SVQ++: Querying for Object Interactions in Video Streams

Daren Chao
University of Toronto
drchao@cs.toronto.edu

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

Ioannis Xarchakos
University of Toronto
xarchakos@cs.toronto.edu

ABSTRACT

Deep neural nets enabled sophisticated information extraction out of images, including video frames. Recently, there has been interest in techniques and algorithms to enable interactive declarative query processing of objects appearing on video frames and their associated interactions on the video feed. SVQ++ is a system for declarative querying on real-time video streams involving objects and their interactions. The system utilizes a sequence of inexpensive and less accurate models (filters), called Progressive Filters (PF), to detect the presence of the query specified objects on frames, and a filtering approach, called Interaction Sheave (IS), to effectively prune frames that are not likely to contain interactions. We demonstrate that this system can efficiently identify frames in a streaming video in which an object is interacting with another in a specific way, increasing the frame processing rate dramatically and speed up query processing by at least two orders of magnitude depending on the query.

ACM Reference Format:
Daren Chao, Nick Koudas, and Ioannis Xarchakos. 2020. SVQ++: Querying for Object Interactions in Video Streams. 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/10.1145/3318464.3384701

1 INTRODUCTION

Recent advances in computer vision - in the form of deep neural networks - have made it possible to query increasing volumes of video data with high accuracy. However, deep neural network inference is computationally expensive at scale. There are still many limitations to apply deep neural models at scale in real-world scenarios imposing real-time requirements.

To understand the visual world, a machine must not only recognize individual object instances but also how they interact. Most of the state-of-the-art video query processing models focus on the queries about the number of classified objects and their location relationships [5, 6, 10]. Semantic information, such as human-object interaction information, is often ignored. Fortunately, several state-of-the-art algorithms in the fields of object detection, object classification and object tracking in images and videos have been proposed [2, 3, 7, 9]. They provide the ability to classify objects and detect object locations in a frame as well as track objects from frame to frame with adequate accuracy.

Figure 1: Examples of interactions.

Video monitoring and surveillance applications are of particular interest in our system, where a static camera records activity in its receptive range. Several recent works [5, 6, 10] focus in these domains, to provide declarative queries based on streaming video feeds. Our work complements and enhances this research line with a focus on efficiently executing interaction query primitives, i.e., capturing interactions among query objects. Many interesting queries become possible when we can query object interactions. For example, automatically detecting frames in which a human holds a gun or a human breaks a window would be of great interest in surveillance and security applications. Figure 1 presents examples of interactions between a human and a ball (videos from Kinetics Dataset [1]). The corresponding query in SQL for the interaction at the frame of Figure 1(a) would be:

```
SELECT cameraID, frameID, C1 (F1 (HumanBox1)) AS
    HumanType1, C2 (F2 (Ballbox1)) AS BallType1,
```


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

ACM ISBN 978-1-4503-6735-6/20/06…$15.00
https://doi.org/10.1145/3318464.3384701

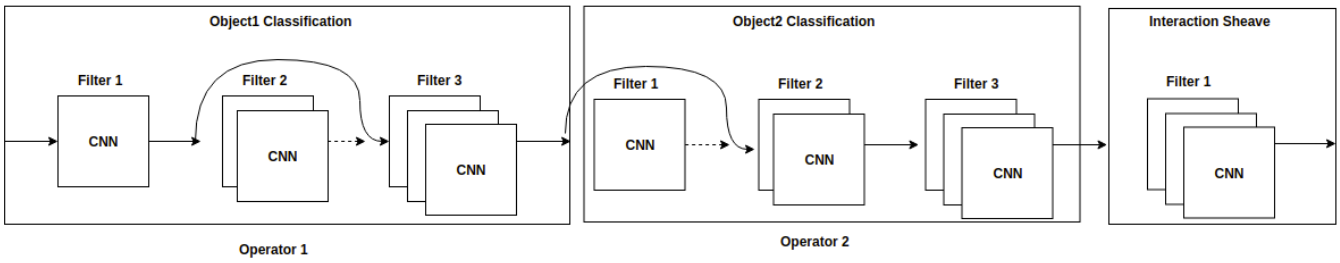

Figure 2: Example of the overall architecture.

```
FROM (PROCESS inputVideo PRODUCE cameraID, frameID
    , HumanBox1 USING HumanDetector, BallBox1
    USING BallDetector)
WHERE HumanType1 = human AND BallType1 = baseball
    AND INTERACTION(HumanType1, BallType1) = THROW
```

