for videos where the same segmented regions within the image frames are connected using common boundaries. Instead of focusing on low-level features like in RAG, VEKG is built over semantic labels (objects) extracted from DNN models capturing spatiotemporal relation among them. Yadav et al. [26] focused on pattern detection like 'wildfire' from the images using crowd knowledge instead of automated pattern detection. Xu et al. [27] presented a Video Structural Description (VSD) technology for discovering semantic concepts in the video with no CEP focus.

C. Window-based Aggregation

Works like [28]–[30] focus on different window aggregation aspects like sharing, adaptivity and load-shedding of using aggregate operator like SUM, MIN, AVG etc. These works consider the incoming data stream having a simple data model like numbers and do not focus on aggregation over graph representation of video streams. Gillani et al. proposed SPECTRA [31], an RDF graph summarization over windows using an incremental indexing approach. This work differs from ours as we focus on the more general labelled spatiotemporal graph instead of RDF. SPECTRA focus summarization based on a single query while EAG summarizes multiple query results.

VIII. Conclusion and Future Work

In this work, we present a novel approach to summarize graphs of video streams over a given window for complex video pattern detection in CEP systems. The summarized EAG graph not only reduces the redundant nodes and edges but also preserve all the spatiotemporal information without any information loss. The efficacy of spatiotemporal pattern detection is shown using different traffic management related query operators. EAG reduces greater than 97.5% of nodes and 96 % of edges and requires 68.35% less storage space as compared to VEKG graph streams. The summarization overhead for EAG is minimal, with 82% less search time for multiple queries. In future work, we will optimize EAG graphs for different queries over sliding windows.

Acknowledgement

The work was supported with the financial support of the Science Foundation Ireland grant 13/RC/2094.

References

[1] S. A. Alvi, B. Afzal, G. A. Shah, L. Atzori, and W. Mahmood, "Internet of multimedia things: Vision and challenges," Ad Hoc Networks, 2015.

[2] "Somebody's Watching You:New York Installation Surveillance in 2017." [Online]. Available: https://bit.ly/2Um1hT5.

[3] H. Zhang et al., "Live Video Analytics at Scale with Approximation and Delay-Tolerance," in USENIX NSDI, 2017.

[4] M. Dayarathna and S. Perera, "Recent Advancements in Event Processing," ACM Comput. Surv., 2018.

[5] "Esper." [Online]. Available: http://www.espertech.com/esper/.

[6] "FlinkCEP - Complex event processing for Flink." [Online]. Available: https://ci.apache.org/projects/flink/flink-docs-stable/dev/libs/cep.html.

[7] Q.-C. To, · Juan Soto, and · Volker Markl, "A survey of state management in big data processing systems," VLDB J., 2018.

[8] D. G. Lowe, "Object Recognition from Local Scale-Invariant Features," in ICCV, 1999.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in IEEE CVPR, 2016.

[10] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in ICCV, 2017.

[11] J. Johnson et al., "Image Retrieval using Scene Graphs," in CVPR, 2015.

[12] P. Yadav and E. Curry, "VEKG: Video Event Knowledge Graph to Represent Video Streams for Complex Pattern Matching," in IEEE Graph Computing, 2019.

[13] P. Yadav and E. Curry, "VidCEP: Complex Event Processing Framework to Detect Spatiotemporal Patterns in Video Streams," in IEEE BigData, 2019.

[14] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in ICIP, 2017.

[15] J. Chen, A. G. Cohn, D. Liu, S. Wang, J. Ouyang, and Q. Yu, "A survey of qualitative spatial representations," Knowl. Eng. Rev., Jan. 2015.

[16] E. Clementini, P. Di Felice, and P. van Oosterom, "A small set of formal topological relationships suitable for end-user interaction," Springer, Berlin, Heidelberg, 1993, pp. 277–295.

[17] D. Hernández, "Relative Representation of Spatial Knowledge: The 2-D Case," in Cognitive and Linguistic Aspects of Geographic Space, 1991.

[18] B. G. Mobasseri and P. Krishnamurthy, "Spatio-temporal object relationships in image sequences using adjacency matrices," Signal, Image Video Process., Jun. 2012.

[19] B. George and S. Shekhar, "Time-Aggregated Graphs for Modeling Spatio-temporal Networks," J. Data Semant. XI, 2008.

[20] "NetworkX." [Online]. Available: https://networkx.github.io/.

[21] L. Wen et al., "UA-DETRAC: A New Benchmark and Protocol for Multi-Object Detection and Tracking," Nov. 2015.

[22] J. Lee, J. Oh, and S. Hwang, "STRG-Index: Spatio-Temporal Region Graph Indexing for Large Video Databases," in ACM SIGMOD, 2005.

[23] J. Kwon and K. M. Lee, "A unified framework for event summarization and rare event detection," CVPR, 2012.

[24] B. Adhikari, Y. Zhang, A. Bharadwaj, and B. A. Prakash, "Condensing Temporal Networks using Propagation," in SIAM, 2017.

[25] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei, "Visual relationship detection with language priors," in ECCV, 2016.

[26] P. Yadav, U. U. Hassan, S. Hasan, and E. Curry, "The Event Crowd: A novel approach for crowd-enabled event processing," in DEBS, 2017.

[27] C. Hu, Z. Xu, Y. Liu, and L. Mei, "Video structural description technology for new generation video surveillance systems," Front. Comput. Sci, 2015.

[28] P. Carbone, J. Traub, A. Katsifodimos, S. Haridi, V. Markl, and † Kth, "Cutty: Aggregate Sharing for User-Defined Windows," in CIKM, 2016.

[29] A. Arasu and J. Widom, "Resource Sharing in Continuous Sliding-Window Aggregates *," in VLDB, 2004.

[30] N. Tatbul and S. Zdonik, "Window-aware Load Shedding for Aggregation Queries over Data Streams *," 2006.

[31] S. Gillani, G. Picard, and F. Laforest, "SPECTRA: Continuous Query Processing for RDF Graph Streams Over Sliding Windows," SSDBM, 2016.