TQVS: Temporal Queries over Video Streams in Action

Yueting Chen
York University
ytchen@eecs.yorku.ca

Xiaohui Yu
York University
xhyu@yorku.ca

Nick Koudas
University of Toronto
koudas@cs.toronto.edu

ABSTRACT

We present TQVS, a system capable of conducting efficient evaluation of declarative temporal queries over real-time video streams. Users may issue queries to identify video clips in which the same two cars and the same three persons appear jointly in the frames for say 30 seconds. In real-world videos, some of the objects may disappear in frames due to reasons such as occlusion, which introduces challenges to query evaluation. Our system, aiming to address such challenges, consists of two main components: the Object Detection and Tracking (ODT) module and the Query Evaluation module. The ODT module utilizes state-of-art Object Detection and Tracking algorithms to produce a list of identified objects for each frame. Based on these results, we maintain select object combinations through the current window during query evaluation. Those object combinations contain sufficient information to evaluate queries correctly. Since the number of possible combinations could be very large, we introduce a novel technique to structure the possible combinations and facilitate query evaluation. We demonstrate that our approach offers significant performance benefits compared to alternate approaches and constitutes a fundamental building block of the TQVS system.

ACM Reference Format:
Yueting Chen, Xiaohui Yu, and Nick Koudas. 2020. TQVS: Temporal Queries over Video Streams in Action. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3318464.3384693

SIGMOD'20, June 14–19, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6735-6/20/06…$15.00
https://doi.org/10.1145/3318464.3384693

1 INTRODUCTION

We have witnessed an explosion of video data over recent decades. Video accounted for 75% of internet traffic in 2017, and is expected to make up 82% in 2022, with close to 1 million minutes of video crossing the Internet per second[1]. According to the Wall Street Journal there will be a billion cameras on the streets by 2021 [5]. However, the way of querying video streams is still primitive, and requires lots of human intervention. In scenarios such as surveillance applications, humans often have to visually inspect a large amount of video to identify persons or objects of interest. It is therefore crucial to develop systems that could support the extraction of meaningful information from videos in real time, utilizing declarative queries, in a way akin to how people are interacting with RDBMS today.

On the other hand, recent breakthroughs in Deep Learning (DL) [2, 12] have made it possible to achieve highly accurate results in tasks such as image classification [11], object detection [13, 14] and object tracking [10, 15, 17], providing the building blocks to make large-scale video query processing a reality. A video clip can be considered as a list of frames that are displayed on a fixed fps (frames per second) rate. Applying object detection algorithms on each frame, one can obtain a set of objects (bounding boxes) and the associated class labels (e.g., car, bus, etc.). The detected objects can also be assigned unique identifiers (ids) by applying object tracking algorithms between frames. Such identifiers can be utilized in declarative query processing on videos. Recently various works focus on processing and understanding videos [4, 6–9, 18] based on DL models and algorithms. Although current systems enable queries at a frame level, our focus is to enable queries across frames utilizing unique object identifiers and effectively enable the temporal dimension into our query framework.

We focus on scenarios such as surveillance cameras where co-occurring objects are of interest. For example, we wish to identify video frames where two men appear jointly for a number of frames (Figure 1a), or where persons and a car appear jointly in a frame sequence of a set duration (Figure 1b). This type of queries can be expressed in Conjunctive

[1] https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

(a) Two Men

(b) A Car Thief

Figure 1: Video Examples

Normal Form (CNF). Such queries can be issued in a declarative manner on the video stream. Alerts are triggered when one of the queries evaluated to true and the corresponding video frame sequence is also returned. For example, an alert can be triggered when a few cars are blocking a traffic intersection for more than one minute. Such queries can also be utilized to retrieve video clips for forensic investigations after accidents.

One critical issue in formulating and processing queries in these scenarios is that objects may disappear from some of the frames due to various reasons, such as occlusions, or even object detection/tracking errors. For example, in Figure 1b, the person may walk out of the scene, or disappear behind the car for a while. To support such semantics, we propose [1] to evaluate CNF queries over a sliding window[2] of fixed size, w, along with a duration threshold, d. For example, one may query objects that appear jointly for 30 seconds within 40 seconds. In this case, "30 seconds" is the duration threshold, while "40 seconds" is the size of the window. The result of such queries is a set of frame sets, where each frame set satisfies the following conditions:

- $n \geq d$, where n is the number of frames in the set.
- If O' is the set of objects that co-occur in all the frames in the frame set, the evaluation result of the given CNF expression on O' is also TRUE.

