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

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 focuses 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.

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 images 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)).

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 = \( \{V, E, A_v, R_k, \lambda_v, \lambda_E\}\) where
- \( V \) = set of object nodes \( O \)
- \( E \) = set of edges such \( E \subseteq V \times V \)
- \( A_v \) = set of properties mapped to each object node such that \( O = \{\text{id}, \text{attributes}, \text{label}, \text{confidence}, \text{features}\}\)
- \( R_k \) = set of spatiotemporal relations classes
- \( \lambda_v, \lambda_E \) = class labelling functions

\[ \lambda_v: V \to O \text{ and } \lambda_E: E \to R_k \]

**Definition2 (VEKG Stream)**: A Video Event Knowledge Graph stream is a sequence ordered representation of VEKG such that \( \text{VEKG} (S) = \{(\text{VEKG}_1, t_1), (\text{VEKG}_2, t_2), \ldots, (\text{VEKG}_n, t_n)\}\) where \( t \) = timestamp such that \( t_i < t_{i+1} \).

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.

**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 (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.

   ![](image331x690-358x709)

   Fig. 6. Spatial relations between objects

2) **Topology-Based Spatial Relation (S_9)**: 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 (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.

\[
bsf \rightarrow s_{ij}(O_1, O_2) = \begin{cases} 
0 & \text{if spatial relation is false} \\
1 & \text{if spatial relation is true}
\end{cases}
\]

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:

\[
msf \rightarrow M(O_1, O_2) | c | r
\]

where \(c \in \text{comparison operator} (<, >, \leq, \geq, =)\) and \(r \in \mathbb{R}\) (real number) and \(M \in \text{metric function like distance and count}\). 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 VEGK 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,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.

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 \(t1\) and \(t2\), such that \(t1<t2\). We see that the relative position of the object \(O_2\) was ‘back’ of \(O_1\) at \(t1\), which becomes ‘front’ at \(t2\). This signifies that object \(O_2\) crosses the \(O_1\) in \(t1\)st time instance. Thus, as per eq.1, we can write the ‘Pass By’ operator as:

\[
[\text{back}(O_1, O_2)] \rightarrow [\text{front}(O_1, O_2)] \quad [\mathbb{E}_{\text{fam}}]
\]

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:

\[
\exists (t_i) \in \mathbb{T} \quad [bsf(S_0(O_1, O_2)x^{t_i}) \cup bsf(S_0(O_1, O_2)x^{t_i})] \quad [\mathbb{E}_{\text{fam}}]
\]

\[
= \begin{cases} 
0 & \text{if Pass By} \\
1 & \text{if no Pass By}
\end{cases}
\]

where \(x \in \text{back} \rightarrow \text{front direction}\) and \(t_{wa} \leq t_i \text{and} t_{tij} \leq t_a\)

In equation 2, \(bsf(S_0(O_1, O_2)x^{t_i})\) means the boolean spatial function over spatial direction (\(S_0\)) on the object \((O_1, O_2)\) at time \(t_i\) where direction we are looking is in back-front. This function evaluates as ‘Is \(O_1\) back of \(O_2\) in back front direction’, which is true, so it will return 1. Similarly, \(bsf(S_0(O_1, O_2)x^{t_i})\) is the calculation for the next frame at time \(t_{tij}\). In this case the relative position of the object \(O_2\) become front of the object \(O_2\), so ‘\(O_1\) back of \(O_2\)’ become false and returns 0. If we do an XNOR \((\cup)\) of these two values, i.e. 1 at \(t_i\) and 0 at \(t_{tij}\), 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 \([\mathbb{E}_{\text{fam}}, t_0]\). 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.
is defined as ‘stream over windows into an Event Aggregated Graph (EAG) to handle different spatiotemporal queries.’

\[ \text{QUERY} \]

\[ \text{follows} \]

\[ \text{lane change} \]

\[ \text{traffic} \]

\[ \text{volume} \]

\[ \text{query operator} \] for a specific location (Fig. 10). 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 \( EAG = \{ V_{Node}, E_{Edge}, W_{Time}, f_q(t, w, \lambda_0, \lambda_1) \} \) where:

- \( V_{Node} = \{ \) set of unique object nodes \( O_i \) from VEKG stream over a window
- \( E_{Edge} = \{ \) set of edges such \( E_{Edge} \subseteq V_{Node} \times V_{Node} \)
- \( W_{Time} = \) window time length
- \( f_q = \) different query mappings over \( E_{Edge} \)
- \( t, w = \) time and spatiotemporal weight mappings over \( f_q \)
- \( \lambda_0, \lambda_1 = \) class labelling functions \( \lambda_0: V_{Node} \rightarrow O \) and \( \lambda_1: E_{Edge} \rightarrow f_q \)
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) \(\odot\) E(t2), E(t2) \(\odot\) E(t3), and E(t3) \(\odot\) E(t4) resulting in \([X,1,0,\ldots]\) 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 \(o_2\) is left of \(o_1\) 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 \(\text{msf}\) is applied or expressed using other formats depending on operator rule.

**Algorithm 1: Event Aggregated Graph**

**Input:**
Window \((W) = \{\\text{VEKG}_1, \text{VEKG}_2, \text{VEKG}_3, \ldots, \text{VEKG}_n\}\)
Query \((Q) = \{0, 0, 0, \ldots, 0\}\)

**Output:**
Event Aggregated Graph (EAG) over Win

**Procedure:**
EAG = initializeEAG(V, E, f_q, t, w)
for each \(\text{VEKG}_i\) in Win do
\(V = V \cup\) getnodes(\(\text{VEKG}_i\))
end
for each \(Q_i\) in Query do
\(w = \) SpatiotemporalOperator(\(Q_i(\text{VEKG}_i)\))
\(f_q = f_q(w)\)
EAG = UpdateEAG(V, E)
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 \([x, 1, 0, \ldots]\) 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 \(n = |V_{\text{Node}}|\), i.e. number of nodes. EAG has two extra dimensions, i.e. time and query. If \(W_{\text{Time}}\) 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 \((|V_{\text{VEKG}}|)\) and summarized graph \(|V_{\text{EAG}}|\) 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).

\[
\text{RIN} = \frac{|V_{\text{VEKG}}| - |V_{\text{EAG}}|}{|V_{\text{VEKG}}|} \quad \text{RIE} = \frac{|E_{\text{VEKG}}| - |E_{\text{EAG}}|}{|E_{\text{VEKG}}|}
\]

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 compared 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-
-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 summarised 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.

\[ F = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \]  

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