The query employs two classifiers (C_i) to detect a *human* and a *baseball*, using features F_i extracted from the frame and checking whether the objects, once identified, are related via a THROW interaction. Obviously, from an execution perspective, it makes sense to invoke the INTERACTION predicate once the operands have been identified on the frame. We seek to automatically identify frames in a video stream where query objects interact in a specific way. This schema of queries can be applied to most objects.

The main emphasis of our system is, given a query involving objects and their interaction on a video stream, to deploy algorithms to efficiently determine which frames are expected to be part of the query answer and filter out all irrelevant frames. Therefore, we try to increase the frame processing rate as frames that are considered irrelevant will not be processed further but skipped quickly. Even without filtering, the application of deep neural networks for object and interaction detection entails false positives/negatives. We will also show the effect that filtering has on the false positive/negative rate of the techniques on this demo.

To overcome these challenges, we employ a set of algorithms that dramatically increase the frame processing rate when executing the query while maintaining high accuracy. In particular we utilize:

- **Progressive Filters.** Progressive filters are a set of filters of increasing cost (i.e., an increasing number of network layers). For a particular object type, progressive filters use the selectivity of the filters to derive the optimal filter sequence. The filters should be applied to minimize the total cost of processing the video stream.
- **Triggering Re-optimization.** Since we observe that the statistical properties of certain types of objects vary over time on a video stream, we adopt a dynamic version of the Kolmogorov-Smirnoff test [8] that can trigger progressive filters algorithm when the statistical properties of the filter selectivity change. We are effectively adjusting the filter sequence, in anticipation of stable (predictable) object statistics, until the next re-optimization. Such a strategy can improve accuracy while maintaining a high frame processing rate.
- **Interaction Sheave.** Interaction sheave is a filtering mechanism for object interaction queries. It can inspect the spatial location of objects on frames and filter out frames that although contain objects relevant to the query but not promising to encompass the suitable interaction among the query objects.

In this demonstration, we showcase SVQ++, which is capable of handling a wide range of interaction queries. Section 2 presents the architecture of our system. Section 3 presents a description of this demonstration. Section 4 concludes this demo and discusses future work in this area.

2 SYSTEM ARCHITECTURE

Deep learning filters is an extensible module that implements our proposed filter predicates, which encompasses popular recent deep learning algorithms for object and interaction detection. These algorithms are embedded in the parsed query representation and relayed to the query execution engine. The execution module utilizes popular deep learning frameworks to execute the query with the assistance of available GPUs. Frames that pass the filters instantiated in the query are subsequently checked with deep learning predicates and then routed to the front end for display.

An example of the architecture of our proposed filter technologies is shown in Figure 2. There are two operators, each consisting of a sequence of filters. In this example, the application of progressive filters for operator 1 determined that filters 1 and 3 are currently in use. Similarly, for operator two, filters 2 and 3 are in use. Frames that successfully pass the operators with a positive determination[1] that they contain the objects of interest, are tested by the heave. The basic

[1]Take the first operator as an example, a frame may get a positive determination by filter 1; in that case, there is no need for further processing by other filters in this operator. Filter 3 (a more accurate and expensive model in this example) is involved only if Filter 1 is unable to make a positive or negative determination for the frame.

observation is that the interaction (e.g., human throwing a baseball) typically takes place at a specific spatial region of the frame involving the first object (i.e., the human). Only when the second object is located within a spatial region relative to the first object, the frame is relayed to an expensive object interaction model for further processing. Thus, by processing the objects spatially, we obtain a filtering mechanism for interaction.