To improve efficiency in the evaluation of such queries, in our work we have proposed to reuse the intermediate results between different windows. We introduced the concept of *Marked Frame Sets* [1] to reduce the number of intermediate results considered.

We present TQVS, which incorporates our developments and is capable to efficiently execute temporal queries over videos. In this demo, we are able to demonstrate:

- The system architecture of TQVS and its modules, including object detection and tracking, query parsing, and query evaluation.
- The current user interface of TQVS. Users are able to select between video sources, queries, window sizes, and query duration thresholds. Based on the user input, queries are evaluated and results are displayed.
- The visualization of query results. We visualize the result of query evaluation with video players to demonstrate the set of frames and objects that satisfy the query. We also provide a performance comparison of our method and baseline approaches for the chosen video source and query.

[2]Other types of windows can be supported as well.

2 OVERVIEW OF THE APPROACH

A query defined by users contains three parts: the CNF expression to be evaluated, the size of the query window (w) and the duration parameter (d). Before processing queries, object detection [3] and tracking algorithms [17] are executed on given videos to obtain the identified object set for each frame. We assume all objects can be assigned with class labels and unique ids in all frames. Queries are evaluated on sliding windows with a step size of 1 by default. Based on the identified object sets, queries can be evaluated and answered in the following steps:

(1) **Computing intermediate results**, where the system enumerates all candidate object sets along with the set of frames in which they appear. These candidate object sets correspond to the sets of objects that co-occur in a certain number of frames, and they have to be *maximal*, meaning that the objects in any proper superset of a candidate set will not have the same number of co-occurrences as those objects in that candidate set do.
(2) **Evaluation**, where the system evaluates queries based on the candidate object sets and associated frame sets obtained in the previous step.

Our Approach. To efficiently execute these queries, we introduce a new structure that we refer to as Marked Frame Sets, which allows us to prune redundant intermediate results. We also introduce a data structure called Strict State Graphs (SSG), $G =< \mathbb{S}, \mathbb{E} >$, to capture the relationships between intermediate results, where $\mathbb{S}$ is the set of intermediate results, and $\mathbb{E}$ is the set of edges. Let $s \in \mathbb{S}$ be an intermediate result, and $S_s \subset \mathbb{S}$ be a set of intermediate results containing all vertices that are adjacent to s (i.e. $\forall s' \in S_s, (s, s') \in \mathbb{E}$). We use O_s to denote the object set of s. The following properties are satisfied on the graph:

- $\forall s_i, s_j \in S_s$, we always have $O_{s_i} \cap O_{s_j} \neq O_{s_i}$ and $O_{s_i} \cap O_{s_j} \neq O_{s_j}$.
- Let O_i be another object set. If $O_i \cap O_s = \emptyset$, then $\forall s' \in S_s, O_i \cap O_{s'} = \emptyset$.

To summarize, our approach uses a Strict State Graph to index intermediate results, while using Marked Frame Sets to prune redundant ones. Then, queries are embedded into a CNF query evaluation framework, utilizing indexing structures [16] to improve efficiency further. Intermediate

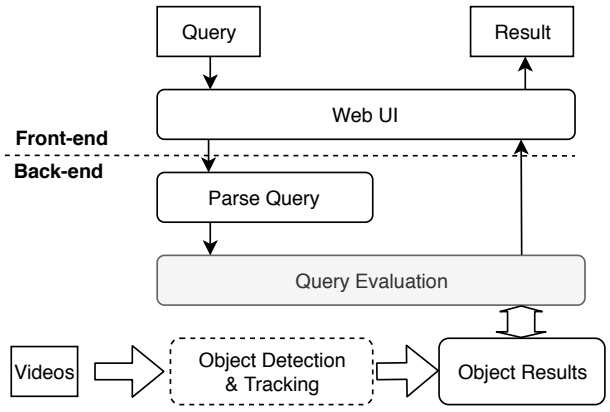

Figure 2: System Architecture

results can further be pruned based on the query evaluation results. More details are available elsewhere [1].

3 SYSTEM ARCHITECTURE

The architecture of TQVS is shown in Figure 2. The system consists of two main parts: the front-end, which assists users to build their queries and displays execution results, and the back-end, which is responsible for extracting objects from videos, as well as parsing and evaluating queries.

Videos are registered to the back-end directly. State-of-the-art object detection and tracking algorithms are implemented as the Object Detection and Tracking (ODT) module to identify and assign an identifier to each unique object in videos. The module is designed to be plug-and-play, which allows new algorithms to be integrated in the future. Queries constructed by the user via the front-end are parsed first, and then evaluated in the Query Evaluation module based on the object results obtained from the ODT module.

4 DEMONSTRATION

The system user interface is shown in Figure 3, which consists of three main panes: Video Pane (Area A), Query Pane (Area B), and Result Pane (Area C). Video Pane provides a list of video clips for users to choose from, which can be previewed in the mini player after selection. Query Pane accepts user input to form the query, including the CNF expression, the window size, and the duration threshold. Result Pane shows the result after executing the query, where the top area (Area C.a) presents the performance comparison between a baseline method and our method, while the main area (Area C.b) depicts the query result.

Two typical use cases that demo participants will be exposed to are outlined below.

Use Case 1: The user would like to identify a frame sequence where more than two cars (the same cars) appear jointly. Suppose we select the video clip *v1* as the input video and we wish to select video frames where two cars appear jointly for at least 240 frames in a window size of 300 frames (the number of frames are adjustable parameters). The user can fill in the CNF expression, the window size, and the duration threshold as shown in Figure 3. After clicking the Execute button, the user will observe the result displayed on the right.

Since the query is evaluated over windows, we use a sliding window starting from the first frame, with its maximum size set to the given window size. By default, results are displayed according to the frame where they are first generated. Users can navigate between results from different windows using the Window Selector (Area C.c). Two other options are also available: "Show All" and "Union Results". If the "Show All" option is checked, all results will be displayed in the table with the frame ID where they are first generated. The "Union Results" option only works with "Show All" enabled; when selected, results from different windows will be merged and displayed based on the entire video clip.

In the result table, each record represents a single result that satisfies the query, and the size of the associated frame set is displayed. Thus the results are a set of video sequences (clips). Each record can be viewed in a video player (shown in Figure 4) by clicking the view icon. Objects that contribute to the query are marked in the video by their corresponding bounding boxes. Selected frames are also highlighted in the progress bar at the bottom. Users can choose to play only the selected frames or the entire video clip.

Use Case 2: The user would like to select video frames where there are at least two persons (the same persons) and a car (the same car) appearing jointly for a number of frames. The query condition can be expressed in CNF, $count(person) \geq 2$ *and* $count(car) \geq 1$. Assume the window size is limited to 200 frames and the duration threshold is set to 150 (adjustable parameters). We can execute such a query and obtain the evaluation result. Suppose we are more interested in the object combinations that satisfy the above condition rather than the selected frames. With both options "Show All" and "Union Results" checked, all the possible object combinations along with the frames in which they appear will be listed in the result table. Similar to the above use case, we can click the "View" icon associated with each result record to visualize the result on the original video clip.

REFERENCES

[1] Yueting Chen, Xiaohui Yu, and Nick Koudas. [n.d.]. Evaluating Temporal Queries over Video Streams. In Submission.
[2] Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. 2016. *Deep Learning*. MIT Press.
[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. 2017. Mask R-CNN. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. 2980–2988.