The main innovations in our system is how to order filters within an object detection operator effectively. Typically, a query will check for the presence of more than one object (as in the case of the sample SQL query presented). The progressive filters can derive the least cost sequence of filters given the current selectivities of query objects, avoiding costly object detection and localization operations, and minimizes the cost per frame for object detection. For the filters employed, we can maintain their selectivities up to date by observing the result of each frame tested by the filter. For the filters that are not part of the optimal solution, we periodically (every few seconds) route a frame from the input sequence through them and obtain a selectivity estimate to trigger re-optimization. The basic idea behind re-optimization is to treat the selectivity of filters as a statistical population and employ a dynamic version of the popular Kolmogorov-Smirnov test [8]. For frames that pass through the filter sequences determined by progressive filters for each object specified by the query, we have confidence that they contain the required objects. These models will derive a bounding box that encloses the location of each object as specified by the query on the frame. We then test the frame whether it relates the objects via the query specified interaction.

3 DEMONSTRATION SETUP

This Demo will highlight the interaction querying process and provide various knobs for video source selection, performance monitoring and query results display and comparisons. Our layout will allow users to express human-object interaction queries and compare the time and accuracy performance of our approach with a baseline approach that employs an interaction detection model at each frame to answer the query [3].

Figure 3 presents the current state of the front end. SQL queries are defined through query definition which is shown in the left sidebar (Area A) and enhanced with UDFs to manipulate video object primitives. The Demo provides video clips for interaction querying. Each video clip corresponds to a specific predicate. For the baseball video clip, for instance, the predicate will be *human THROW baseball*. The Demo also provides two modes - normal and slow mode. The normal mode shows the real video processing speed of our system, while it may run too fast to show the results of interaction detection. For better presentation, it also provides a slow mode to show the resulting video which is slower than the real processing speed.

The generated SQL queries, which are shown on the upper left center (Area B), are dispatched to the back end which is responsible for parsing the query and incorporating the supported querying predicates and progressive filters. It currently incorporates our proposed algorithms for filtering frames (based on humans, objects classes and interaction), allowing to express semantically meaningful video frame queries in an interactive fashion. The impact of progressive filters will be analyzed based on the total query processing time (when compared to a query that does not make use of the progressive filters but instead executes the query in a brute force manner [3]) as well as the resulting frame processing rate.

The filters' ordering per batch is presented on the upper right corner (Area C). It will be updated at the end of each batch. The three boxes with different colors represent different operators and a node in a box represents a filter in the operator. As introduced in section 2, the application of algorithm progressive filters may employ a subset of filters. The links represent the ordering of filters that are used for the current batch. For the filters that are not part of the optimal solution, we periodically route a frame from the input sequence through them and obtain a selectivity estimate as well. The links will be bold if they have been changed from the previous batch to the current. Figure 4 presents the filters ordering module. The red box and nodes represent human operators, whose Filter 1, 3, 4 and 6 are used. Similarly, the yellow and blue represent the object operator and interaction sheave respectively.

By clicking on the Optimization button, our approach will be invoked, while by clicking the Brute Force button, state of the art object detectors [4] and action recognition [3] models will evaluate the respective input video. The videos of both our approach and full model's approach will produce bounding boxes for humans and objects which participate in an interaction. Additionally, a heat-map on frames where a specific interaction exists between the human and the object will be displayed. Areas D and G present the actual query results for our proposed techniques and brute force respectively. F1 score and precision/recall results are reported in Areas E (for our algorithms) and H (for brute force), complemented with detailed performance numbers (response times and frame rates) in areas F and I respectively.

4 CONCLUSION AND FUTURE WORK

SVQ++ showcases our ongoing work on querying for object interactions on streaming videos. The system we proposed constitutes a robust approach to process video streams for

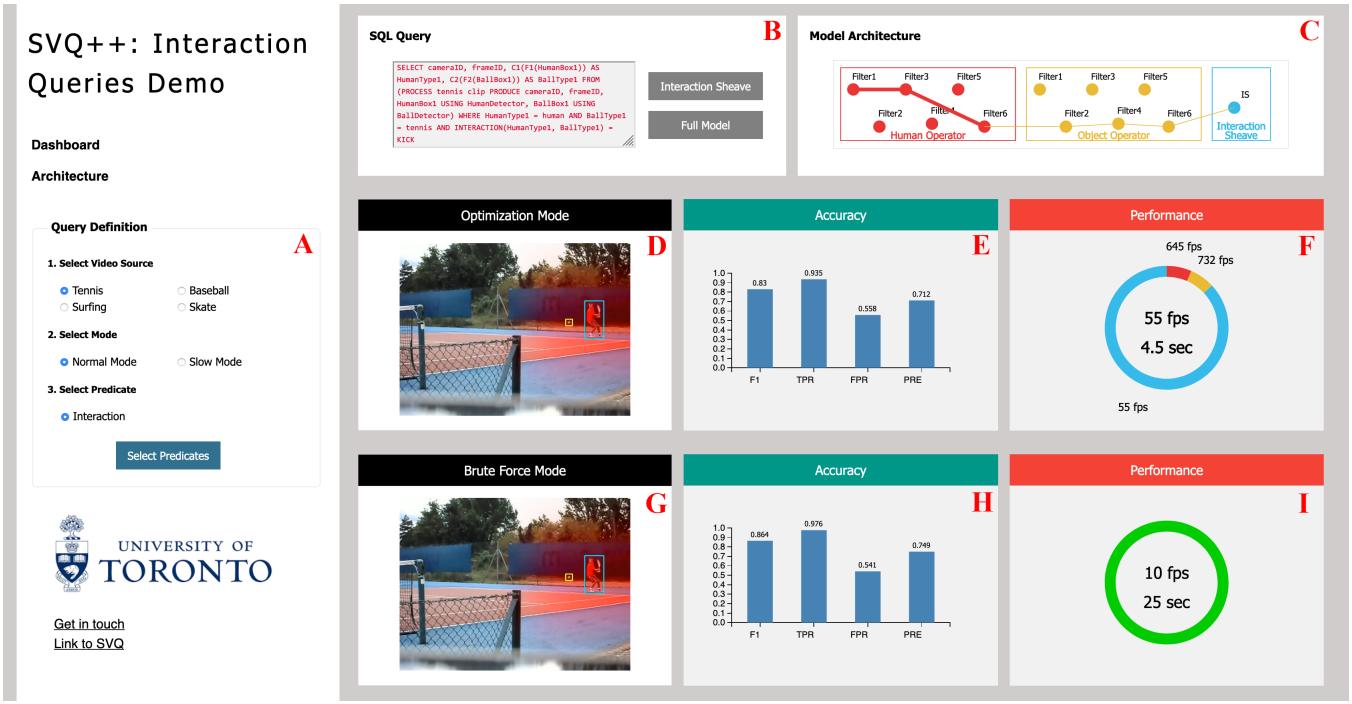

Figure 3: Front end of the SVQ++.

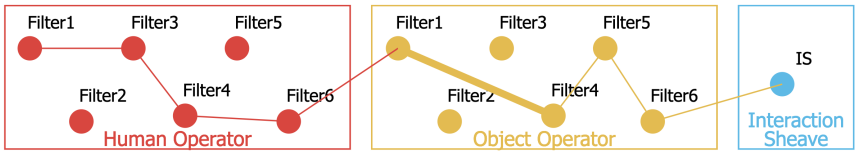

Figure 4: Example of the filters ordering module.

query specified object interactions, achieving a very high frame processing rate. This work sheds light on the declarative query process for video streams. New query types involving spatial and temporal predicates require intensive future research. The extension of the filters for crowd counting and estimation scenarios is also in our future plans.

REFERENCES

[1] João Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. 2019. A Short Note on the Kinetics-700 Human Action Dataset. *CoRR* abs/1907.06987 (2019). arXiv:1907.06987 http://arxiv.org/abs/1907.06987

[2] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 580–587.

[3] Georgia Gkioxari, Ross Girshick, Piotr Dollár, and Kaiming He. 2018. Detecting and recognizing human-object interactions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8359–8367.

[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick. 2017. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 2980–2988.

[5] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. Noscope: optimizing neural network queries over video at scale. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1586–1597.

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

[7] 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.

[8] Frank J Massey Jr. 1951. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association* 46, 253 (1951), 68–78.

[9] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. http://arxiv.org/abs/1409.1556

[10] Ioannis Xarchakos and Nick Koudas. 2019. SVQ: Streaming Video Queries. In *Proceedings of the 2019 International Conference on Management of Data (SIGMOD '19)*. 2013–2016.