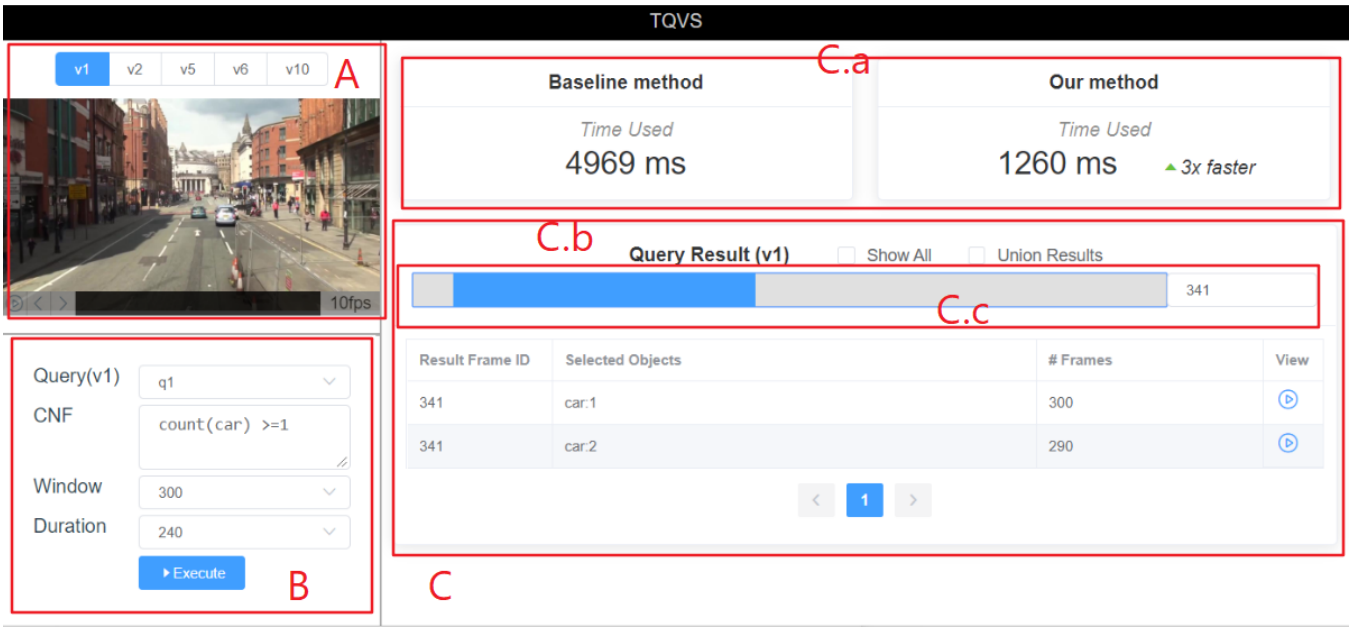

Figure 3: User Interface

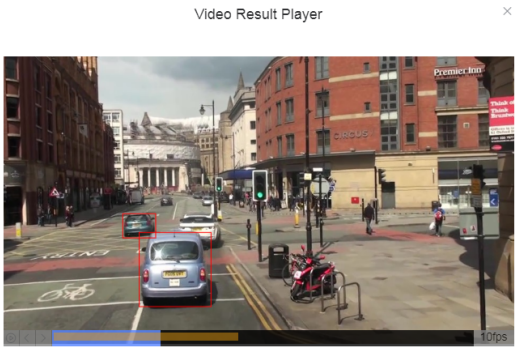

Figure 4: Result Player

[4] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B. Gibbons, and Onur Mutlu. 2018. Focus: Querying Large Video Datasets with Low Latency and Low Cost. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. Carlsbad, CA, 269–286.

[5] Wall Street Journal. 2019. https://www.wsj.com/articles/a-billion-surveillance-cameras-forecast-to-be-watching-within-two-years-11575565402.

[6] D. Kang, P. Bailis, and M. Zaharia. [n.d.]. BlazeIT: Fast Exploratory Video Queries Using Neural Networks. In *https://arxiv.org/abs/1805.01046*.

[7] Daniel Kang, Peter Bailis, and Matei Zaharia. 2019. Challenges and Opportunities in DNN-Based Video Analytics: A Demonstration of the BlazeIt Video Query Engine. In *CIDR 2019, 9th Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 13-16, 2019, Online Proceedings*.

[8] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: Optimizing Neural Network Queries over Video at Scale. *Proc. VLDB Endow.* 10, 11 (Aug. 2017), 1586–1597.

[9] Nick Koudas, Raymond Li, and Ioannis Xarchakos. 2020. Video Monitoring Queries. In *Proceedings of IEEE ICDE*.

[10] Sebastian Krebs, Bharanidhar Duraisamy, and Fabian Flohr. 2017. A survey on leveraging deep neural networks for object tracking. In *20th IEEE International Conference on Intelligent Transportation Systems, ITSC 2017, Yokohama, Japan, October 16-19, 2017*. 411–418.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[12] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[13] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99.

[14] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556 (2014).

[15] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. 2015. Visual Tracking with Fully Convolutional Networks. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. 3119–3127.

[16] Steven Euijong Whang, Hector Garcia-Molina, Chad Brower, Jayavel Shanmugasundaram, Sergei Vassilvitskii, Erik Vee, and Ramana Yerneni. 2009. Indexing boolean expressions. *Proceedings of the VLDB Endowment* 2, 1 (2009), 37–48.

[17] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. 2017. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 3645–3649.

[18] Ioannis Xarchakos and Nick Koudas. 2019. SVQ: Streaming Video Queries. In *Proceedings of ACM SIGMOD, Demo Track*.